KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

KAPALI ORTAMLAR İÇİN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARI İLE MOBİL ROBOTLARIN NAVİGASYONU

EMRE APAYDIN

KOCAELİ 2023

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

KAPALI ORTAMLAR İÇİN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARI İLE MOBİL ROBOTLARIN NAVİGASYONU

EMRE APAYDIN

Dr. Öğr. Üyesi Alpaslan Burak İNNER		
Danışman,	**Kocaeli Üniv.**	**.....**
Prof. Dr. Hasan OCAK		
Jüri Üyesi,	**Kocaeli Üniv.**	**.....**
Doç. Dr. Adem TUNCER		
Jüri Üyesi,	**Yalova Üniv.**	**.....**

Tezin Savunulduğu Tarih: 26.05.2023

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- Bu tezin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı,

beyan ederim.

■ Bu tez çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

☐ Bu tez çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Emre APAYDIN

YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet hakları bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki makale, kitap, tebliğ, lisans, patent gibi çalışmalarda kullanımı, danışmanımın isim hakkı saklı kalmak koşuluyla ve her iki tarafın bilgisi dâhilinde bana ait olacaktır.
Tezin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.
Yükseköğretim kurulu tarafından yayınlanan ***"Lisanüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge"*** kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.

☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.

■ Tezim ile ilgili gizlilik kararı verilmemiştir.

Emre APAYDIN

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışması, mobil robot navigasyonunun harita olmadığı durumlarda pekiştirmeli öğrenme algoritmaları ile gerçekleştirilmesi amacıyla hazırlanmıştır.

Tez çalışmamda desteğini esirgemeyen, çalışmalarıma yön veren, bana güvenen ve yüreklendiren danışmanım Dr. Öğr. Üyesi Alpaslan Burak İNNER'e teşekkürlerimi sunarım.

Sonsuz sabrı ve desteği ile bana her zaman destek olan sevgili eşime teşekkür ederim. Bu çalışmayı kıymetli evlatlarım, Elif Ece'ye ve Ahmet Asaf'a ithaf ediyorum.

Mayıs – 2023 Emre APAYDIN

İÇİNDEKİLER

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ i
YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI ii
ÖNSÖZ VE TEŞEKKÜR iii
İÇİNDEKİLER iv
ŞEKİLLER DİZİNİ vi
TABLOLAR DİZİNİ viii
SİMGELER VE KISALTMALAR DİZİNİ ix
ÖZET x
ABSTRACT xi
1. GİRİŞ 1
2. GENEL BİLGİLER 8
2.1. Hareket Planlama 8
2.1.1. Hareket Planlayıcıları 8
2.1.2. Çevre Gösterimleri 8
2.1.3. Küresel Yol Planlayıcıları 9
2.1.4. Yerel Yol Planlayıcıları 10
2.2. Makine Öğrenmesi 10
2.2.1. Denetimli-Gözetimli Öğrenme 11
2.2.2. Denetimsiz-Gözetimsiz Öğrenme 12
2.2.3. Pekiştirmeli Öğrenme 12
2.3. Derin Öğrenme 13
2.4. Pekiştirmeli Öğrenme 14
2.4.1. Tarihsel Bağlam 14
2.4.2. Pekiştirmeli Öğrenme 16
2.4.3. Markov Karar Süreçleri 18
2.4.4. Değer Fonksiyonları 20
2.4.6. Politika Belirleme 23
2.4.7. Dinamik Programlama 23
2.4.8. Monte Carlo 25
2.4.9. Zamansal Fark 26
2.4.10. Keşif ve Sömürü 27
2.4.11. Q Öğrenme 28
2.4.12. Sarsa 31
2.4.13. Sarsa ve Q Öğrenmenin Karşılaştırılması 32
2.4.14. Fonksiyon Yaklaşımı 33
2.5. Derin Pekiştirmeli Öğrenme 35
2.5.1. Derin Q Öğrenme 36
2.5.2. Deterministik Politika Gradyanı 39
2.5.3. DDPG 41
2.5.4. A2C 42
2.5.5. TRPO 43
2.5.6. PPO 44
2.5.7. TD3 46
2.5.8. SAC 47
2.6. Benzetim Ortamları (Simülatörler) 49
2.6.1. Webots 50

2.6.2. Gazebo 50
2.6.3. V-rep (Copeliasim) 51
2.6.4. Microsoft Robotics Developer Studio 52
2.6.5. Robologix 53
2.6.6. AnyKode Marilou 53
2.6.7. Graspit! 53
2.6.8. MuJoCo 53
2.6.9. OpenAI-Gym 53
2.7. Literatür İncelemesi 54
3. MALZEME VE YÖNTEM 59
3.1. İşletim Sistemi 59
3.2. Robot İşletim Sistemi (ROS) 59
3.3. Gazebo 60
3.4. Turtlebot 60
3.5. Programlama Dili 61
3.6. Matlab Pekiştirmeli Öğrenme Araç Kutusu 62
3.7. OpenAI Gym ve Stable Baselines 62
3.8. Derin Pekiştirmeli Öğrenme Temsilcileri 63
3.9. Çevre-Ortam Temsilleri 63
3.10. Ödül Modelleri 64
3.11. Parametre Optimizasyonu 64
3.12. Donanım 64
4. BULGULAR VE TARTIŞMA 66
4.1. Grid Dünya 66
4.2. MiniGrid-FourRooms 70
4.4. Gazebo Empty World 73
4.5. Eğitim Süreleri 76
4.6. Test 76
5. SONUÇLAR VE ÖNERİLER 80
KAYNAKLAR 83
KİŞİSEL YAYIN VE ESERLER 93
ÖZGEÇMİŞ 94

ŞEKİLLER DİZİNİ

Şekil 1.1. Mobil Robot Navigasyon Yaklaşımlarının Sınıflandırılması 2
Şekil 1.2. Mobil Robot Navigasyon Yaklaşımlarının Yıllara Göre Gelişimi…….3
Şekil 1.3. Otonom Araçlar İçin Derin Pekiştirmeli Öğrenme Araştırmaları. 5
Şekil 2.1. Çevrenin metrik, topolojik ve hibrit gösterimleri 8
Şekil 2.2. Yapay Zeka Hiyerarşisi 11
Şekil 2.3. Derin Yapay Sinir Ağlarının Mimarisi 14
Şekil 2.4. Thorndike yapboz kutusu 15
Şekil 2.5. Pekiştirmeli öğrenme yapısı 17
Şekil 2.6. Değer fonksiyonu diyagramı 21
Şekil 2.7. V* ve Q* yedek diyagramları 22
Şekil 2.8. Genelleştirilmiş politika yinelemesi 24
Şekil 2.9. PÖ algoritmalarının kategorileri 29
Şekil 2.10. Q-Öğrenme akış şeması 30
Şekil 2.11. Q değeri tablosu 30
Şekil 2.12. Q-Öğrenme Algoritması 31
Şekil 2.13. Sarsa algoritması 32
Şekil 2.14. Uçurumda yürüme görevinin grid dünyası 32
Şekil 2.15. Uçurumda yürüme görevinin sonuçları 33
Şekil 2.16. Derin pekiştirmeli öğrenme modeli 36
Şekil 2.17. Değer tabanlı ve Politika tabanlı Derin Pekiştirmeli Öğrenme 36
Şekil 2.18. Derin Q-Öğrenme eğitim aşaması 37
Şekil 2.19. Derin Q-Öğrenme Akış Diyagramı 38
Şekil 2.20. Aktör-Kritik algoritmasının basit gösterimi 39
Şekil 2.21. DDPG algoritması 41
Şekil 2.22. A2C algoritması 43
Şekil 2.23. TRPO algoritması 44
Şekil 2.24. PPO algoritması 45
Şekil 2.25. TD3 algoritması 47
Şekil 2.26. SAC algoritması 48
Şekil 2.27. Simülatörlerin bilinirlik ve kullanılma durumları 49
Şekil 2.28. Örnek webots sahnesi 50
Şekil 2.29. Örnek Gazebo sahnesi 51
Şekil 2.30. Örnek Copeliasim sahnesi 52
Şekil 3.1. ROS yayıncı-abone iletişimi 60
Şekil 3.2. Turtlebot Burger 61
Şekil 3.3. Gazebo-Ros-OpenAI-Stable Baselines Mimarisi 63
Şekil 4.1. 5x5 grid dünya 66
Şekil 4.2. Q ajanın ödül performansı 67
Şekil 4.3. Sarsa ajanın ödül performansı 67
Şekil 4.4. 8x7 grid dünya 68
Şekil 4.5. Q ajanın 8x7 grid dünyada ödül performansı 69
Şekil 4.6. DQN ajanın 8x7 grid dünyada ödül performansı 69
Şekil 4.7. MiniGrid-FourRooms-v0 70
Şekil 4.8. DPÖ Ajanlarının MiniGrid-FourRooms-v0 Ortamındaki Ödül Performansı 71
Şekil 4.9. MiniGrid- Dynamic-Obstacles-16x16-v0 72

Şekil 4.10. DPÖ Ajanlarının MiniGrid-Dynamic-Obstacles-16x16-v0 Ortamındaki Ödül Performansı 73
Şekil 4.11. Gazebo Empty World 75
Şekil 4.12. TD3, PPO ve SAC ajanı için Gazebo Empty World ortamında ödül performansı 75
Şekil 4.13. Gazebo Maze World 77
Şekil 4.14. TD3 ve PPO ajanı için Gazebo Maze World ortamında ödül performansı 78
Şekil 4.15. TD3 ajanı için Gazebo Maze World ortamında ödül performansı 78
Şekil 4.15. PPO ajanı için Gazebo Maze World ortamında ödül performansı 79

TABLOLAR DİZİNİ

Tablo 2.1. Makine Öğrenmesi Yöntemlerinin Karşılaştırılması 13
Tablo 2.2. Pekiştirmeli öğrenmenin parametreleri 17
Tablo 3.1. Turtlebot donanım özellikleri 61
Tablo 3.2. Lazer Mesafe Sensörü LDS-01 61
Tablo 4.1. Q ve Sarsa ajanları için kullanılan hiperparametreler ve eğitim seçenekleri 66
Tablo 4.2. Q ve DQN ajanları için kullanılan hiperparametreler ve eğitim seçenekleri 69
Tablo 4.3. A2C, DQN, PPO, TRPO ajanı için kullanılan hiperparametreler ve eğitim seçenekleri 71
Tablo 4.4. TD3, PPO ve SAC ajanı için Gazebo Empty World ortamında kullanılan hiperparametreler 74
Tablo 4.5. Ajanların eğitim ortamlarına göre eğitilme süreleri 76

SİMGELER VE KISALTMALAR DİZİNİ

a	:Eylem
A	:Eylem uzayı
r	:Ödül
R	:Ödül uzayı
s_t	:Mevcut durum
s’	:Bir sonraki durum
V	:Değer fonksiyonu
ε	:Epsilon
γ	:İndirim faktörü
π	:Politika
α	:Öğrenme oranı

Kısaltmalar

A2C	:Advantage Actor-Critic(Avantaj Aktör-Kritik)
D3QN	:Duello Double Deep Q Network (Düello Çift Derin Q-Öğrenme)
DDPG	:Deep Determintic Policy Gradient(Derin Deterministik Politika Gradyanı)
DÖ	:Derin Öğrenme
DP	:Dinamik Programlama
DPG	:Determintic Policy Gradient(Deterministik Politika Gradyanı)
DPÖ	:Derin Pekiştirmeli Öğrenme
DQN	:Deep Q Network(Derin Q Ağı)
MC	:Monte Carlo
MDP	:Markov Desicion Process(Markov Karar Süreçleri)
MÖ	:Makine Öğrenmesi
PÖ	:Pekiştirmeli Öğrenme
PPO	:Proximal Policy Gradient(Proksimal Politika Optimizasyonu)
ROS	:Robot Operating System(Robot İşletim Sistemleri)
RRT	:Random Rapid Tree (Rastgale Hızlı Ağaçlar)
SAC	:Soft Actor-Critic(Soft Aktör-Kritik)
TD	:Time Difference(Zamansal Fark)
TD3	:Twin Delayed DDPG(İkiz Gecikmeli Derin Deterministik Politika Gradyanı)
TRPO	:Trust Region Policy Optimization(Güven Bölgesi Politikası Optimizasyonu)
YSA	:Yapay Sinir Ağları
YZ	:Yapay Zeka

KAPALI ORTAMLAR İÇİN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARI İLE MOBİL ROBOTLARIN NAVİGASYONU

ÖZET

Mevcut mobil robotik araştırmalarındaki en önemli konulardan biri otonom navigasyondur. Navigasyon; yol planlama ve hareket planlama olarak iki kısımdan oluşur. Bununla birlikte yol ve hareket planlama, haritası çıkarılmamış ortamlarda zorlu bir görevdir. Bu zorlukları aşmak için son yıllarda derin pekiştirmeli öğrenme (DPÖ) yöntemleri sıklıkla kullanılmaktadır. Bu çalışmanın amacı, haritası çıkarılmamış ortamlarda düşük maliyetli sensörler kullanarak bir mobil robotun navigasyonu için derin pekiştirmeli öğrenme yöntemlerinin kullanımını araştırmak, modellemek ve kıyaslamaktır. Belirtilen amaca ulaşmak için iki aşamalı bir yöntem belirlenmiştir. Birinci aşamada kapalı bir oda ortamı iki boyutlu grid olarak temsil edilmiştir. Bu ortam üzerinde A2C, DQN, TRPO, PPO gibi ayrık eylem uzayında çalışabilen farklı pekiştirmeli öğrenme algoritmalarının performansları kıyaslanmıştır. Bu karşılaştırmayı yaparken belirli bir öğrenme kriteri eklenmiştir ve ayrıca epsilon değeri, adım sayısı gibi parametreler değiştirilerek eğitim ve test aşamalarındaki değişiklikler analiz edilmiştir. Değerlendirme ölçütü olarak bölüm başına alınan ortalama ödül kullanılmıştır. Daha yüksek ödül, bir robotun çarpışmadan veya zaman adımı sınırını aşmadan daha fazla sayıda hedefe ulaşabildiği anlamına gelir. Bu ortamlarda PPO ajanının daha başarılı olduğu görülmüştür. İkinci aşamada Gazebo benzetim ortamında üç boyutlu hazır ortamlarda algoritmaların performansı değerlendirilmiştir. Sürekli eylem uzaylarında çalışan TD3, SAC, PPO algoritmaları Gazebo ortamında kıyaslandı. 2B ortamında başarı sağlayan hiperparametreler 3B ortamda da kullanıldı. Bu şekilde TD3 ajanı daha başarılı sonuçlar almıştır. Son olarak ise hem ayrık hem de sürekli eylem uzayında çalışabilen ve 2B ortamda en başarılı olan PPO ajanı ile sadece sürekli eylem uzayında çalışan ve 3B ortamda başarısı görülen TD3 ajanı kıyaslandı ve gözlemler sonucunda TD3 ajanının daha başarılı olduğu görüldü.

Anahtar Kelimeler: Derin Pekiştirmeli Öğrenme, Hareket Planlama, Mobil Robotlar, Otonom Navigasyon, Yol Planlama.

NAVIGATION OF MOBILE ROBOTS WITH DEEP REINFORCEMENT LEARNING ALGORITHMS FOR INDOOR ENVIRONMENTS

ABSTRACT

One of the most important topics in current mobile robotics research is autonomous navigation. Navigation consists of two parts: path planning and motion planning. However, path and motion planning is a challenging task in unmapped environments. To overcome these challenges, deep reinforcement learning (DRL) methods have been widely used in recent years. The aim of this work is to investigate, model and benchmark the use of deep reinforcement learning methods for navigation of a mobile robot using low-cost sensors in unmapped environments. In order to achieve the stated goal, a two-stage methodology was defined. In the first stage, an indoor room environment is represented as a two-dimensional grid. The performances of different reinforcement learning algorithms such as A2C, DQN, TRPO, PPO which can operate in discrete action space are compared on this environment. While making this comparison, a specific learning criterion was added and also parameters such as epsilon value, number of steps were changed and the changes in the training and testing phases were analyzed. The average reward per episode was used as the evaluation criterion. Higher reward means that a robot is able to reach a greater number of targets without colliding or exceeding the time step limit. The PPO agent was found to be more effective in these environments. In the second phase, the performance of the algorithms was evaluated in three-dimensional ready-made environments in the Gazebo simulation environment. TD3, SAC, PPO algorithms operating in continuous action spaces were compared in Gazebo environment. Hyperparameters that were successful in 2D environment were also used in 3D environment. Finally, PPO agent, which can work in both discrete and continuous action space and is the most successful in 2D environment, and TD3 agent, which works only in continuous action space and is successful in 3D environment, were compared and TD3 agent was more successful.

Keywords: Deep Reinforcement Learning, Motion Planning, Mobile Robots, Autonomous Navigation, Path Planning.

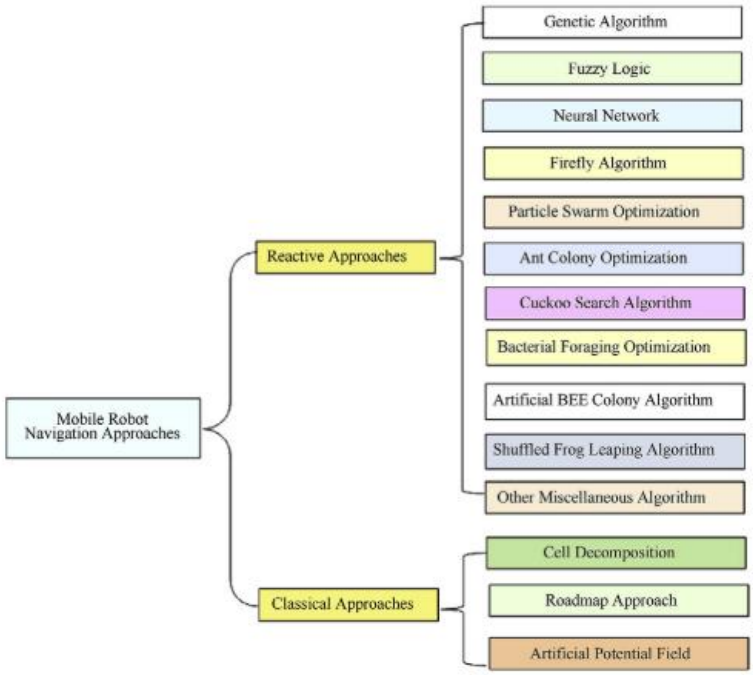
1. GİRİŞ

Robot, elektronik bileşenler, mekanik aksam, sensör ve yazılımdan oluşan, otonom ya da insan destekli makinelerdir. Robotik ise robotlarla ilgilenen bilim dalıdır. Mobil robotlar, fiziksel olarak sabit olmayan tanımlanmış bir çevrede (karada, su altında ya da havada) hareket ederek istenilen görevleri yerine getirebilen robotlardır (Bölük, 2019). Bu sebeple gezgin robot da denilmektedir. Mobil robotlar, birçok alanda giderek daha fazla kullanılmaktadır. Mobil robotların örnek uygulamaları, yaşlılar için hizmet robotları, bir fabrikada mal nakletmek için otomatik yönlendirmeli araçlar, insansız bomba imha robotları ve gezegen keşif robotları gibi geniş bir yelpazeyi içerir.

Navigasyon, mobil robotik alanında temel bir görevdir ve küresel navigasyon ve yerel navigasyon olarak iki tür olarak sınıflandırılabilir. Küresel navigasyonda, çevre hakkında önceden bilgi mevcut olmalıdır. Küresel navigasyon için Voronoi Grafiği, Yapay Potansiyel Alan Yöntemi, Dijkstra algoritması, Görünürlük Grafiği, Gridlar, ve Hücre Ayrıştırma yöntemi vb. gibi birçok yöntem geliştirilmiştir. Yerel navigasyonda robot, ultrasonik sensörleri, keskin kızılötesi mesafe sensörleri ve görüş (kamera) sensörleri vb. gibi donanımlı sensörleri kullanarak hareketine ve yönüne otonom olarak karar verebilir. Yerel navigasyon problemini çözmek için Bulanık Mantık, Genetik Algoritma, Parçacık Sürüsü Optimizasyon algoritması, Karınca Kolonisi Optimizasyon Algoritması, vb. algoritmalar çeşitli araştırmacılar tarafından başarıyla kullanılmaktadır(Pandey, 2017).

Mobil bir robotun navigasyonu için kullanılan çeşitli yöntemler genelde klasik ve reaktif (reactive) yaklaşımlar olarak iki kategoriye ayrılmaktadır. Küresel navigasyonda mobil robot, ortamın ön bilgilerine, engel pozisyonu ve hedef pozisyonu bilgisine ihtiyaç duyarken, yerel navigasyonda ortam hakkında önceden bilgi gerekmez. Küresel navigasyon stratejisi, tamamen bilinen bir ortamla ilgilenir. Yerel navigasyon stratejisi bilinmeyen ve kısmen bilinen ortamla ilgilidir. Bilinen bir ortam için yol planlama algoritması, klasik bir yaklaşıma dayanmaktadır. Bu algoritmalar gelenekseldir ve zekası sınırlıdır. Yerel navigasyon yaklaşımları, daha akıllı oldukları ve bir planı bağımsız olarak kontrol edebildiği ve uygulayabildiği için reaktif yaklaşımlar olarak bilinir. Şekil 1.1'de bu yaklaşımlara ait algoritmalar gösterilmiştir. Başlangıçta, klasik yaklaşımlar robot navigasyon problemlerini çözmek için çok popülerdi, çünkü o günlerde yapay zeka (YZ) temelli teknikler geliştirilmiyordu. Bir görevi yerine getirmek için klasik

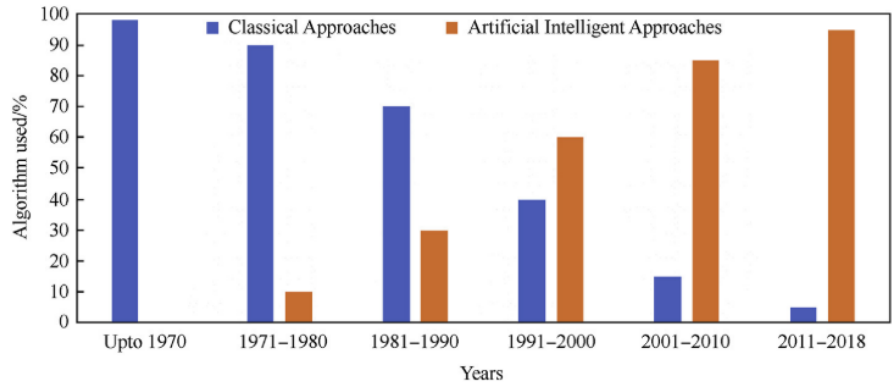
yaklaşımlar kullanılarak ya bir sonuç elde edileceği ya da bir sonucun mevcut olmadığı teyit edilir. Bu yaklaşımın en büyük dezavantajı yüksek hesaplama maliyeti ve çevrede mevcut olan belirsizliğe yanıt vermemesidir. Bu nedenle gerçek zamanlı uygulamalar için daha az tercih edilir. Son zamanlarda, Genetik Algoritma, Bulanık Mantık gibi reaktif yaklaşımlar Sinir Ağı, Ateşböceği Algoritması, Parçacık Sürüsü Optimizasyonu, Karınca Kolonisi Optimizasyonu, Bakteriyel Yemleme Optimizasyonu, Yapay Arı Kolonisi, Yarasa Algoritması ve daha fazlası mobil robot navigasyonu için en popüler araç olarak kabul edilmiştir. Çevrede mevcut olan belirsizlikle başa çıkma konusunda büyük yetenekleri vardır. Günümüzde reaktif yaklaşımlar, daha az hesaplama çabasıyla belirsiz bir ortamı hızlı bir şekilde ele alma yeteneğine sahip oldukları için daha popülerdir. Şekil 1.2'de mobil robot navigasyon yaklaşımlarının yıllara göre kullanılma durumları gösterilmiştir(Patle ve diğ., 2019).



Şekil 1.1. Mobil Robot Navigasyon Yaklaşımlarının Sınıflandırılması (Patle ve diğ., 2019)

Yazılım alanındaki gelişmeler ve donanımın maliyetlerinin kademeli olarak düşürülmesi nedeniyle robotik sistemler, kapsamlarını giderek genişletiyor ve çok sayıda sektörün üretkenliğini, verimliliğini ve çalışma ortamı güvenliğini artırıyor. Robotik alanındaki ilerlemeye rağmen, mobil robotlar gündelik hayatta yeteri kadar yer almıyor. İnsanların yoğun olduğu alanlarda çalışacak robotların, farklı ve beklenmedik insan davranışlarını

modelleme ve buna göre çalışma yeteneğini sergilemeleri gerekir (Cheng ve diğ., 2018). Bu, robotik alanında çalışan araştırmacılar tarafından üzerine yoğun bir şekilde çalışılan zorlu bir özelliktir.



Şekil 1.2. Mobil Robot Navigasyon Yaklaşımlarının Yıllara Göre Gelişimi(Patle ve diğ., 2019).

Mobil robotlarının geliştirilmesindeki ana zorluklardan biri, kusursuz ve robotun yürüteceği göreve uygun bir navigasyon yöntemi tasarlamaktır. Navigasyon yöntemleri, haritalama, yerelleştirme (robotun o anki konumunu bulma) ve yol planlama olarak üçe ayrılır (Ruan ve diğ., 2019).Yol planlama, engellerin olduğu bir ortamda, otonom mobil robotun bir başlangıç noktasından başka bir hedef noktasına gidebilmesi için engellere çarpmayacağı uygun bir yolu bulma işlemidir(Hu ve Yang, 2004). Daha az dönüş eyleminin yapıldığı, daha az fren yapılan, hedefe en kısa yoldan ulaşılan yol en uygun yol olarak ifade edilebilir.

Geçtiğimiz yıllarda yol planlama problemini çözmek için çeşitli stratejiler formüle edilmiştir ve Makine Öğrenimi tabanlı yaklaşımlar en umut verici sonuçları sergileyen metodolojilerden bazılarıdır (Aradi, 2020).

Makine Öğrenimi (ML) (Mitchell, 1997), deneyim yoluyla kendi kendini geliştiren algoritmalar üreten ve Denetimli Öğrenme (Nasteski, 2017), Denetimsiz Öğrenme (Celebi ve Aydin, 2016) ve Pekiştirmeli Öğrenme (PÖ) (Sutton ve Barto, 2018) şeklinde düzenlenen yapay zekanın bir alt kümesidir. Denetimli algoritmalarda öğrenme, veri kümeleri adı verilen önceden düzenlenmiş veri kümeleri aracılığıyla örüntü bulma modelleri oluşturmaktır.

Denetimsiz öğrenmede etiketsiz verilerden, bu verilere ait gizli özellikler ortaya çıkarılır. PÖ' de, ortamla etkileşim halinde olan akıllı bir ajan, önceden belirlenmiş bir hedefe ulaşmak için hangi eylemlerin benimsenmesi gerektiğini deneme yanılma yoluyla öğrenir.

Pekiştirmeli öğrenmede amaç, zaman içinde ve eylemlerinin kalitesini değerlendiren bir ödül sistemi tarafından yönlendirilerek, beklenen ödül toplamını en üst düzeye çıkaran komutları seçmek için yeterli deneyimin toplamasıdır.

Bina içi navigasyonla ilgili olarak, geleneksel yaklaşımlar, bir engel haritasına dayanarak bir eylem planı oluştururlar (Ruan ve diğ., 2019). Mobil robotların, dinamik alanlarda daha önce karşılaşmadığı senaryolara karşı esnek bir hareket planı kurgusu olmalıdır. Navigasyonun hareket planlama görevi için kullanılan PÖ algoritmaları ile ajan, çevresel uyaranları değerlendirmek için gelişmiş bir yetenek kazanır ve sonuç olarak bilinmeyen ortam navigasyonunda en ideal ideal eylemleri belirler(Chen ve diğ., 2017).

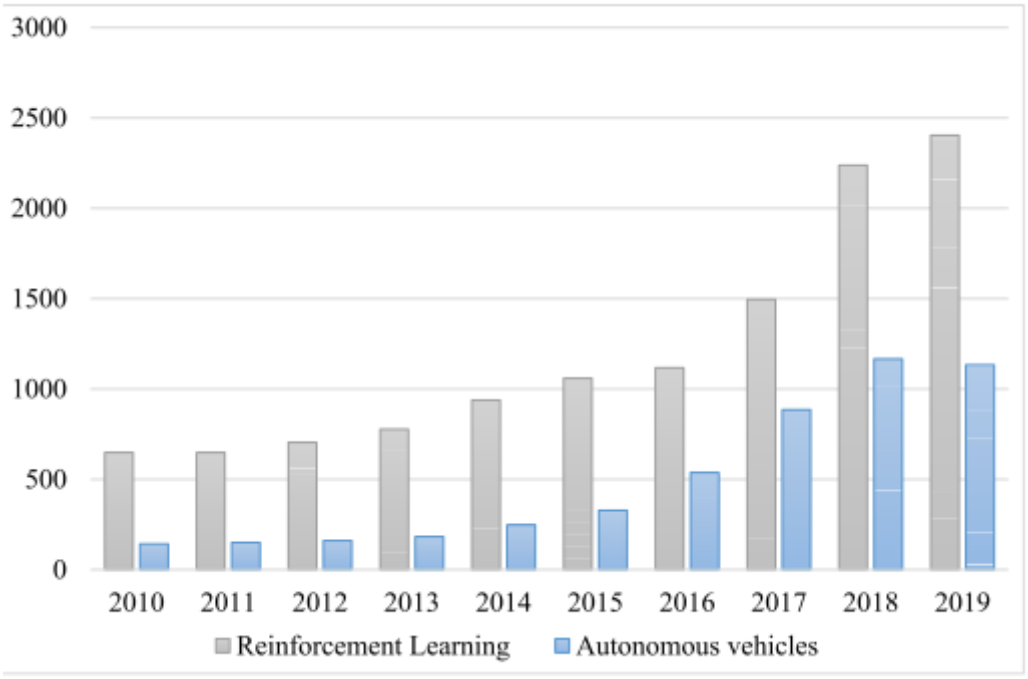
Pekiştirmeli Öğrenme algoritmaları, örnek, bellek ve hesaplama karmaşıklığı ile ilgili bazı sınırlamalarla karşı karşıyadır (François-Lavet ve diğ., 2018). Ancak bu sorunlar, Derin Öğrenme (DÖ) (Goodfellow ve diğ., 2016) kullanılarak aşılabilir. Derin Öğrenme, üst düzey soyutlamalarda kompakt özellikleri tanımlayabilen hesaplama sistemleri oluşturmayı amaçlar. DÖ'de, Yapay Sinir Ağları (YSA) kullanılır. YSA'lar biyolojik sinir ağlarından esinlenen katmanlı yapılardır. PÖ ve DÖ'nün birleştirilmesinden meydana gelen uygulamalar, Derin Pekiştirmeli Öğrenme (DPÖ) (François-Lavet ve diğ., 2018) olarak tanımlanır.

Mnih ve arkadaşlarının yaptığı yayından(Mnih ve diğ., 2013) bu yana önemli miktarda araştırma bu alana odaklandı ve yeni kullanım durumları ve uzantılar geliştirildi.(Kempka ve diğ., 2016; Hasselt ve diğ., 2015; Mnih ve diğ., 2016; Silver ve diğ., 2016).

DeepMind (URL-1), 2013 yılında, atari video oyunlarında uzman oyuncuları yenebilecek yöntemler geliştirmek için Derin Q-Öğrenme'yi kullanarak yapay zeka alanında devrim yarattı(Mnih ve diğ., 2013). Bundan hareketle, AlphaGo (Silver ve diğ., 2016), AlphaZero (Silver ve diğ., 2017) ve AlphaStar (Vinyals ve diğ., 2019) uygulamaları sırasıyla hayata geçirilmiştir. 2015 yılında AlphaGo, kendi kendini oyundan eğitmek için

evrişimsel sinir ağlarını (CNN) ve PÖ'yü birleştirerek Go oyununda olağanüstü performanslar elde etti. 2017 yılında, Go'nun yanı sıra Satranç ve Shogi oyunlarında ustalaşan AlphaZero adlı benzer bir AlphaGo yaklaşımı piyasaya sürüldü. 2019'da AlphaStar, DPÖ 'yü kullanarak karmaşık StarCraft II oyununa hakim oldu ve gerçek zamanlı maçlarda dünyanın en iyi takımlarından bazılarını yendi.

DPÖ, video oyun alanında ciddi başarılar elde etmiştir. Bunu sağlarken ham görüntüleri işlemiştir. Bu da mobil robot navigasyonuna ilham olmuştur. Otonom araçlar ve DPÖ ile ilgili araştırma makalelerinin sayısı son birkaç yılda artmıştır. Şekil 1.3'te bununla ilgili bir grafik verildi.



Şekil 1.3. Otonom Araçlar İçin Derin Pekiştirmeli Öğrenme Araştırmaları (Aradi, 2020).

Son yıllarda, mobil robotların yol planlaması için DPÖ uygulamaları artarak kullanılmaktadır ve bu konuda büyük başarılar elde edilmiştir. DPÖ haritasız ortamlarda güçlü öğrenme yeteneği ve düşük sensör doğruluğu bağımlılığı gibi avantajlara sahip olsa da, yol planlaması için uzun eğitim süresi, özellikle sınırlı hesaplama kaynakları durumu için mobil robotlara uygulama engeli oluşturmuştur. Çoğu durumda, robotların bir fizik motoruna sahip 3B simülasyon ortamında eğitilmesi gerekir. Bu senaryoda, robotun hareketi fiziksel kurallarla sınırlıdır. DPÖ, eğitim süresini önemli ölçüde artıran

etkileşimli deneme-yanılma yöntemine dayalı olarak uygulanır. Ortamlar veya görevler karmaşık olduğunda, algoritmanın rastgele başlatılan ağ parametreleri ile istenen hedeflere yakınsaması zordur.

Bu tezin amacı, haritası çıkarılmamış ortamlarda düşük maliyetli sensörler kullanarak bir mobil robotun navigasyonu için derin pekiştirmeli öğrenme yöntemlerinin kullanımını araştırmak, modellemek ve doğrulamaktır. Belirtilen amaca ulaşmak için iki aşamalı bir yöntem belirlenmiştir. Birinci aşamada kapalı bir ortam iki boyutlu grid olarak temsil edilmiştir. Bu ortam üzerinde Q-Öğrenme, SARSA, A2C, DQN, PPO,TRPO gibi farklı pekiştirmeli öğrenme algoritmalarının karşılaştırmalı bir simülasyon çalışması sunulmaktadır. Bu karşılaştırmayı yaparken belirli bir öğrenme kriteri eklenmiştir ve ayrıca epsilon değeri, adım sayısı gibi parametreler değiştirilerek eğitim ve test aşamalarındaki değişiklikler analiz edilmiştir. Bu çalışma, simülatör programı tarafından sağlanan aktörler (ajan, sensör, engeller vb.) ile desteklenmiştir. İkinci aşamada ise Gazebo benzetim ortamında üç boyutlu hazır ortamlarda PPO, SAC, TD3 algoritmaların performansı değerlendirilmiştir.

Bu tezde ilk olarak, ilgili PÖ algoritmalarını hafif bir 2 boyutlu ortamda değerlendirildi. Daha sonra, 3B ortam için zaman alıcı çalışmaları engellemek için 2B ortamda gözlem durumları, ödül fonksiyonu, ağ yapısı ve parametre optimizasyonu dahil olmak üzere DPÖ'ye dayalı algoritma tasarlandı. Tasarlanan algoritmayı, derin sinir ağının ağırlıkları ve önyargıları vb. dahil olmak üzere yakınsanmış ağ parametrelerini elde etmek için yeniden eğitim için basit bir 3B ortama aktarıldı. Bu parametreleri başlangıç değerleri olarak kullanarak, modeli karmaşık bir 3B ortamda eğitmeye devam edildi.

Tezin literatüre katkısı aşağıdaki gibi listelenebilir;

- Haritasız ve dinamik ortamlarda farklı DPÖ algoritması performansların karşılaştırıldı.
- Ortam ve ağ parametrelerini içeren iki aşamalı bir yol tercih edildi. Böylece DPÖ tabanlı yol planlamasının geliştirme verimliliğini ve yakınsamasını iyileştirilmesine katkı sunudur. Ayrıca algoritmaların, ağ yapısının test edilmesine ve değerlendirilmesine katkıda bulunuldu.

- Grid dünya büyüdükçe Q öğrenme algoritmasının yetersiz kaldığı tespit edildi. Ayrık eylem uzayında kullanılabilecek DPÖ algoritmalarından PPO'nun daha başarılı olduğu görüldü.
- Çalışmada kullanılan Gazebo ortamları sürekli eylem uzayına sahip ve dinamik engeller vardı. Bu ortamlarda TD3 algoritmasının daha başarılı olduğu tespit edildi.
- Yapılan deneylerle geleneksel yol planlama algoritmalarının aksine pekiştirmeli öğrenme modellerinin haritasız bir şekilde yol planlama yapabildiği tespit edildi.

Tezin kalan bölümlerin başlıkları şöyledir; İkinci bölümde genel bilgiler, üçüncü bölümde malzeme ve yöntem, dördüncü bölümde bulgular ve tartışma ve beşinci bölümde sonuçlar ve öneriler başlıkları işlenmiştir. Genel bilgiler bölümünde pekiştirmeli öğrenme, hareket planlama, benzetim ortamları konuları hakkında bilgiler verilmiştir. Literatürde pekiştirmeli öğrenme ile mobil robot navigayonu hakkında yapılan çalışmalar incelenmiştir. Malzeme ve yöntem bölümünde tezde kullanılan yazılım ve donanım hakkında bilgiler verilmiştir. Ayrıca kıyaslanan pekiştirmeli öğrenme algoritmaları hakkında, ödül-ceza, çevre tasarımı, fonksiyon yaklaştırıcılar vb. gibi konularda bilgi verilmiştir. Bulgular ve tartışma bölümünde pekiştirmeli öğrenme ajanlarının, ortamın iki boyutlu ve üç boyutlu olduğu durumlarda ve statik ya da dinamik olduğu durumlarda performans sonuçları ortaya koyulmuştur. Sonuçlar ve öneriler bölümünde ise algoritmalardan elde edilen veriler değerlendirilmiş ve gelecekte yapılması planlanan çalışmalardan bahsedilmiştir.

2. GENEL BİLGİLER

2.1. Hareket Planlama

Hareket planlaması, bir robotik sistemin belirli bir başlangıç durumunu söz konusu sistem için belirli bir hedef bölgeye bağlayan sürekli bir yol bulma problemidir. Çarpışmadan kaçınma, sınırlı kuvvetler, sınırlı ivme gibi kısıtlamalara çözüm geliştirilir. (Şucan ve diğ., 2012). Hareket planlaması, önemli miktarda zaman kazandırabileceğinden ve bir mobil robotun yıpranmasını ve maliyetini azaltabileceğinden (Zhang ve diğ., 2018), kendisini ilkel bir navigasyon olarak sunar.

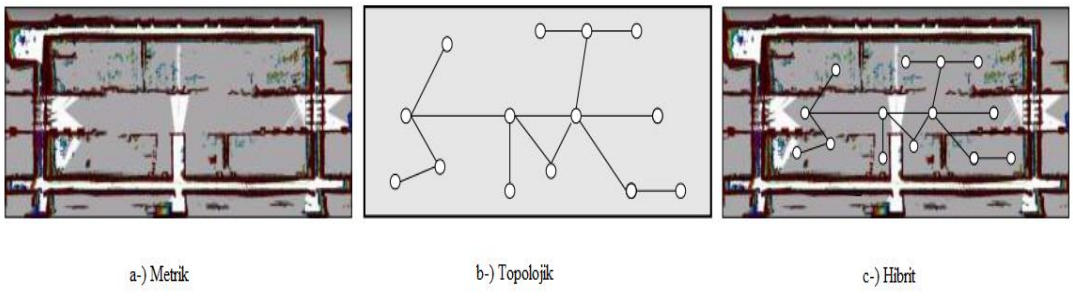
2.1.1. Hareket Planlayıcıları

Hareket planlamasında, küresel ve yerel olarak iki tür planlayıcı vardır. Burada ayrım çevreden bilgilere erişim düzeyidir (Zhang ve diğ., 2018). Küresel planlayıcılar, tamamen bilinen ortam altında en uygun ya da en uyguna yakın yollar oluştururlar. Genel haritanın sürekli güncellenmesi gerekir. Diğer taraftan yerel planlayıcıların çevre bilgilerine erişimi kısıtlıdır. Sensörlerden gelen verilere göre anlık olarak eylem seçilir. Kısa vadeli yollar oluşturulur.

2.1.2. Çevre Gösterimleri

Hareket planlayıcıların çevre hakkında ön bilgiye ya da robotun hareketi boyunca veri toplamasına ihtiyacı vardır. Toplanan veriler ile ise bir özellik haritası temsil edilir. (Zhang ve diğ., 2018). Harita gösterimi iki şeklide yapılabilir:

1) Metrik Gösterim: Şekil 2.1a'da çevrenin metrik gösterimi verildi. Bu gösterimde çevre grid(ızgara) tabanlı bir düzenleme ile bölünür.
2) Topolojik Gösterim: Şekil 2.1b'de çevrenin topolojik temsili görülmektedir.



Şekil 2.1.Çevrenin metrik, topolojik ve hibrit gösterimleri

Topolojik gösterimde, düğüm ve yay modeli ile temsil edilir. Düğümler, birbirinden farklı engelsiz konumları ve sınırları temsil eder. Yaylar, düğümler arasındaki bağlantılardır ve konumlar arasındaki yolları temsil eder.

Grid tabanlı yöntemler, doğru ölçümler üretirken, genellikle karmaşıklıkları büyük ölçekli iç mekan ortamlarında verimli planlamayı engeller. Topolojik haritalar ise robot-ajan tarafından öğrenilmesi zor gösterim türüdür (Thrun, 1998; Tomatis ve diğ., 2001). Bununla birlikte, her iki yöntemi birleştirmek, her birinin olumsuz yanlarını aşmak ve daha sağlam bir ortam modeli oluşturmak mümkündür. Şekil 2.1c'de küresel topolojik haritanın üstünde yerel grid tabanlı gösterimlerin kullanıldığı (Thrun, 1998; Tomatis ve diğ., 2001) karma(hibrit) bir örnek sunulmaktadır.

2.1.3. Küresel Yol Planlayıcıları

İç mekan navigasyonundaki baskın küresel yol arama yöntemleri, sezgisel-arama algoritmaları olarak sınıflandırılır (Zhang ve diğ., 2018). Bu kategori altında, Dijkstra(Marin-Plaza ve diğ., 2018; Risald ve diğ., 2017), Floydwarshall (Marin-Plaza ve diğ., 2018), A * (Zhang ve diğ., 2018; Liu ve Gong, 2011; Stentz, 1995) ve hızlı bir şekilde keşfedilen rastgele ağaçlar (RRT) gibi algoritmalardır(Garrote ve diğ., 2014).

Dijkstra'da, yollar topolojik çevre temsillerinde komşu düğümlerinin seçimi yoluyla yaratılır. Formüle edilmiş her bir yol, rotanın düğüm bağlantılarını belirleyen ağırlıklı yaylardan kaynaklanan ilişkili bir maliyete sahiptir. Minimum maliyetle olan yol seçilir ve böylece en kısa yol hesaplanır. Floyd-Warshall (Risald ve diğ., 2017), tüm düğüm çiftleri arasındaki en kısa yolu belirler. Bu algoritmanın girdisi, ağırlıklı ve yönlendirilmiş bir grafiktir. Algoritma ayrıca negatif ağırlıklı tarafı da hesaplar. A *, başlangıç düğümü ve son düğüm arasındaki en kısa yolu bulmak için sezgisel bir değerlendirme fonksiyonu kullanır(Stentz, 1995). Denklem (2.1)'de sezgisel bir değerlendirme fonksiyonu gösterilmiştir;

$$f(i) = g(i) + h(i) \quad (2.1)$$

- i: robotun mevcut konumu;
- g(i) : Başlangıç düğümünden i için geçmiş maliyet fonksiyonu;
- h(i) : i'den hedef düğüme Öklid uzaklığı.

RRT(Garrote ve diğ., 2014), örnekleme tabanlı bir yöntemdir. Her tekrarda, rastgele bir düğüm seçilir ve en yakın düğüme bağlanır. Düğüm seçenekleri tamamen rasgele olduğundan, birkaç ağaç dalı (yol gösterimleri) oluşturulur. Kavramsal olarak, en az bir rota kademeli olarak hedef alana doğru birleşir.

2.1.4. Yerel Yol Planlayıcıları

Küresel yol planlamasının yanı sıra, dinamik metrik özellik haritaları üzerinde yerel yol arama algoritmaları kullanılarak sağlam hareket planlama metodolojileri uygulanabilir.

Yapay potansiyeli alan (APF) (Zhu ve diğ., 2006) stratejisi tarafından indüklenen bir platform, bir yapay kuvvet alanına tabidir. Bu alanda robot kendisini iten ve çeken kuvvetlere maruz kalır. Sonuçta robotu hareket ettiren bir kuvvet oluşur. Dinamik Pencere Yaklaşımı (DWA) (Fox ve diğ., 1997), bir mobil robotun mevcut hızlarından oluşan pencere adı verilen bir yapı sunar. Bu pencereden en uygun çözümü seçer. Robotu bu şekilde hedefe ulaştırır.

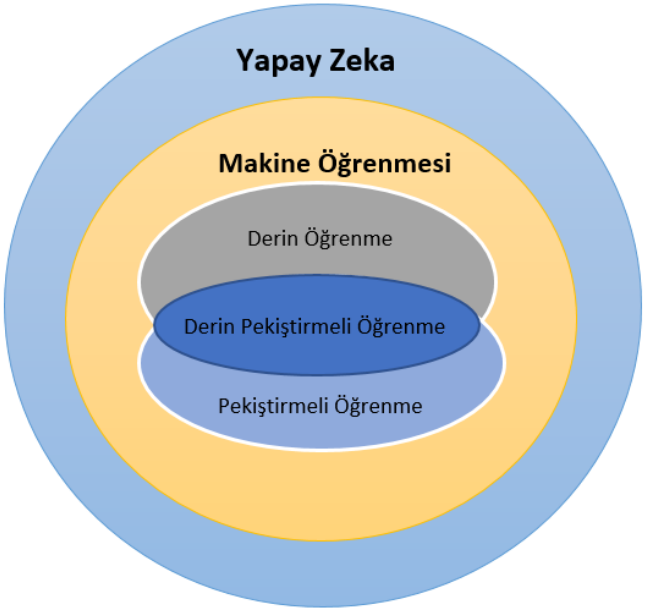
Zaman Elastik Bantları (TEB) (Marin-Plaza ve diğ., 2018; Keller ve diğ., 2014) yaklaşımı da benzer bir çalışma prensibine sahiptir: aracın geometrik, kinetik ve dinamik kısıtlamalarının bilgisi ile bir robotun yerleşik ara yol noktalarında gezinmesine izin veren bir dizi hız komutu oluşturur.

2.2. Makine Öğrenmesi

Bilgisayarlar ilk icat edildiğinden beri, bilim adamları makineleri daha akıllı hale getirmeye çalıştılar. Ancak, zekanın tanımı bugün hala devam eden bir tartışma konusudur. Alan Turing, Turing Testini ilk olarak 1950'de Manchester Üniversitesi'nde "Bilgisayar Makineleri ve Zeka" (Harnad, 2006) başlıklı makalesinde tanıttı. Turing testi, bir makinenin insan davranışını taklit etme yeteneğini ölçer. Spesifik olarak, bir sorgulayıcının başka bir odadaki bir adama ve bir bilgisayara bir dizi soru sorduğu ve diğer iki oyuncudan hangisinin insan, hangisinin bilgisayar olduğunu belirlemek için bir "taklit oyunu" tanımlar. Bilgisayar sorgulayıcıyı andırabilirse test geçilir. Yapay Zeka, 1956 yazında ünlü Dartmouth konferansında John McCarthy tarafından ortaya konuldu. Bu konferans, YZ'nin bir bilgisayar bilimi alanı olmasının başlangıç noktası olarak

görülmüştür. YZ'nin ilk günlerinde, YZ algoritmaları esas olarak matematiksel kurallar ve mantık kuralları ile formüle edilebilen problemleri çözmek için tasarlandı.

Makine öğrenmesi(MÖ), 1959'da Arthur Samuel (Bell Labs, IBM, Stanford) tarafından icat edildi. Bir YZ sistem, ham verilerden kendi bilgisini öğrenme yeteneğine sahip olmalıdır. Bu kapasite MÖ olarak bilinir. Birçok yapay zeka sorunu, bu sorun için ham verilerden özellikler çıkarmak için bir model tanıma algoritması tasarlayarak ve ardından bu özellikleri MÖ algoritmasına sağlayarak çözülebilir. Şekil 2.2'de yapay zeka hiyerarşisi gösterildi.



Şekil 2.2.Yapay Zeka Hiyerarşisi

Makine Öğrenimi ((Mitchell 1997), Denetimli Öğrenme (Nasteski 2017), Denetimsiz Öğrenme(Celebi and Aydin 2016) ve Pekiştirmeli Öğrenme (Sutton and Barto 2018) şeklinde düzenlenen yapay zekanın bir alt kümesidir.

2.2.1. Denetimli-Gözetimli Öğrenme

Denetimli öğrenme, eğitim verilerine dayalı bir fonksiyon üreten makine öğrenimi tekniğidir. Başka bir deyişle, bu öğrenme tekniğinde girdiler (etiketli veriler) ile istenen çıktıları eşleştiren bir fonksiyon üretilir. Eğitim verileri hem girdilerden hem de çıktılardan oluşur. Fonksiyon, regresyon veya sınıflandırma algoritmaları ile belirlenebilir. Doğrusal regresyon, girdiler ve çıktılar arasında doğrusal bir ilişki olup

olmadığını belirlemek için sıklıkla kullanılan bir tekniktir. Genellikle tahmin ve tahmin problemlerini ve diğer birçok veri madenciliği problemini çözmek için kullanılabilir. Sınıflandırma teknikleri, kalıpları tanıyarak ve verileri inceleyerek nitel bir yanıtı tahmin etmeye odaklanır. Yaygın olarak kullanılan bazı sınıflandırma teknikleri vardır. Bu teknikler lojistik regresyon, lineer diskriminant analizi, K-en yakın komşular, ağaçlar, sinir ağları, destek vektör makinelerini içerir (Talabis ve diğ., 2015).

2.2.2. Denetimsiz-Gözetimsiz Öğrenme

Denetimsiz öğrenmede girdi verilerinin hangi sınıfa ait olduğu net değildir. Bu MÖ algoritması, etiketlenmemiş veriler üzerinde bilinmeyen bir yapıyı tahmin etmek için bir fonksiyon kullanır. Veri örneklerinin uzaklıklarına, komşuluk ilişkilerine ve yoğunluğuna göre veriler hakkında çıkarımlarda bulunur. Genel olarak tavsiye sistemleri, pazarlama sistemleri, müşteri segmentasyonu ve boyut küçültme gibi alanlarda kullanılmaktadır. En çok kullanılan denetimsiz öğrenme algoritmaları kümeleme, birliktelik kuralları, temel bileşen analizidir.

2.2.3. Pekiştirmeli Öğrenme

Pekiştirmeli öğrenme, en uygun davranış veya eylemin olumlu bir ödülle pekiştirildiği kavramdır. Bir PÖ ajanı/modeli, ortamıyla etkileşime girerek ve eğitim veri kümesinin olmadan bu etkileşimlerin sonuçlarını gözlemleyerek öğrenir. Ajan, bu öğrenmeyi gerçekleştirmek için PÖ algoritmalarını kullanır.

PÖ, gerçek zamanlı karar verme, tavsiye sistemleri, sağlık, oyunlar için yapay zeka, robotik, otonom sürüş, bilgisayarla görme (tanıma, algılama, algılama) ve daha sonra öğrenebilme becerisi gibi beceri kazanım sistemleri gibi becerilere sahip sistemlerde kullanılır. Pekiştirmeli öğrenme ile ilgili detaylı anlatım bölüm 2.4 ve 2.5'te yapılmıştır. Daha önce yapılan çalışmalar ise bölüm 2.7' te incelenmiştir.

Tablo 2.1, makine öğrenmesi kavramları arasındaki avantaj ve dezavantajları göstermektedir. Denetimli/denetimsiz veya denetimli/pekiştirmeli öğrenme kombinasyonları gibi karma öğrenme yaklaşımları genellikle iyi sonuçlara yol açar. Örneğin, denetimli/denetimsiz karma öğrenme yaklaşımı bankacılık sektöründe hesap

hareketlerindeki anormallikleri tespit etmek için yaygın olarak kullanılmaktadır(Engin, 2019).

Tablo 2.1. Makine Öğrenmesi Yöntemlerinin Karşılaştırılması

Denetimli Öğrenme	**Denetimsiz Öğrenme**	**Pekiştirmeli Öğrenme**
Referans cevaplarına yaklaşmayı öğrenme	Temel veri yapısını öğrenme	Deneme yanılma yoluyla öğrenme
Doğru cevaplara ihtiyacı var	Geri bildirim gerekmez	Ajan kendi eylemleri hakkında geri bildirime ihtiyaç duyar
Model giriş verilerini etkilemez	Model giriş verilerini etkilemez	Ajan kendi gözlemlerini etkileyebilir

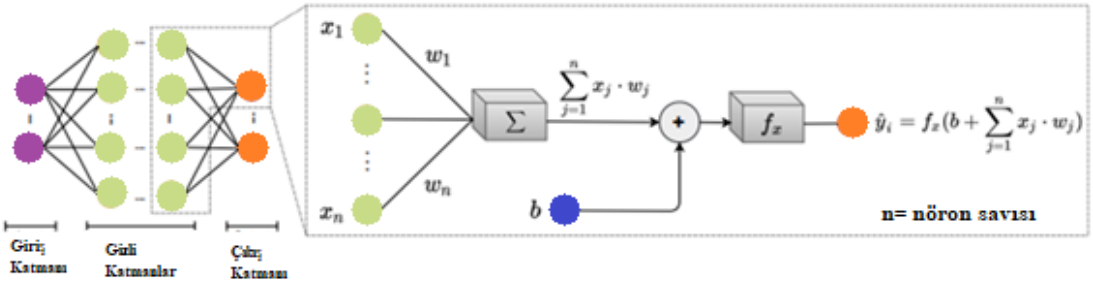
2.3. Derin Öğrenme

Derin Öğrenme (Fadlullah ve diğ., 2017; Schmidhuber, 2015) veri yapılarından temel özellikleri çıkarmak için Yapay Sinir Ağları'nı kullanan makine öğrenmesinin bir alt kümesidir (Alom ve diğ., 2019). YSA'lar insan beyninde bulunan sinir bağlantılarından esinlenmiştir ve her bir temel unsuruna nörön denir. Bir yapay nöron, giriş verilerini alan, işleyen ve bir çıkış sinyali döndüren bir fonksiyon olarak nitelendirilebilir. DÖ'de nöronlar katmanlara bağlanır ve düzenlenir, katmanlar YSA mimarilerindeki yerleşimlerine göre kategorize edilebilir:

- Giriş katmanı: YSA iş akışının başlangıcıdır. Dış DÖ sisteminden veriyi alır ve daha fazla işlem için iletir;
- Çıktı katmanı: YSA iş akışının sonlandıran katmandır. Önceki katmanlar tarafından ele alınan verileri DÖ sistemine geri döndürür;
- Gizli katmanlar: Giriş çıkış katmanları arasında veri özelliklerini tanımlamak ve işlemek için tasarlanmış aradaki katmanlardır. Tek bir gizli katmanla sınırlı YSA'lar Sığ Ağlar olarak adlandırılır. Birden fazla gizli katmana sahip ağ mimarilerine Derin Ağlar denir.

Şekil 2.3'te derin yapay sinir ağlarının mimarisi gösterilmiştir. Buna göre her nöronun çıkış sinyali $\hat{y}_i$,lineer olmayan bir aktivasyon fonksiyonu f_x tarafından gerçekleştirilen,

önceki x_j katmanlarındaki aktif nöronların sinyalleri ile kanallarla ilişkili ağırlıklar arasındaki çarpım ile bir sapma b toplamının hesaplanmasından kaynaklanır.



Şekil 2.3.Derin Yapay Sinir Ağlarının Mimarisi

2.4. Pekiştirmeli Öğrenme

2.4.1. Tarihsel Bağlam

Pekiştirmeli öğrenmenin temelleri eğitim psikolojisine dayanmaktadır. Eğitim psikolojisinin babası olarak bilinen Thorndike'ın hayvan zekası üzerine yaptığı tezin(Thorndike, 1911) yılı olan 1898, davranışın deneysel analizi olarak bilinen alanın başlangıcını işaret eder. Tez, hayvan ve insan öğrenimi hakkında düşünmede büyük bir değişim başlattı, önemli metodolojik yenilikler sağladı ve özellikle B. F. Skinner tarafından daha sonraki araştırma ve teorinin tohumlarını attı(Chance, 1999).

Thorndike temel davranışsal süreçleri araştırmaya başladı ve bu süreçlerin farklı türlerde oldukça benzer göründüğünü fark etti. Civcivler, köpek ve kedilerden daha yavaş öğrendiler, ama farkı yaratan bedensel organlarındaki ve içgüdüsel dürtülerindeki farklılıktır. Zekadaki herhangi bir farklılığa atıfta bulunulamayacağını öne sürdü(Thorndike, 1911) .

Thorndike'ın yapboz kutusu adı verilen meşhur deneyinde yapboz kutusuna aç bir kedi konuldu ve kutunun hemen dışına bir parça balık yerleştirildi. Kedi, kutuyu kapalı tutan mandalı serbest bırakan bir pedala basarak kutudan kaçabilirdi. Kedi ilk başta kutunun çıtaları arasına sıkıştırmaya veya çıtaları ısırmaya çalıştı. Sonunda kedi yanlışlıkla kapıyı açan pedala bastı ve kedi kaçıp balığı yedi. Bu, aynı kedi ile birkaç kez tekrarlandı. Denemeler ilerledikçe, kedinin kapıyı açması daha az zaman aldı. Sonunda kedi kutuya konur konmaz kaçacaktı. Thorndike, yapboz kutusundan kedilerin deneme yanılma yoluyla öğrendiğini buldu. İnsanların da aynı şekilde öğrendiğini düşünüyordu. Bu onun

'damgalama' teorisini oluşturdu. Thorndike'a göre, bir kişi bir duruma bir dizi tepki verecek ve bunlardan en az biri tatmin edici bir duruma - bu duruma bir çözüme - yol açacaktır. Damgalama, o belirli yanıtın bağlantısının bu durumda hoş bir çözüm sunduğu zamandır. Bu tür öğrenmeye pekiştirme denir. Öğrenme, davranışın sonuçları nedeniyle gerçekleşir, davranış, hoş olmayan bir şeyin ortadan kaldırılması veya her ikisi için de hoş sonuçlara yol açar (Lefrançois, 2000).

Thorndike, etkili tepkilerin "başarı tarafından seçildiğini" (Thorndike, 1911) fark etti ve davranışın tanımlanması için üç unsurun gerekli olduğunu anladı. Bunlar durum, eylem ve sonuç öğeleridir (Chance, 1999). Şekil 2.4'de Thorndike'ın yapboz kutusunun temsili gösterilmiştir.

Şekil 2.4.Thorndike yapboz kutusu

Thorndike'ın teorisinin iki temel kavramı vardı. Bunlardan biri etki kanunu diğeri ise deneme yanılmayla öğrenmedir. Etki kanunu ile sonucundan memnun kalınan davranışların artacağını, memnun kalınmayan davranışların ise azalacağını söyler. Bir uyaran ile bir tepki arasındaki bağlantı güçlendirilebilir veya zayıflatılabilir. Bu bağlantı, sık pratik yaparak güçlendirilebilir veya uygulamayı bırakarak zayıflayabilir. Deneme ve yanılma yoluyla öğrenme ise, insan veya hayvan belli bir problemle karşılaştığında problemin çözümüne katkısı olmayan başarısız davranışları eler, problemleri çözen ya da başarıya götüren davranışları ise seçer. Thorndike bu duruma seçme (eleme) ve bağlama adını verir.

Thorndike gibi Skinner de davranış ve sonuç ilişkisi üzerinde durmuştur. Thorndike'ın çalışmalarından hareket eden Skinner, organizmanın davranışlarını uyarıcılara karşı gösterilen otomatik bir tepki olmaktan çok kasıtlı olarak yapılan hareketler olarak kabul

etmektedir. İnsanların herhangi bir ihtiyaç durumunda organizmanın kendiliğinden ortaya koyduğu davranışlara "edim" adını veren Skinner, bu edimlerin, onları izleyen sonuçlardan etkilendiğini ileri sürmektedir. Skinner'in geliştirdiği edimsel koşullanmaya göre edimsel davranış; bilinen bir uyarıcı tarafından oluşturulmaz; organizma tarafından ortaya konur ve sonuçları tarafından kontrol edilir (Skinner, 1938).

Gereksinimleri organizmayı eyleme iterken, davranışlarına yön veren kuvvetlerin de güdüler olduğu bilinmektedir. O anda içinde bulunduğu şartlarla ilgili önceden öğrenmiş olduğu deneyimleri yoksa hedefe varmak için çeşitli tepki ve davranışlarda bulunarak denemeler yapacaktır. Duruma göre belli sayıda deneme yanılmanın sonunda hedefe ulaşacaktır. Böylelikle organizma ya bir ödül elde edecek ya da bir cezadan kurtulacaktır. Süreç içinde yaşanan tekrarlar sonucu hedefe ulaştırıcı tepkilerin sayısı artarken sonuca götürmeyen davranışlar elenir ve hedefe ulaştırıcı tepkiler giderek öğrenilmiş davranış durumuna gelir (Yeşilyaprak, 2005).

Kısacası eğitim psikolojinde davranışçı kuramlar öğrenmenin ödül ceza sistemi üzerine kurulduğunu düşünmüştür. Eğer bir davranışın sonucu olumlu sonuçlar getiriyorsa o davranış devam ettirilir aksi durumda ise o davranıştan kaçınılır ve böylece öğrenme gerçekleşir.

2.4.2. Pekiştirmeli Öğrenme

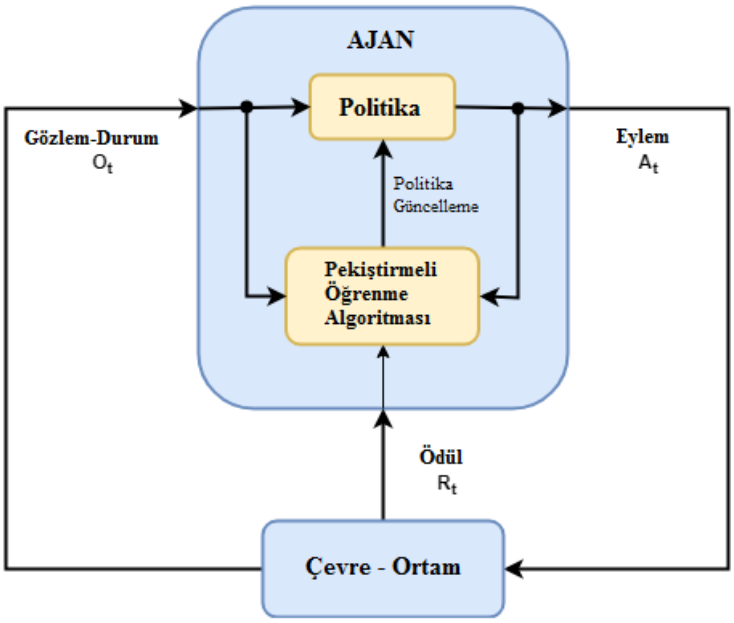
Pekiştirmeli Öğrenme(Sutton and Barto 2018), optimizasyon problemlerini çözmek için kullanılan bir Makine Öğrenmesi tekniğidir. Yapısal olarak, PÖ modelleri bir ajan ve çevresinden oluşur. Her ikisi arasında (ajan-çevre), ajanın yürütülen eylemlerin sonuçlarından sayısal ödüller şeklinde geri bildirim alarak deneme yanılma yoluyla öğrenmesini sağlayan çift yönlü bir iletişim kurulur. Şekil 2.5'te PÖ'nün genel yapısının şeması verilmiştir. Geleneksel Pekiştirmeli Öğrenme metodolojilerinin unsurları Tablo 2.2'de sunuldu. PÖ yapısının en önemli unsurları durumlar, eylemler ve ödüller olduğu söylenebilir.

- Durumlar: Ajanın çevreden aldığı anlık bilgilerdir. Bir robot için kameradan aldığı bir görüntü ya da bir sensörden gelen sıcaklık verisi örnek olarak verilebilir.
- Eylemler: Ajanın bir durum karşısında uyguladığı eylemdir. Örneğin; bir robot engele 5 cm yaklaşırsa uyarı sesi çıkarması.

- Ödüller: Ajanın bir durum karşısında uyguladığı eylemin başarısın ölçen skaler değerdir. Örneğin; bir robotun bir hedefe ulaşınca 100 puan alması gibi.

Tablo 2.2. Pekiştirmeli öğrenmenin parametreleri

Parametre	Adı	Tanımı
a_t	Eylem	Ajan tarafından yürütülen, bir dizi geçerli eylemden A seçilen komut
s_t	Durum	Bir dizi ortam temsili olan S durum uzayının somutlaştırılması
r_t	Ödül	Yürütülen bir eylem a_t tarafından tetiklenen $s_t \rightarrow s_{t+1}$ durum geçişine göre döndürülen sayısal ödül
$P(s_{t+1} \mid s_t, a_t)$	Durum Geçiş Modeli	Ajanın eylemlerine yanıt olarak ortamın nasıl değiştiğinin temsili
$\pi(s_t)$	Politika	Ajanın her durumda hangi eylemi benimsemesi gerektiğini belirten eşleme fonksiyonu
γ	İndirim Faktörü	Anlık ve gelecekteki ödüller arasındaki dengeyi ayarlar.
$V(s_t)$	Değer Fonksiyonu	Bir ajanın durumdan s_t alabileceği, beklenen toplam getiri
$Q(s_t, a_t)$	Eylem-Değer Fonksiyonu	s_t durumundaki eylemi a_t seçmek için beklenen toplam getiri
α	Öğrenme Oranı	Öğrenme sürecinde geçmiş deneyimlerin etkisini belirler



Şekil 2.5.Pekiştirmeli öğrenme yapısı

Pekiştirme Öğrenme algoritmalarında ajan, her *t* adımda, çevreyi gözler, gözlemlerini bir s_t durumu olarak belirler ve son olarak duruma göre uygulayacağı eylemi a_t seçer. Ajan daha sonra yeni durumu algılar s_{t+1} ve ortamdan bir ödül r_t alır. Ajan bu işlemi hedef duruma ulaşıncaya kadar ya da adım sınırı *T*'ye ulaşana kadar tekrarlar. Daha sonra çevre ilk halinde döndürülür (reset fonksiyonu) ve yeni bir bölüm başlar. Her bölümün *e*

sonunda, indirim faktörü γ tarafından indirimli olarak birikmiş toplam adım ödülleri elde edilir. Öğrenmenin amacı, akıllı ajanın, deneyimini kullanarak, birikmiş ödülleri R'yi en üst düzeye çıkarmak için karar verme yeteneğini geliştirmesidir. Ödül Denklem (2.2)'deki formüle edilir;

$$R_e = \sum_{i=0}^{T} \gamma^i . r_{t+i} \qquad (2.2)$$

2.4.3. Markov Karar Süreçleri

Pekiştirmeli öğrenme sistemlerinde ajan çevredeki her şey hakkında bilgi sahibi olamayabilir. Önemli olan ajanın öğrendiği bilgileri unutmamasıdır. Bu nedenle ideal bir sistemde, ilgili tüm bilgiler korunurken, geçmiş bilgileri bütüncül bir şekilde özetleyen bir durum sinyali istenir. Bilgiyi koruyan bu durum sinyalinin Markov özelliği olduğu söylenir(James, 2016). Ardışık durumların yalnızca mevcut duruma bağlı olduğu durumlarda, durumların Markov özelliğini yerine getirmesi gerekir. Örneğin, bir satranç oyunundaki piyonların mevcut konumu, oyunun sonraki süreci için önemli olan tek şey olduğundan, Markov durumu olarak hizmet edecektir. Ayrıca Markov özelliğiyle mevcut duruma sahip olmak, mevcut zaman adımına kadar tam geçmişe sahip olmakla eşittir (örneğin, piyon konumları tüm oyunu bu ana kadar özetler) (Roy, 2018).

Markov özelliğini sürekli bir durumda ve eylem alanında karşılayan bir pekiştirmeli öğrenme görevinin Markov Karar Süreci (MDP) olduğu söylenirken, sınırlı bir durumda ve eylem alanında olan görevin, sınırlı bir Markov Karar Süreci olduğu söylenir(James, 2016).

MDP, bir modelle ilgili karar vermemiz için bize matematiksel bir çerçeve sağlar. Bir PÖ ajanının çalıştığı ortam, ortamın tamamen gözlemlenebilir olduğu bir MDP olarak tanımlanabilir. Bu durum, geleceğin şimdiki zamanda verilen geçmişten bağımsız bir karar olduğunu belirten Markov özelliği olarak bilinir(Kardell ve Kuosku, 2017)

Hesaplamalı PÖ'de, ilgili ortam uzayı, ayrık zamanlı MDP (Puterman, 2005) olarak modellenebilenlerdir. Yüksek düzeyde, MDP'lerin alanı, bir sonraki ödülün ve dünyanın bir sonraki durumuna ulaşma olasılığının mevcut dünya durumu (ve belki de bir ajanın

eylem seçimi) tarafından tam olarak tahmin edilebileceği dünyaları tanımlar. Bir MDP aşağıdaki gibi tanımlanır.

- ***S***: Dünyanın olası durumların tanımlayan bir durum kümesi
- ***A***: Bir ajana sunulan olası seçenekleri tanımlayan bir eylem kümesi.
- **R :** S × A × S → [RMin, RMax]: Bir ödül işlevi.
- ***T*** **:** S × A → Δ(S): Mevcut durumda bir eylem yürütüldükten sonra dünyanın bir sonraki durumuna gelme olasılığını gösteren bir geçiş fonksiyonu.
- $\gamma \in [0, 1)$: Bir ajanın kısa vadeli ve uzun vadeli ödüller arasındaki tercihini gösteren bir indirim faktörü.
- $p_0 \in \Delta(S)$: Her durumda başlama olasılığı.

MDP'deki "Markov", geçiş işlevi T ve ödül işlevi R'nin her ikisinin de tam durum geçmişine değil, yalnızca dünyanın mevcut durumuna ve eyleme bağlı olduğunu gösterir. Denklem (2.3) ve Denklem (2.4), bir sonraki durum dağılımını ve mevcut durumdan sonraki ödülü ve yalnızca eylemi tam olarak karakterize eden işlevlerin mevcut olduğunu belirtir;

$$p(s', r|s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{2.3}$$

$$p(s'|s, a) = \Pr\{S_{t+1} = s'|S_t = s, A_t = a\} = \sum_{r \in R} p(s', r|s, a) \tag{2.4}$$

Bu varsayım, uygun genelliği korurken, analizi basitleştirmek için oldukça faydalıdır. Ayrıca, herhangi bir ortam Markov değilse, dünyanın son $k \in N$ adımları bellek açısından zengin, yeni bir durum temsili şeklinde gösterilebilir, böylece bir Markov modeli elde edilir. Bu şekilde, MDP'ler, bir ajanın *T(s′ / s, a)* ve ödül *R(s, a, s′)* durum dağılımını etkilemesine izin vererek Markov zincirlerini (Bremaud, 2000)ve Markov ödül süreçlerini (Reibman ve diğ., 1989) genelleştirir.

2.4.4. Değer Fonksiyonları

Bir ajan yeni bir duruma girdiğinde *t* adımında hangi eylemin yapılacağına karar vermesi için bu durumda olmanın ne kadar değerli olduğunu bilmelidir. Bir durumun iyiliğinin ölçüsü değer fonksiyonudur. Bir durumun değeri durum değeri fonksiyonu ve eylem değeri fonksiyonu olarak iki şekilde ölçülebilir. Ajanın gelecekte almayı bekleyebileceği ödüller, yaptığı eyleme bağlı olduğundan değer fonksiyonu politikaya göre tanımlanır. Bir politika π için durum-değer fonksiyonu, bir *s* durumundan başlarken ve sonrasında π'yi takip ederken beklenen getiridir ve sonlu MDP'ler için, $V^{\pi}(s)$ her *s* durumu için bir tabloda giriş olarak saklanır ve Denklem (2.5)'de gösterildiği şekilde tanımlanır;

$$V^{\pi}(s) = E_{\pi}(R_t|s_t = s) = E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s). \tag{2.5}$$

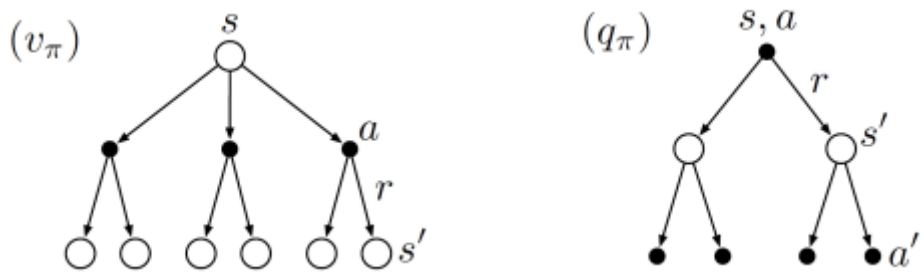
Burada E_{π}, politikanın π beklenen değeri olarak tanımlanır. Bir politika π için eylem-değer fonksiyonu, *s* durumunda başlayıp *a* eylemini gerçekleştirirken ve ardından π'yi takip ederken beklenen getiridir ve Denklem (2.6)'da gösterildiği şekilde tanımlanır;

$$Q^{\pi}(s,a) = E_{\pi}(R_t|s_t = s, a_t = a) = E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a). \tag{2.6}$$

Değer fonksiyonlarının temel bir özelliği, Bellman denklemi(Bellman, 1957a) olarak adlandırılan bir dizi özyinelemeli tutarlılık denklemini sağlama yeteneğidir. Denklem (2.7)'de gösterildiği şekilde tanımlanır;

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}(R_t|s_t = s) \\ &= E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s). \\ &= \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+2}|s_{t+1} = s'). \right] \\ &= \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s') \right]. \end{aligned} \tag{2.7}$$

Yukarıdaki son denklem, bir durumun değeri ile ardıl durumun değeri arasındaki ilişkiyi ifade eder. Her birini gerçekleşme olasılığına göre ağırlıklandırarak tüm olasılıkların ortalamasını almamızı sağlar. Başlangıç durumunun değerinin, beklenen sonraki durumun (indirimli) değeri ile ödüle eşit olması gerektiğini özetler. Bu durum, Şekil 2.6'da her bir açık dairenin bir durumu temsil ettiği, her bir katı dairenin bir durum-eylem çiftini ve ödülü temsil ettiği, V^{π} *ve* Q^{π} için yedek diyagramlar şeklinde gösterilmektedir.



Şekil 2.6. Değer fonksiyonu diyagramı (Kersandt, 2018)

2.4.5. Optimal Değer Fonksiyonları

Politika π, durum-eylem eşlemesidir ve ajanların beklenen geri dönüşlerini en üst düzeye çıkarmaya çalışır. Pekiştirmeli öğrenme görevinde amaç, uzun vadede en fazla ödülü elde eden politikayı bulmaktır. Bu nedenle, tüm durumlar için bu politikanın beklenen getirisi π'ye eşit veya bundan büyükse, π politikasının bir π' politikasına eşit veya daha iyi olduğu kabul edilir. Denklem (2.8) bunu ifade eder;

$$V^{\pi}(s) \geq V^{\pi'}(s) \rightarrow \pi \geq \pi', \forall s \in S. \tag{2.8}$$

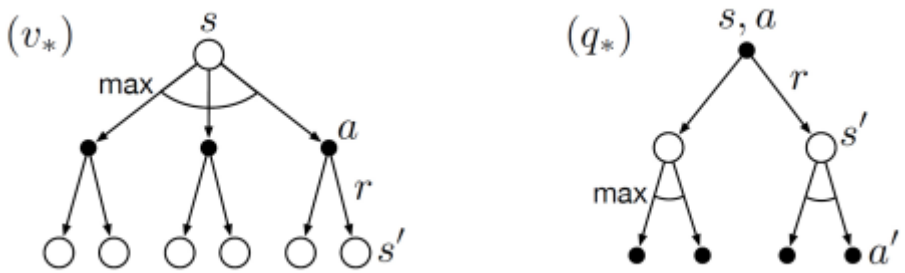
Optimal politikaları π^*, diğer tüm politikalardan daha iyi veya bunlara eşit politikalar olarak tanımlarız. Sonlu MDP'ler için, diğer tüm politikalara karşı Denkelem(2.8)'i her zaman yerine getiren ve optimal politika olarak adlandırılan ve π^*, ile gösterilen en az bir politika vardır. Bu bize, her duruma veya durum-eylem çiftine, herhangi bir politika tarafından elde edilebilecek en büyük getiriyi atayan optimal değer fonksiyonlarını tanımlama yeteneği verir. Optimal durum-değer fonksiyonu V^* ile gösterilir ve Denklem (2.9)'da gösterildiği şekilde tanımlanır;

$$V^*(s) \geq max_{\pi} V^{\pi}(s), \forall s \in S, \tag{2.9}$$

Optimal politikalar aynı zamanda Q^* ile gösterilen ve Denklem (2.10)'daki gibi tanımlanan aynı optimal eylem-değer fonksiyonunu paylaşır;

$$Q^*(s,a) \geq max_\pi Q^\pi(s,a), \forall s \in S, a \in A. \tag{2.10}$$

Şekil 2.7'de, Bellman denklemi altında $v*$ ve $q*$ için yedek diyagramları göstermektedir. Bu, grafiksel olarak, ajanın seçim noktalarında, bazı politikalara verilen beklenen değer yerine maksimumun seçildiğini gösterir. Bu anlamda tek adımlı bir arama yapılır ve optimal değer fonksiyonları için bu tek adımlı aramadan sonra en iyi görünen eylemler optimal eylemler olacaktır. Bu nedenle, $v*$ dikkate alındığında açgözlü davranan herhangi bir politika optimal bir politikadır. Özellikle, $v*$ gelecekteki olası tüm davranışların ödül sonuçlarını zaten hesaba kattığından, kısa vadeli sonuçların bu değerlendirmesi uzun vadede de optimaldir. Böylece, uzun vadeli optimal eylemler, tek adımlı bir ileriye dönük bakış açısına indirgenir. Bu nedenle, V^*optimal değer fonksiyonuna göre açgözlü olan herhangi bir politika, aslında bir optimal politikadır.



Şekil 2.7. V* ve Q* yedek diyagramları

Denklem (2.8)'deki Bellman denklemi ve V^*'nin bir politika için basitçe bir değer fonksiyonu olduğu göz önüne alındığında, V^* sabitlik koşulunun herhangi bir özel politikaya başvurmadan özel bir biçimde yazılabileceğini gösterebiliriz. Bu, V^* için Bellman optimallik denklemine yol açar ve Denklem (2.11)'deki gibi gösterilir;

$$\begin{aligned} V^\pi(s) &= max_a Q^{\pi^*}(s,a) \\ &= max_a E_{\pi^*}(R_t | s_t = s, a_t = a) \\ &= max_a E_{\pi^*}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a). \\ &= max_a E_{\pi^*}\left[r_{t+1} + \gamma E_\pi(\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a) \right] \end{aligned} \tag{2.11}$$

$$= max_a E[r_{t+1} + \gamma V^{\pi}(s_{t+1}|s_t = s, a_t = a)]$$

$$= max_a \sum_{s}^{'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s')\right]$$

Benzer şekilde, Q^* için Bellman optimallik denklemi Denklem (2.12)'deki gibi tanımlanır;

$$Q^{\pi}(s,a) = \sum_{s}^{'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma max_{a'} Q^{\pi}(s',a')\right]. \quad (2.12)$$

N durumları ve bilinen dinamikleri olan bir ortam göz önüne alındığında, her durum için V^* değerini çözmek için Bellman optimalite denklemleri kullanılabilir.

Sonlu Markov karar problemleri, model tabanlı ve modelden bağımsız ortamlar olarak ayırt edilir. Model tabanlı ortamlar için, tüm olası durumlar *S*, eylemler *A*, durum geçiş olasılıkları *p(s'|s,a)* ve anlık ödüller *r(s,a,s')* hakkında tam bilgi mevcuttur. Burada, çözüm uygulamadan önce bulunup değerlendirilebildiğinden, problem algoritmik planlamanın bir parçası haline gelir. Modelsiz ortamlar için, ortam hakkında hiçbir bilgi yoktur ve bu nedenle ajan, örnekler şeklinde deneyimler toplamak zorundadır.

2.4.6. Politika Belirleme

Daha önce belirtildiği gibi, Pekiştirmeli Öğrenme, beklenen toplam ödülleri en üst düzeye çıkarmayı amaçlar. Politika *π,* durumlarla eylemleri eşleyerek, ona göre bir değer fonksiyonu çıkarmayı ifade eder. En iyi politika oluşturmaya ve uygulanma durumuna göre, PÖ yöntemleri Modelden Bağımsız ve Model tabanlı olarak kategorize edilebilir. Modelden bağımsız yöntemlerde ajan, doğrudan deneyerek bir değer fonksiyonu oluşturur. Modele bağlı yöntemlerde ise ajana çevre hakkında bilgi verilir.

PÖ, bir politikayı değerlendirmek ve geliştirmek için Politika Değerlendirme teknikleri olarak adlandırılan çeşitli algoritmalara başvurur(Sutton ve Barto, 2018).Bunlar; Dinamik Programlama (DP), Monte Carlo (MC) ve Zamansal Fark (TD) metodolojileridir.

2.4.7. Dinamik Programlama

Tam bilgiye sahip sonlu MDP'ler için, optimal politika, gerçek bir ajan-ortam etkileşimi olmaksızın tamamen hesaplamalı yinelemeli bir yolla bulunabilir. Bu yaklaşıma sahip

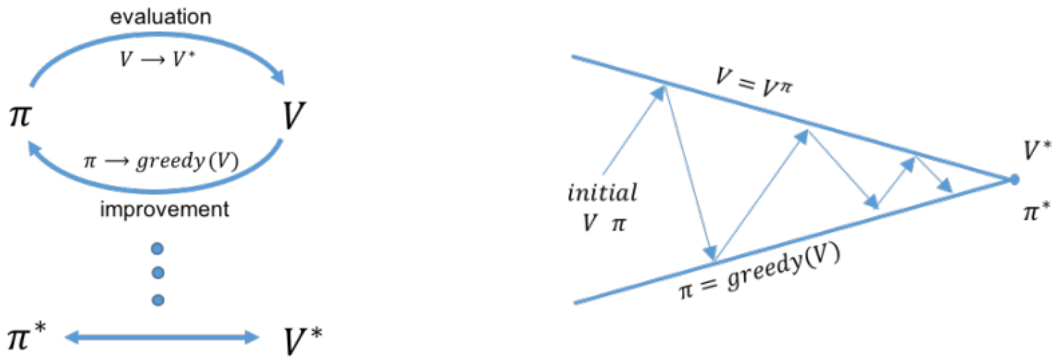
algoritmalar, dinamik programlama terimi altında sınıflandırılır. DP içinde kullanılan iki yöntem vardır. Bunlar politika yinelemesi ve değer yinelemesidir. Politika yinelemesi 2 adımdan oluşur. Bunlar, politika değerlendirmesi ve politika iyileştirme adımlarıdır.

Bu iki yöntem, optimal politikaları ve değer fonksiyonlarını güvenilir bir şekilde hesaplamak için kullanılabilir ve her ikisi de politika değerlendirmesi ve politika geliştirmenin iki hesaplaması arasında dönüşümlü olarak elde edilir.

İlk olarak, mevcut politika için değer fonksiyonunun yinelemeli hesaplamasını belirleyen bir dizi politika değerlendirmesi uygulanır. Ardından, değer fonksiyonuna göre en iyi eylemi açgözlülükle seçilerek iyileştirilmiş bir politikanın hesaplanmasını belirleyen bir dizi politika iyileştirmesine geçer. Yukarıdaki süreç, politika artık değişmeyene kadar tekrarlanır.

Politika yinelemesinde, politika iyileştirmesi yalnızca politika değerlendirme adımında çalıştırılır. Buna karşılık, değer yinelemesi, her bir politika iyileştirme adımı arasında yalnızca tek bir politika değerlendirme yinelemesini çalıştırır.

Genel politika yinelemesini, Şekil 2.8'teki gibi özetlenebilir.



Şekil 2.8. Genelleştirilmiş politika yinelemesi (James, 2016)

Model tabanlı yöntemler, politika/değer işlevini geliştirmek için ağırlıklı olarak DP (Sutton and Barto 2018) kullanır. Geçiş ve ödül modelleri, gelecekteki durumlar üzerinden beklenen ödül toplamının doğru bir şekilde hesaplanmasını sağlar. Değer fonksiyonu $V(s_t)$ aşağıdaki Denklem (2.13)'e göre güncellenir;

$$V(s_t) \leftarrow r_t + \gamma . \sum_{s_{t+1} \in S} P(s_{t+1}|s, a_t) . V(s_{t+1}) \quad (2.13)$$

Denklem (2.13)'ün son terimi, indirimli toplam gelecek ödülleridir. Her olası s_{t+1} durumu için, P geçiş modelinin tahmini getirileri $V(s_{t+1})$ ile toplamından elde edilir. DP, önyükleme(bootstrapping) adı verilen bir tekniğe izin verir. Önyükleme, ardıl durumların tahminlerine dayalı olarak her bir durum için değerlerin $V(s_t)$ tahminlerini güncellememizi sağlar.

2.4.8. Monte Carlo

Bilinmeyen bir ortama sahip sonlu bir MDP için, Monte Carlo(Rothman 1984) yöntemleri, örnek bölümler biçimindeki deneyimlerden değer fonksiyonlarını ve optimal politikaları öğrenebilir. MC yöntemleri, çevreden örnek durum, eylem ve ödül dizilerinden öğrenmemizi sağlayan modelden bağımsız bir yöntemdir. Çevrenin dinamikleri hakkında önceden tam bilgi sahibi olmayı gerektirmez. Monte Carlo yöntemleri, her olası durumun değerini hesaplamak için bir model kullanmak yerine, V^{π} ve Q^{π} değer fonksiyonlarını deneyimden tahmin edebilir(Aghaei, 2019).

MC kontrol yöntemlerinde ajan, ödüller ve çevre hakkında bilgi edinmek için keşif-sömürü ödünleşmesini göz önünde bulundurmalıdır. Ajanın hem daha önce kullanılmayan eylemleri hem de olumsuz ödüllere yol açabilecek belirsiz eylemleri göz önünde bulundurarak keşfetmesi gerekir. Güvenli bir şekilde hareket etmeli ve iyi bilinen ödüllere bağlı kalmalı veya daha yüksek ödüller keşfetmek için yeni şeyler deneme riskine girmelidir.

MC kontrol yöntemlerinde yeterli keşif yapıldıktan sonra bunun sürdürülüyor olması bir sorundur. Genel olarak, bunu çözmek için kullanılabilecek iki yaklaşım, politika içi yöntemler ve politika dışı yöntemlerdir. Politikaya dayalı yöntemler, bir yandan araştırma yaparken bir yandan da karar vermek için kullanılan politikayı değerlendirmeye veya iyileştirmeye çalışır. Politika dışı yöntemler ise, karar vermek için kullanılan politika ile ilgisiz olabilecek deterministik bir politika öğrenmeye çalışır.

Monte Carlo yöntemleri, DP'den farklı olarak önyükleme yapmaz. Bunun yerine, değer fonksiyonu güncellemeleri, örneklenmiş bir ortamdan durum-eylem-yeni durum-ödül (s_t, a_t, s_{t+1}, r_t) şeklinde Denklem (2.14)'deki gibi hesaplanır.

π politikasını izleyerek ve karşılaşılan her durum için, o belirli durumu takip eden gerçek r_t getirilerinin bir ortalamasını koruyarak elde edilir. Daha sonra, belirli bir durumla karşılaşılma sayısı sonsuza yaklaştıkça ortalama, durumun değerine, $V^{\pi}(s)$ yakınsayacaktır. Benzer şekilde, belirli bir durumda gerçekleştirilen her eylem için ayrı ortalamalar tutulursa, bu ortalamalar eylem değerlerine, $Q^{\pi}(s,a)$ yakınsayacaktır. Buna göre, durum-değer fonksiyonu $V(s_t)$, durumla her karşılaşıldığında, gerçek dönüş r_t 'ye doğru güncellenebilir;

$$V(s_t) \leftarrow V(s_t) + \alpha.[r_t - V(s_t)] \tag{2.14}$$

α, öğrenme oranını etkileyen adım boyutu parametresi olarak adlandırılan küçük bir pozitif kesirdir. Değer işlevleri, değer işlevlerinin artık hesaplanmadığını, ancak örneklenmiş getiriler temelinde tahmin edildiğini vurgulamak için büyük harfle gösterilir.

2.4.9. Zamansal Fark

Zamansal fark öğrenme (Sutton, 1988), DP'nin (önyükleme yoluyla öğrenme yeteneği) ve MC'nin MDP'ye erişmeden doğrudan ortamdan alınan örneklerden öğrenme yeteneğinin bir birleşimidir.

MC yöntemlerinden farklı olarak, TD'nin değer işlevini güncellemek için bölümün sonuna kadar beklemesi gerekmez. Bunun yerine, TD yöntemleri, yeni değerin eski tahminden ne kadar farklı olduğunu bize bildirmek için zamansal hatalar kullanarak yalnızca bir sonraki zaman adımına kadar bekler.

Bu güncelleştirme genel olarak şu şekildedir:

$$YeniTahmin \leftarrow EskiTahmin + AdımSayısı[Hedef - EskiTahmin]$$

TD yöntemleri, modelden bağımsız oldukları için, DP yöntemlerine göre büyük bir avantaj sunarken, çevrimiçi, tamamen artımlı güncellemeleri MC yöntemlerini geliştirir.

Bu, özellikle uzun, muhtemelen sonsuz bölümlerle uğraşırken önemlidir, bu nedenle güncellemeleri bir bölümün sonuna kadar ertelemek ideal değildir.

TD, ortamın örneklemini MC'den yaklaşık bir beklenti durumu (sonraki durum dağılımı) ve DP'den gelecekteki ödüllerin indirimli toplamını tahmin etmek için önyükleme kavramıyla birleştirerek değer işlevini Denklem (2.15)'te gösterildiği gibi güncelleştirir;

$$V(s_t) \leftarrow V(s_t) + \alpha.[r_t + \gamma.V(s_{t+1}) - V(s_t)] \quad (2.15)$$

2.4.10. Keşif ve Sömürü

Pekiştirmeli öğrenmede keşif, ajanın çevre hakkındaki bilgisini geliştirmek için eylemde bulunmasını sömürü ise mevcut bilgisine göre ödülleri en üst düzeye çıkarmak için eylemde bulunmasını ifade eder. Bir ajanın amacı, bilinmeyen bir ortamla etkileşimler yoluyla elde edilen gelecekteki ödüllerin toplamını en üst düzeye çıkarmaktır. Bunu yaparken, ajan keşif ve sömürüyü dengelemelidir. (Taïga ve diğ., 2018).Pekiştirmeli öğrenmede ortaya çıkan zorluklardan biri, keşif ve sömürü arasındaki dengedir. Ajan, çok fazla ödül elde etmek için daha yüksek ödüller üretmede etkili olan deneyimlerine ve eylemlerine güvenmelidir. Öte yandan, bu iyi eylemler ilk etapta daha önce seçmediği eylemleri deneyerek keşfedilmelidir. Başka bir deyişle, ajan ödül almak için zaten bildiklerinden yararlanmak zorundadır, ancak aynı zamanda gelecek için olası daha iyi bir eylem bulmak için araştırmak zorundadır.

PÖ ajanları en uygun politika hakkında bilgi edinmek isterler, ancak ilk bölümde karşılaştıkları "iyi" eylemleri asla araştırmazlarsa ve yalnızca güçlendirirlerse, en uygun politikanın neye benzediğini bilemezler. Politika içi yöntemlerde, ajan her zaman keşfeden bir yapıdadır ve bu nedenle hala keşif yapan en iyi politikayı bulmaya çalışır. Buna karşılık, politika dışı yöntemler iki ilke kullanarak bu uzlaşmayı önler. Hakkında bilgi edinilen politikaya hedef politika π adı verilir. Davranış oluşturmak için kullanılan politikaya davranış politikası μ adı verilir ((Roy, 2018).

Modelden Bağımsız PÖ algoritmalarında keşif ve sömürü arasındaki geçiş açgözlü(ε-açgözlü) stratejilerle kurulur. Açgözlü stratejilerde, ajan uygulayacağı eylemi epsilon-ε olasılıkla rastgele seçer. Buna karşılık sömürüde ise, en yüksek tahmini değere sahip

eylemler olan *Q*(s,a)* tamamlayıcı olasılıkla seçilir. Zaman içinde ε değeri azaltılarak, ajan keşiften sömürüye doğru ilerler (Arulkumaran ve diğ., 2017).

Bir davranış politikasının çok basit ama etkili bir versiyonu epsilon-açgözlülük politikasıdır. Bu yöntemle keşif miktarı, eylem seçimlerinde rastgeleliği belirleyen bir parametre olan ε ile global olarak kontrol edilir. Diğerlerinin aksine, ε-açgözlülüğün bir avantajı, keşfe özel verilerin ezberlenmesine gerek olmamasıdır, bu da yöntemi çok büyük ve hatta sürekli durum uzayları için özellikle avantajlı kılmaktadır. Diğer karmaşık yöntemlerle karşılaştırıldığında, ε-açgözlülüğü (Vermorel ve Mohri, 2005) genellikle ilk tercih edilen yöntem olduğu bildirilmiştir (Sutton ve diğ., 2018).Denklem (2.16)'da ε-açgözlü yöntem formüle edilmiştir;

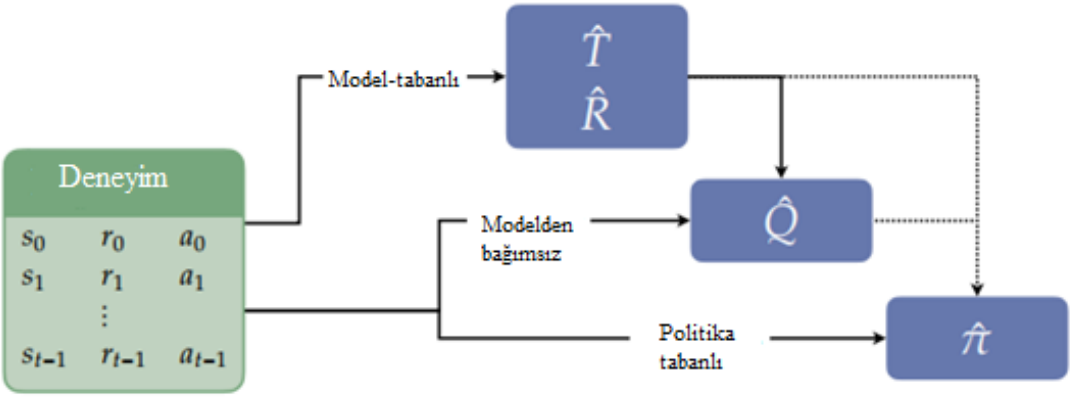
$$\mu(s_t) = \begin{cases} argmax_a Q(s_t, a) & 1 - \varepsilon \; olasılıkla \\ random\; A & \varepsilon \; olasılıkla \end{cases} \tag{2.16}$$

2.4.11. Q Öğrenme

PÖ algoritmaları politika tabanlı, modelden bağımsız ve model tabanlı olarak üç kategoriye ayrılabilir. Her kategori şu sorunun cevabını arar: "Öğrenme sırasında hangi fonksiyonlar tahmin ediliyor?". Bir PÖ algoritması geçiş ve ödül fonksiyonlarının tahminlerini tutarsa (*T* ve *R)*, o zaman model tabanlı olduğu söylenir. Ödül ve geçiş fonksiyonları tahmin edilmiyorsa, sadece eylem-değer fonksiyonu *Q* tutuluyorsa algoritma modelden bağımsızdır. Son olarak, tahmin edilen tek şey doğrudan politika ise o zaman politika tabanlıdır. Bununla birlikte, bunlar çokta net olmayan sınırlardır çünkü birçok algoritma genellikle farklı işlevlere kısmi çözümler hesaplar veya bu işlevlerden birinin inşasına benzeyen örtük hesaplama çalışmaları yürütür (Seijen ve Sutton, 2015). Bu üç yaklaşım arasındaki ayrım kabaca Şekil 2.9'daki şema ile gösterilmiştir.

Model tabanlı PÖ'de, geçiş ve ödül işlevleri genellikle açıkça tahmin edilir. Daha sonra, bu tahminleri kullanarak *T* ve *R,* ajan genellikle iyi davranış için arama yapmak veya farklı politikaları değerlendirmek için kullanılabilecek simüle edilmiş bir MDP, *M̂* oluşturur. Diğer bir deyişle, çevresel MDP *M*'ye yeterince benzeyen bir MDP *M̂*'ye simülasyon erişimi verildiğinde, ajan *π* veya belki de *Q* oluşturmak için *M̂* üzerinde hesaplamalar yapabilir ve bu da en yüksek değere sahip eylemi seçerek bir politikayı indükleyebilir.

Modelsiz ve politika tabanlı PÖ'de, ajan genellikle eylem-değer fonksiyonu (*Q)'* nun veya bir politika (*π)* nın tahminini doğrudan tutar. Bu işlevleri daha verimli bir şekilde atayarak, daha sağlam bir şekilde genelleştirerek veya daha hassas bir şekilde keşfederek daha hızlı öğrenmek için çeşitli mekanizmalar ayarlanır.



Şekil 2.9. PÖ algoritmalarının kategorileri(Abel, 2020)

Bağlama bağlı olarak her bir algoritma türünü kullanmak için iyi argümanlar vardır. Özellikle, modelsiz ve politika tabanlı yöntemler, derin sinir ağları ile birleştirildiğinde büyük bir başarı elde etti ve Atari'den (Mnih ve diğ., 2015) robotiğe kadar çeşitli zorlu alanlarda etkili bir şekilde öğrenen DPÖ yöntemlerine yol açtı(Levine ve diğ., 2016).

En bilinen PÖ algoritması, ilk olarak Watkins ve Dayan (Watkins ve Dayan, 1992)tarafından tanıtılan Q-öğrenme olarak adlandırılır. Optimal kontrol teorisi bağlamında, Q-öğrenme, tamamen bilinmeyen sistemler için en uygun kontrol çözümüne çevrimiçi olarak yakınsayan, uyarlanabilir, bir kontrol algoritması olarak sınıflandırılabilir(Lewis ve diğ., 2012).

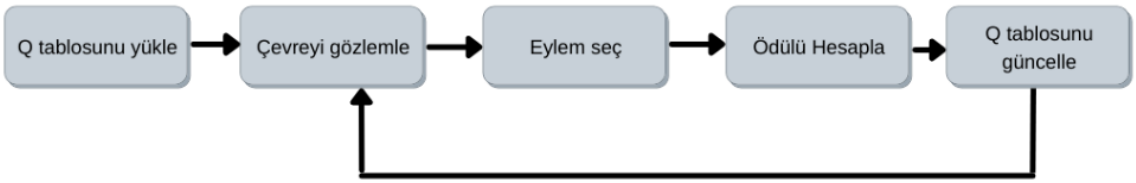
Q-öğrenme, her durum-eylem çifti için *Q* fonksiyonunun bir tahminini korur ve son deneyime (s_{t-1}, a_{t-1}, r_{t-1}) ve bir öğrenme oranına $\alpha \in [0, 1]$ dayalı olarak bu *Q* fonksiyonu tahminine basit bir güncelleme yapılması temelinde ilerler. Yani, ilk önce *Q* değerlerini [*QMin,QMax*] aralığından rastgele seçmek gibi bazı protokollere göre bir *Q* fonksiyonu uygulanır veya daha yaygın olarak, ilk Q fonksiyonu sıfır olarak ayarlanır veya tüm *s, a* için $Q_0(s, a) = QMax$ olduğunda iyimser olarak ayarlanır. Ardından, Denklem (2.17)'deki gibi tanımlanan açgözlülük politikasına göre eylemler seçilir;

$$\pi_{Q,\varepsilon}(a|s) = \begin{cases} 1-\varepsilon & a = argmax_{a'} Q(s_t, a'), \\ \dfrac{\epsilon}{|A|-1} & diğer\ durumlarda. \end{cases} \quad (2.17)$$

Q-Öğrenme (Watkins, 1989), takip edilen politikadan bağımsız olarak doğrudan Q^*'ya yaklaşan bir politika dışı TD kontrol algoritmasıdır. Bir deneyim, (s, a, r ,s') olarak tanımlanır, burada ajan s durumunda başlar, a eylemini gerçekleştirir, bir r ödülü alır ve s' durumuna geçer. *Q(s,a)* güncellemesi daha sonra s' 'dan bir eylem için mümkün olan maksimum ödülü alarak ve Denklem(2.18)'deki güncellemeyi uygulayarak gerçekleştirilir;

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha.[r + \gamma.max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (2.18)$$

Q-Öğrenme, Şekil 2.10'da gösterildiği gibi bir Q-Tablosunu yükleyerek başlar, her satır bir duruma, s ∈ S'ye ve her sütun bir eyleme, a ∈ A'ya karşılık gelir.



Şekil 2.10. Q-Öğrenme akış şeması

	$Eylem_1$	...	...	$Eylem_N$
$Durum_1$	Q(1,1)	...	...	Q(1,N)
...	...	...	...	...
...	...	...	...	...
$Durum_M$	Q(M,1)	...	...	Q(M,N)

Şekil 2.11. Q değeri tablosu

İlk aşamada ajan, ε-açgözlü keşif yöntemini kullanarak yeni durumları ve eylemleri keşfederek çevre ile etkileşime girer. Q değerleri *Q(s, a)*, ajanın *s* durumunda *a* eylemini gerçekleştirmesinden beklenen toplam ödüllerini temsil eder. Şekil 2.19 da verilen Q-tablosu yapısına benzer bir şeklide, her bir durum-eylem eşleşmesi için tablo Q değeri ile doldurulur. Her Q değeri, Bellman denkleminin hesaplanmasından kaynaklanır ve Denklem (2.19)'daki gibi ifade edilir;

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha.[r(s_t, a_t) + \gamma.max_a Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (2.19)$$

- $r(s_t, a_t)$: mevcut s_t durumundan bir sonraki duruma s_{t+1} geçiş için verilen ödül;
- $max_a Q(s_{t+1}, a)$: Ardışık durum s_{t+1}'in optimal *Q* değeri biçiminde, dizinin birkaç adım daha derininde döndürülebilen ödüller;
- $Q(s_t, a_t)$: Zamansal Fark olarak Q değeri gösterimi;
- $r(s_t, a_t) + \gamma \cdot max_a Q(s_{t+1}, a)$: Zamansal Fark ile hedeflenen değer;
- $r(s_t, a_t) + \gamma \cdot max_a Q(s_{t+1}, a) - Q(s_t, a_t)$: Zamansal Fark hatası (δt).

Ajan keşif yaparken çeşitli çevre durumları ile karşılaşır. Yeni durumlarla karşılaştıkça Q-Tablosu dolmaya başlar. Ajan, yeni durumlar keşfettikçe tablo büyür. Ajan keşiften sömürüye ilerlediğinde, Q-Tablosundaki değerlere bakar ve toplam birikmiş ödülleri en üst düzeye çıkaran eylemleri *Q*(s, a)* Denklem(2.20)'de gösterilen formül ile seçer;

$$a_t \leftarrow argmax_{a \in Q(s,:)}(Q(s_t, a)) \equiv Q^*(s_t, a) \tag{2.20}$$

Q-öğrenmenin algoritması Şekil 2.12'de gösterilmiştir.

Q-Öğrenme Algoritması

Q(s,a)'yı rastgele başlatın

İskonto faktörü γ belirleyin

Adım boyutu parametre α belirleyin

tekrarla(her bölüm için):

Başlangıç durumu S_1'i gözlemleyin

tekrarla (bölümün her adımı için):

Eylemi seçin A_t Q'dan türetilen politikayı kullanırken (örn. ε-açgözlü)

Eylemi uygula

Ödül R_{t+1} ve yeni durum S_{t+1}'i gözlemleyin

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha.[r(s_t, a_t) + \gamma.max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$

$S_t \leftarrow S_{t+1}$

S_t son durum olana **kadar devam et**

öğrenmenin sonuna kadar devam et

Şekil 2.12. Q-Öğrenme Algoritması

2.4.12. Sarsa

Sarsa, State-Action-Reward-State-Action anlamına gelen, politikaya bağlı bir TD kontrol algoritmasıdır. Bu ad, ajanın *s* durumunda başladığı, *a* eylemini gerçekleştirdiği, *r*

ödülünü aldığı, *s'* durumuna geçtiği ve ardından *a'* eylemini yapmaya karar verdiği bir deneyimden (*s; a; r; s'; a'*) türetilir. Bu deneyim, Denklem (2.21)'deki denklemi kullanarak *Q(s; a)*'yı güncellemek için kullanılır;

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha.[r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]. \quad (2.21)$$

Algoritmanın genel formu aşağıda Şekil 2.13'de verilmiştir.

Q(s,a)'yı rastgele değerlerle başlat.

tekrarla(her bölüm için):

s'yi başlat

Bölümün her adımı için:

Q'dan türetilen politikayı kullanarak bir s durumda bir eylem a seçin. (örn. ε-açgözlü)

a eylemini gerçekleştirin, r, s'yi gözlemleyin.

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha.[r(s_t, a_t) + \gamma.max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$

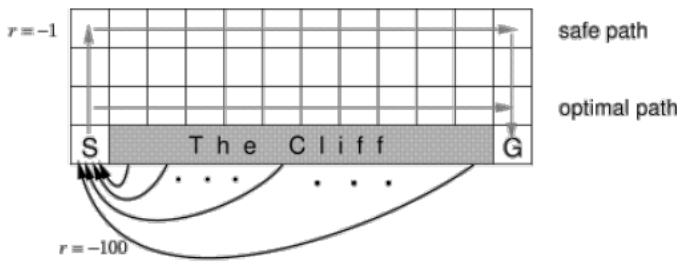
$S_t \leftarrow S_{t+1}$

Şekil 2.13. Sarsa algoritması

2.4.13. Sarsa ve Q Öğrenmenin Karşılaştırılması

Sarsa ve Q-öğrenme algoritmaları arasındaki farklar oldukça incedir. Sarsa politika içi bir yöntem olduğundan, Q değerlerini güncellemek için kullanılacak hamleleri yaparken bir kontrol politikası izler. Öte yandan Q-öğrenme, en uygun politikanın izlendiğini varsayan ve bu nedenle her zaman en iyi eylemi gerçekleştiren politika dışı bir yöntemdir. Özetlemek gerekirse, temel fark gelecekteki ödüllerin bulunma şeklidir.

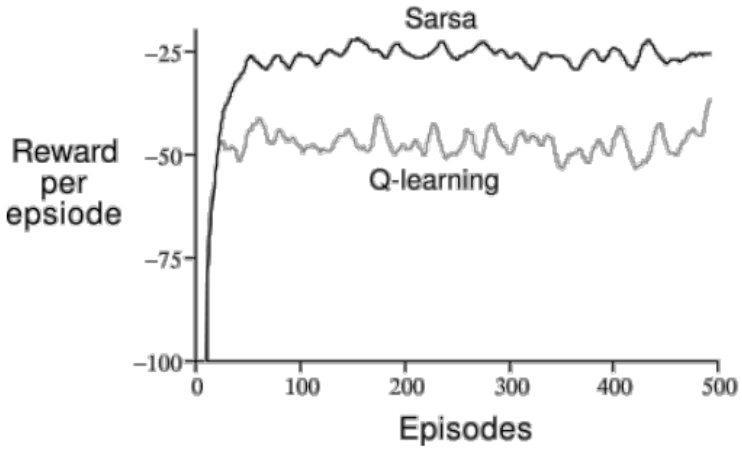
Bu iki yöntem arasındaki fark, Sutton ve Barto'nun (Sutton ve Barto, 1998) kitabından alınan örnekte (Şekil 2.14) iyi bir şekilde gösterilmiştir.



Şekil 2.14. Uçurumda yürüme görevinin grid dünyası (Sutton ve Barto, 1998)

Şekil 2.14'te gösterilen grid dünyası, indirim faktörü uygulanmamış ($\gamma = 1$), epizodik bir görevin parçasıdır. Görevin amacı, uçurumdan düşmeden yukarı, aşağı, sağ ve sol hareketlerini kullanarak başlangıç durumundan (S) hedef duruma (G) gitmektir.

Ajan, "Uçurum" olarak işaretlenmiş bölgeye girmesi dışında her durum geçişinde -1 ödül alır, uçuruma düşerse -100 ödül alır ve ardından başlangıç durumuna geri gönderilir. Ajan görevini yerine getirirken, sabit bir epsilon $\varepsilon = 0.1$'e ayarlı olarak ε-açgözlü bir eylem seçimini izler.



Şekil 2.15. Uçurumda yürüme görevinin sonuçları (Sutton ve Barto, 1998)

Şekil 2.15'te bir Q-öğrenme kontrol yöntemi ve bir Sarsa kontrol yöntemi izlendiğinde her bölüm için toplam ödülü göstermektedir.

Q-Öğrenme bazen ε-açgözlü eylem seçimini izleyerek ajanı uçurumdan itecek rastgele bir eyleme yol açmasına rağmen kısa bir süre sonra uçurumun kenarı boyunca seyahat etmeyi içeren en uygun politikayı öğrenmeyi başarır. Tersine, Sarsa bunu dikkate alarak eylem seçim yöntemi uçurumdan uzakta, daha uzun ama daha güvenli bir yol izleyen bir politika ile sonuçlanır. Q-Öğrenme'nin en uygun politikayı bulmasına rağmen, performansı Sarsa'dan daha kötüdür, ancak her ikisi yöntemde de kademeli olarak epsilon değeri 0'a düşürülürse optimal politikaya yaklaşacaktır.

2.4.14. Fonksiyon Yaklaşımı

Pekiştirmeli öğrenmenin zorluklarının çoğu, robotiğe uygulandığında ortaya çıkar. Bunun nedeni çoğu robotun doğası gereği sürekli durumlar ve eylemler ile hareket

etmesidir. Bunun gibi görevlerle uğraşırken, boyutsallığın laneti (curse of dimensionality) ile (Bellman, 1957b) karşı karşıya kalınır. Bu bize durum sayısının, durum değişkenlerinin sayısıyla katlanarak arttığını söyler.

Şimdiye kadar tartışılan yöntemlerin tümü, her durum için bir giriş içeren tablo olarak temsil edilen değer fonksiyonlarına sahiptir. TD yöntemleri ve Q-öğrenme, her durumun bir *V(s)* girişine sahip olduğu veya her durum-eylem çiftinin bir *Q(s,a)* girişine sahip olduğu tablo halinde tartışıldı. Tüm durumlar için tüm değer fonksiyonları bir arama tablosuna kaydedilir. Bu prosedür, dama veya tic-tac-toe oyunu gibi sınırlı sayıda durum ve eylem içeren ortamlar için uygundur. Ancak, problemin durum uzayı büyükse, bu tablo biçimi pratikte mümkün olmaz. Soru şudur: Mobil robotlarda olduğu gibi büyük, hatta sürekli bir durum alanı verildiğinde-örneğin bir hastane ortamı- ne olur? Otonom bir araç bağlamında, kamera görüntülerinin olası piksel düzenlemelerinin sayısı sonsuzdur. Her bir görüntü bir durumu temsil ettiğinden, durum uzayı sürekli veya sonsuz olarak ayarlanır. Sürekli durum alanı ile büyük miktarda zaman ve veri gerektiren büyük, bellek tüketen tablolar gelir. Belirli bir durum için bir değer aramaya çalışırken, önce bu değerin tabloda bulunması gerekir, bu da çevrimiçi güncellemeleri zorlaştırır. Öğrenme süreci çok sayıda durum tarafından yavaşlatılır, çünkü her bir durumun değeri ayrı ayrı öğrenilmelidir. Bu nedenle, bu gibi durumlarda öğrenmenin tek yolu, önceki durumlardan daha önce görmediğimiz durumlara genelleme yapmaktır. Bu sorunu büyük veya sonsuz MDP'ler için çözmek için, mevcut olana benzeyen farklı durumlarla önceki karşılaşmalardan genelleme yapmak gerekir.

Pekiştirmeli öğrenme bağlamındaki bu genellemeye fonksiyon yaklaşımı denir, çünkü *Q(s,a)* gibi istenen fonksiyondan veri örnekleri alır ve tüm fonksiyonun bir yaklaşımını üretmek için onlardan genellemeye çalışır. Bunu yaparken parametre vektörü θ kullanılarak fonksiyon Denklem (2.22)'deki gibi temsil edilir.

$$\hat{V}(s;\theta) \approx V_{\pi}(s)$$
$$\hat{Q}(s,a;\theta) \approx Q_{\pi}(s,a) \qquad (2.22)$$

Makine öğreniminde incelenen birincil konu olan denetimli öğrenmenin bir örneği olarak fonksiyon yaklaşımı, halihazırda kapsamlı bir şekilde incelenmiştir. Teoride, bu alanda

çalışılan yöntemlerden herhangi biri, fonksiyon tahmincisi rolünde kullanılacak pekiştirmeli öğrenme ile birleştirilebilir. (Schmidhuber, 2015).

2.5. Derin Pekiştirmeli Öğrenme

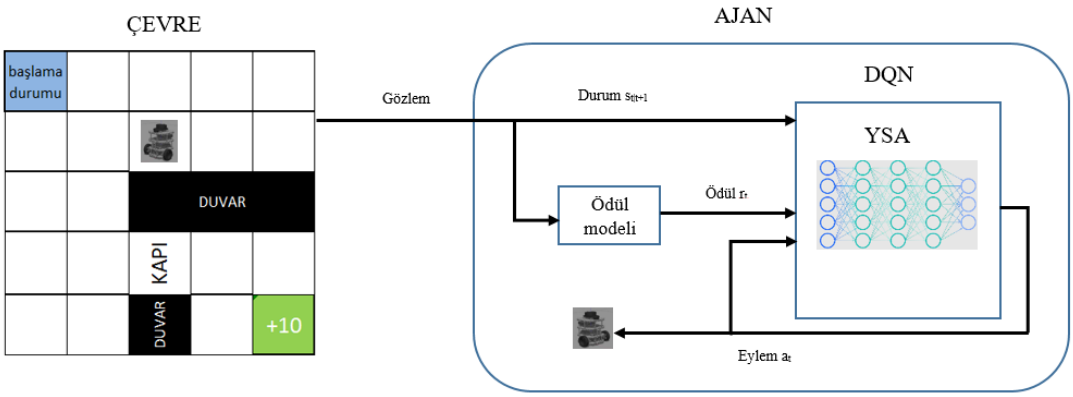
Pekiştirmeli öğrenmenin fonksiyon yakınlaştırıcısı olarak derin sinir ağları ile kombinasyonuna Derin Pekiştirmeli Öğrenme denir. Değerlerin tablo gösterimlerinin sinir ağı gibi bir fonksiyon yaklaşımıyla değiştirilmesi, yüksek boyutlu sensör girdilere sahip görevleri başarmak için gereklidir.

Pekiştirmeli Öğrenme, temel gerçeğe atıfta bulunmadan kontrol modellerini öğrenmenin etkili bir yoludur(Tai ve Liu, 2016). Bununla birlikte PÖ algoritmaları, tablo çerçeveleri nedeniyle kapsamlı ve dinamik ortamlarda hesaplama açısından maliyetli olma eğilimindedir. Bunu çözmek için derin öğrenme ve pekiştirmeli öğrenme algoritmaları birlikte kullanılabilir. Bu tür yapıları içeren yöntemler, bilimin tüm yelpazesinde uygulamalar üreten yenilikçi bir alan olan DPÖ kapsamındadır (Li, 2018).

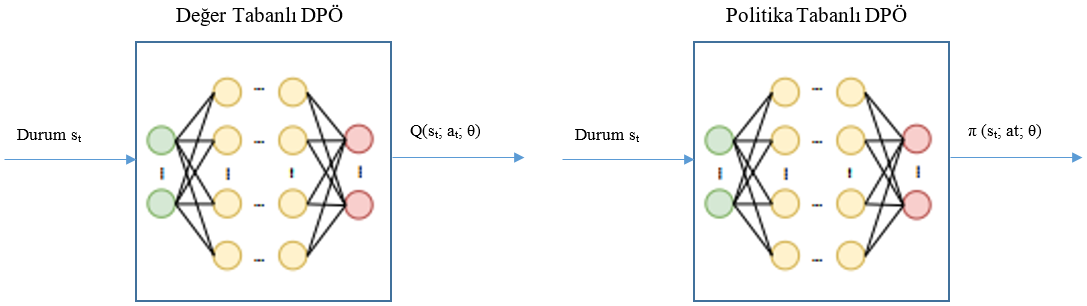
Pekiştirmeli Öğrenme ile ilgili sorunlardan biri, durum sayısı arttıkça verileri depolamak için gereken bellek miktarının hızla artmasıdır. Derin Pekiştirmeli Öğrenme, durum uzayı büyük veya sürekli olduğunda, görünenden görünmeyen durumlara genelleme yaparak, fonksiyon tahmin edicileri (Li, 2018) olarak Yapay Sinir Ağları'nı kullanarak bu problemin üstesinden gelmeye çalışır. Şekil 2.16'da DPÖ modeli gösterilmiştir.

Pekiştirmeli öğrenme için fonksiyon yaklaşımı olarak sinir ağları yeni bir fikir değildir ve 1989'a kadar uzanmaktadır (Werbos, 1989).Burada yazar, TD benzeri algoritmaları kullanarak politikaları ve değer fonksiyonlarını öğrenmek için hata geri yayılımı ile eğitilmiş sinir ağlarını kullanan bir yaklaşım geliştirdi. Öte yandan erken araştırmalar, politika dışı, doğrusal olmayan fonksiyon yaklaşımı ve önyükleme işleminin birleştirilmesinin kararsızlığa ve ayrışmaya neden olabileceğini göstermiştir(Tsitsiklis ve Roy, 1997), bu da ölümcül üçlü sorunu (Sutton ve Barto, 1998)olarak adlandırılır.

Sorunun arkasındaki sebep hala bir araştırma konusu olsa da 2015 yılında Google DeepMind'dan bir ekip, derin sinir ağlarını kullanan Q-öğrenmenin bir uyarlaması olan Derin Q-Öğrenme algoritmasını(Mnih ve diğ., 2015) sundu.



Şekil 2.16. Derin pekiştirmeli öğrenme modeli



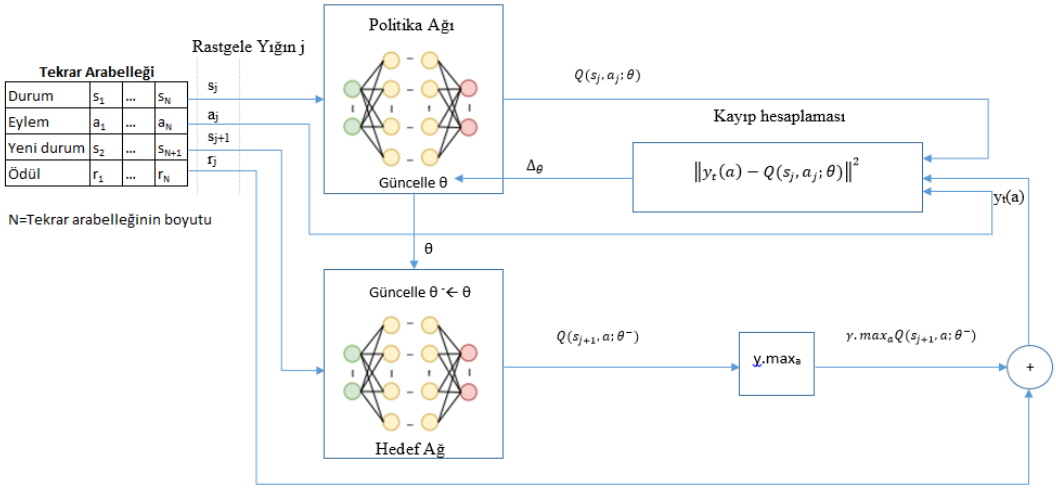
Şekil 2.17. Değer tabanlı ve Politika tabanlı Derin Pekiştirmeli Öğrenme

DPÖ'de, değere dayalı veya politikaya dayalı yöntemler kullanılır. Şekil 2.17'de bu yöntemlere ait modeller gösterildi. Değer tabanlı öğrenmede, YSA'lar eylem-değer fonksiyonları, *Q(s, a; θ)* olarak hareket eder. İnce ayar yapıldıktan sonra, durum-eylem *Q* değerlerini tahmin ederler ve mevcut durum göz önüne alındığında hangi eylemin gerçekleştirileceğine dair deterministik bir sinyal çıkarırlar (Ejaz ve diğ., 2019).Aksine, politika tabanlı yaklaşımlar, politika *π(s, a; θ)* parametresini ayarlamak ve Politika Gradyan tekniklerini kullanarak eylem alanını optimize etmek için YSA'ları kullanır (Wang ve diğ., 2019; Mnih ve diğ., 2016)

2.5.1. Derin Q Öğrenme

Derin Q-Öğrenme (Mnih ve diğ., 2015), Q-Öğrenme algoritması üzerine tasarlanmış değer tabanlı bir öğrenme yöntemidir. Yapısında bulunan YSA mimarisi nedeniyle Derin Q-Ağı olarak da adlandırılır. Her iki yaklaşım da durum-eylem Q değerlerini hesaplamak için aynı prensibi paylaşır, ancak işleyiş tarzında farklılık gösterir. Q-Öğrenme, eğitim

sürecini ve bunun sonucunda ajanın karar vermesini yönlendirmek için bir Q-Tablosuna başvururken, DQN'de bu tür görevler YSA'lar tarafından yürütülür.



Şekil 2.18. Derin Q-Öğrenme eğitim aşaması

Şekil 2.18'de gösterildiği gibi, DQN modeli, tekrar arabelleği, hedef ağ ve politika ağı, olarak üç ana bileşenden oluşur. Politika ağı (θ), mevcut durum geçişi için Q değerlerini tahmin etmekten sorumluyken, hedef ağ (θ^-) ardıl durumun optimal Q değerini hesaplar. DQN'de her bir Q değeri, Bellman denklemi ($\alpha = 1$) kullanılarak Denklem (2.23)'de gösterilen formül ile hesaplanır;

$$Q(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma . max_a Q(s_{t+1}, a) \tag{2.23}$$

Tekrar arabelleği (S. Zhang and Sutton 2018), sırasıyla, her eğitim adımı *t*'de tanık olunan geçiş demetlerini (s_t, a_t, s_{t+1}, r_t) tablo şeklinde depolar.

DQN'de ajan, çevre ile etkileşime girerek yeni durumları keşfeder. Keşif yöntemi olarak ε-açgözlü stratejisini kullanır. Tekrar arabelleğinde, yeni durumlar keşfedildikçe, geçiş demetleri depolanır. Hedef ve politika ağlarına iletilmek üzere, rastgele bir demet grubu (s_j, a_j, s_{j+1}, r_j) örneklenir. Ayrıca bir kayıp değeri ağların çıktı Q değerlerine göre Denklem (2.24)'deki gibi hesaplanır;

$$L(\theta) = \left\| y_t(a) - Q(s_j, a_j; \theta) \right\|^2 \equiv \left\| r_j + \gamma . max_a Q(s_{j+1}, a; \theta^- - Q(s_j, a_j; \theta) \right\|^2 \tag{2.24}$$

Denklem(2.23)'teki hedefler $y_t(a)$ ile $Q(s_j, a_j; \theta)$ tahminleri arasındaki hata daha sonra politika ağı parametreleri θ'yı ayarlamak için Denklem(2.25)'deki formül ile geri yayılır;

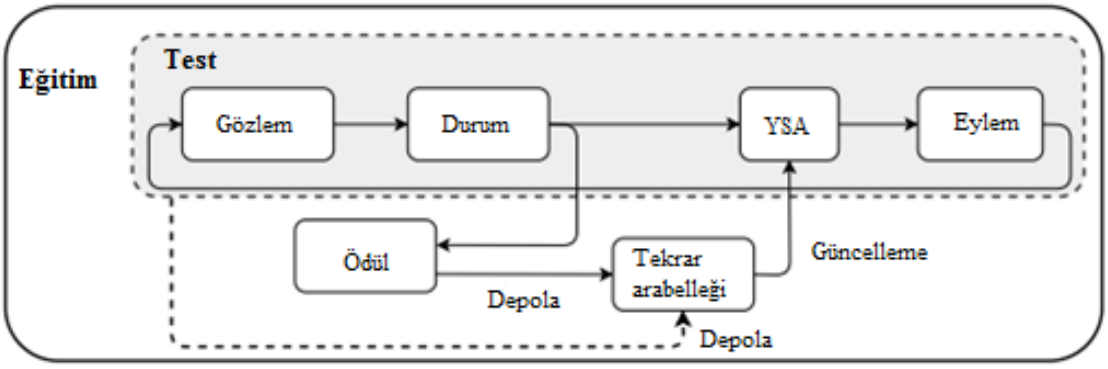
$$\Delta_\theta = \alpha . L(\theta) . \nabla_\theta Q(s_j, a_j; \theta) \quad (2.25)$$

Tekrar arabelleğinin kullanılması sayesinde, politika ağı güncellemelerinin ayarlamak için rastgele durum geçiş demetleri (s_j, a_j, s_{j+1}, r_j) kullanılır. Yani politika ağı (θ), yalnızca ortamın son gözlenen durumuna bağlı değildir. Bu durum ajanın uyguladığı yeni eylemler için eğitilmesini sağlar.

Her eğitim adımında ayarlanan θ politika ağı parametrelerinden farklı olarak, hedefin ağırlıkları ve önyargıları, θ^-politika ağı parametre değerlerini devralarak $\theta^- \leftarrow \theta$ periyodik olarak güncellenir. Bu işlem, hedef fonksiyonun hızla değişmesini önleyerek eğitim sürecinin sağlamlığını artırır.

Eğitim sürecinin ardından hedef ve politika ağı güncellenir. Böylece eğitilmiş bir ağ modeli (θ^*) ortaya çıkar. Bu ağ modeli ile ajan, karar verme süreçlerini kendi kendine öğrenmeye başlar. Bu nedenle, DQN ajan artık tekrar arabelleğine veya hedef ağa ihtiyaç duymaz, önemli bir hesaplama maliyeti düşüşü sağlar. Bu test aşamasında, sistemin işleyişi Şekil 2.19'da gösterildiği gibi oldukça basittir. Kullanılan ağ (θ^*), her t adımında, girdi durumunu s_t işler ve buna göre Q değerlerini $Q(s_t, a; \theta^*)$ hesaplar. Ajan, en yüksek ödülü getirecek bir eylem seçer. Denklem (2.26)'da gösterildiği gibi formüle edilir;

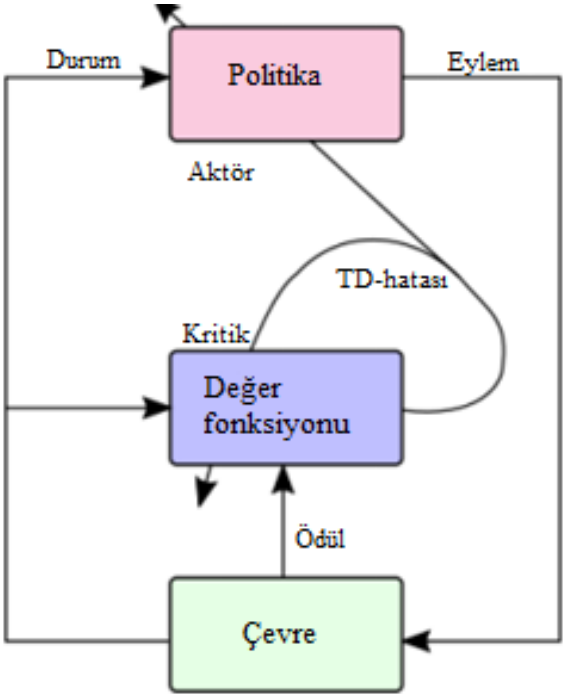
$$a_t \leftarrow argmax_{a \in Q(s,:;\theta^*)}(Q(s_t, a; \theta^*)) \equiv Q^*(s_t, a; \theta^*) \quad (2.26)$$



Şekil 2.19. Derin Q-Öğrenme Akış Diyagramı

2.5.2. Deterministik Politika Gradyanı

Deterministik Politika Gradyanı'nın (DPG) birçok çeşidi vardır. Bu tezde, β'nın indirgenmiş durum dağılımı olduğu ve β mevcut politika π ayrı bir politikayı temsil ettiği politika dışı deterministik Aktör-Kritik dikkate alınacaktır. Konun detayı için David Silver ve arkadaşlarının (Silver ve diğ., 2014) çalışmasına bakılabilir. Algoritmanın çalışma yapısı basitçe Şekil 2.20'de gösterilmektedir.



Şekil 2.20. Aktör-Kritik algoritmasının basit gösterimi

Aktör-Kritik mimarisi, beklenen getiriyi optimize etmek için iki yapı kullanır. Aktör ve Kritik birlikte çalışır ve algoritmalarda amaçlarına göre ayrı ayrı eğitilirler. Aktör, mevcut politikayı tanımlar ve bu nedenle mevcut politikaya göre eylemler üretmesi amaçlanır. Kritiğin görevi, problemin değer fonksiyonunu tahmin etmektir. Öğrenme genellikle politika üzerinedir ve Kritik, Aktör tarafından tanımlanan mevcut politikadan yürütülen beklenen eylem değerinin ne olduğunu öğrenmelidir. Kritik daha sonra politika tarafından gerçekleştirilen eylemi bir Zamansal Fark-Hatası (TD-Hatası) olarak eleştirebilir. TD-Hatası, iki farklı durumun değer fonksiyonu tahminleri arasındaki zamansal farktır. Değerlendirme matematiksel olarak Denklem (2.27)'deki gösterilen şekilde tanımlanır;

$$\delta_t = r_{t+1} + \gamma V(s_t) - V(s_{t+1}). \tag{2.27}$$

TD hatası daha sonra Aktör ve Kritik modelindeki parametreleri optimize etmek için kullanılır. $\delta_t > 0$ ise, mevcut eylemin sonucu a_t beklenenden daha iyidir ve bu nedenle $\pi(a_t|s_t)$ olasılığını artırmak arzu edilir.

DPG'ye geri dönersek performans hedefi, değer fonksiyonundan veya eylem değer fonksiyonundan Denklem (2.28)'deki gibi ifade edilebilir;

$$J_\beta(\mu_\theta) = \int_s p^\beta(s) V^\mu(s) ds$$
$$= \int_s p^\beta(s) Q_\theta^\mu(s)) ds. \tag{2.28}$$

Aktörler modelinin parametreleri için gradyanlar daha sonra Denklem (2.29)'da gösterilen şekilde tahmin edilebilir;

$$\nabla_\theta J_\beta(\mu) \approx \int_S p^\beta(s) \nabla_{\theta\mu\theta}(a|s) Q^\mu(s,a) ds$$
$$= \mathbb{E}_{s\sim p^\beta}\left[\nabla_{\theta\mu\theta}(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]. \tag{2.29}$$

Gerçek eylem değeri fonksiyonu, bir genel fonksiyon yaklaştırıcı $Q^w \approx Q^\mu$ ile değiştirilir ve gerçek eylem-değer fonksiyonunu en aza indirmek için eğitilir. Ayrıca, algoritmanın temel adımlarında kullanılan denklemler (Denklem (2.30), Denklem (2.31), Denklem (2.32)) şunlardır;

TD hatasını hesaplamak için:

$$\delta_t = r_t + \gamma Q^\omega\left(s_{t+1}, \mu_\theta(s_{t+1})\right) - Q^\omega(s_t, a^t), \tag{2.30}$$

Kritik ağırlığını hesaplamak için:

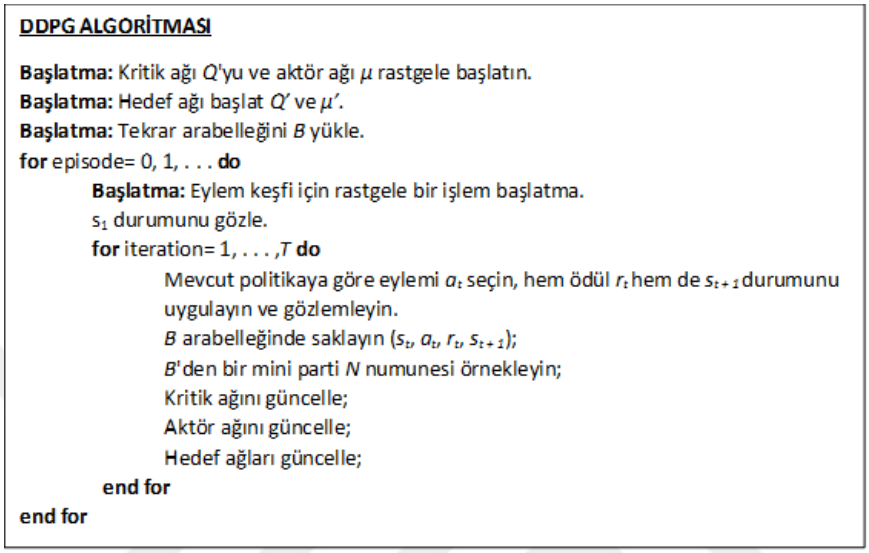
$$\omega_{t+1} = \omega_t + \alpha_\omega \delta_t \nabla_\omega Q^\omega(s_t, a^t), \tag{2.31}$$

Aktör ağırlığını hesaplamak için:

$$Q_{t+1} = Q_t + \alpha_\theta \nabla_{\theta\mu\theta}(s_t) \nabla_a Q^\omega(s_t, a^t)|_{a=\mu_\theta(s).} \tag{2.32}$$

2.5.3. DDPG

Derin Deterministik Politika Gradyanı(DDPG) (Lillicrap ve diğ., 2016), sinir ağlarını genel fonksiyon yaklaştırıcı olarak kullanarak DPG algoritmasını uygular. Şekil 2.21'de DDPG algoritması verildi.



Şekil 2.21.DDPG algoritması

Sinir Ağlarını sürekli eylem alanlarıyla PÖ'de uygulamak üç ana soruna yol açar. Bunlar; ilişkili veriler, istikrarsızlık ve yetersiz keşif sorunlarıdır. Bu kısımda, bu sorunları çözen üç yöntem açıklanacaktır. Eğitim Sinir Ağları, eğitim verilerinin bağımsız ve aynı şekilde dağıtılmasını gerektirir (Goodfellow ve diğ., 2016), ancak ortamda sıralı olarak örnekler üretilirken durum böyle değildir. DDPG algoritması, önceki deneyimi depolamak için tekrar arabelleğini kullanır. Yeterli miktarda veri toplandığında tekrar arabelleği kullanılabilir. Amaç, "ilişkili verilerin lanetine" karşı koymaktır. Aktör-Kritik için kayıp işlevi daha sonraki örneklerden alınan kare kaybı olarak Denklem (2.33) ve (2.34)'deki gibi formüle edilebilir;

$$L(\theta^Q) = \mathbb{E}_{s\sim p^\beta, a_t\sim\beta, r_t\sim E}[(Q(s_t, a_t|\theta^Q) - y_t)^2] \tag{2.33}$$

Buradan;

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q). \tag{2.34}$$

Kritik için Sinir Ağı, optimize ettiğimiz ağ ile aynı ağ ile hesaplandığından, sapmaya eğilimlidir. Bu sorunun çözümü, ağların kopyalarını oluşturmak ve daha sonra güncellemektir. Bu, istikrar sağlamak için hem Aktör hem de Kritik kopyalarını almanın en verimli yöntemdir. Kopyalar Denklem (2.35)'deki gibi gösterilir;

$$Q'(s,a|\theta^{Q'}) \qquad (2.35)$$

$$\mu'(s|\theta^{\mu'})$$

ve güncellemeler matematiksel olarak Denklem(2.36)'de gösterilen şekilde formüle edilebilir;

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \qquad (2.36)$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

$\tau \ll 1$'dir.

Sürekli eylem alanlarının araştırılması, var olan sonsuz sayıda permütasyon nedeniyle zordur. Politika dışı algoritmalarda, keşif öğrenme algoritmasından bağımsız olarak oluşturulabilir. Keşifçi bir Aktör oluşturmanın en basit yolu, Aktörler eylemine bir keşif gürültüsü eklemektir. Denklem (2.37);

$$v_{exp}(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N} \qquad (2.37)$$

Burada N, çevreye uyacak şekilde seçilebilir.

2.5.4. A2C

A2C ve A3C gibi aktör-eleştiri algoritmaları, model içermeyen, çevrimiçi, politika üzerinde pekiştirmeli öğrenme yöntemini kullanan aktör-kritik (AC) ajanlardır. Bu ajanın amacı, politikayı (aktörü) doğrudan optimize etmek ve getiriyi veya gelecekteki ödülleri tahmin etmek için bir eleştirmeni(kritik) eğitmektir (Mnih ve diğ., 2016).

A2C Asynchronous Advantage Actor Critic'in (A3C) senkronize, deterministik bir çeşididir. Yeniden oynatma tamponunun kullanılmasını önlemek için birden fazla işçi kullanır.

Şekil 2.22'de, A2C algoritmasının temel adımları gösterilmektedir.

A2C ALGORİTMASI

1. Ortamı başlat
2. Aktör ve eleştirmen ağlarını başlat
3. İterasyonları başlat
   4. Aktör adımlarını başlat
      5. Durumu başlat ve eylem seç
      6. Ortamda eylemi uygula ve yeni durumu ve ödülü al
      7. Belleğe durumu, eylemi ve ödülü kaydet
      8. Durumu yeni durumla güncelle
   9. Aktör adımlarını sonlandır
   10. Avantajları ve indirgenmiş ödülleri hesapla
   11. Eleştirmen ağını güncelle
   12. Aktör ağını güncelle
   13. İterasyonları kontrol et
14. Ortamı kapat

Şekil 2.22.A2C algoritması

İlk olarak, ortam başlatılır ve aktör ve eleştirmen ağları başlatılır. Ardından, belirli bir sayıda iterasyon yapılır. Her bir iterasyonda, aktör adımları gerçekleştirilir. Aktör, bir durum alır, bir eylem seçer, bu eylemi ortamda uygular ve yeni durumu ve ödülü alır. Bu bilgiler belleğe kaydedilir ve durum güncellenir. Daha sonra, avantajlar ve indirgenmiş ödüller hesaplanır. Eleştirmen ağı bu hesaplamaları kullanarak güncellenir. Aktör ağı da aynı hesaplamaları kullanarak güncellenir. Bu adımlar hem eleştirmen ağının hem de aktör ağının öğrenmesini sağlar. Son olarak, belirli bir sayıda iterasyon gerçekleştirildikten sonra işlem tamamlanır ve ortam kapatılır.

2.5.5. TRPO

Güven Bölgesi Politika Optimizasyonu (TRPO), model içermeyen, çevrimiçi, politika üzerinde, politika gradyan pekiştirmeli öğrenme algoritmasıdır. TRPO, çevresel etkileşim yoluyla veri örnekleme ve kısıtlı bir optimizasyon problemini çözerek politika parametrelerini güncelleme arasında geçiş yapar. Eski politika ile yeni politika arasındaki KL sapması, optimizasyon sırasında bir kısıtlama olarak kullanılır. Sonuç olarak bu algoritma, güncellenen politikayı mevcut politikaya yakın bir güven bölgesi içinde tutarak standart politika gradyan yöntemlerine kıyasla önemli performans düşüşlerini önler (Schulman ve diğ., 2015).

Şekil 2.23'de gösterilen sözde koda göre, başlangıçta bir politika fonksiyonu parametresi (θ) başlatılır. Daha sonra, döngü, yakınsamaya ulaşıldığında duracak şekilde devam eder. Her döngü adımında, mevcut politika kullanılarak veri (D) toplanır ve bu veriye dayanarak avantajlar (A) hesaplanır. Daha sonra, politika gradyanı (g) ve bir güven bölgesi alt-problemini çözmek için bir adım yönü (d) hesaplanır. Son olarak, bir adım büyüklüğü (α) belirlenerek politika parametreleri güncellenir.

TRPO ALGORİTMASI

Politika θ'yı başlat

Yakınsamaya ulaşılıncaya kadar:
  Mevcut politika kullanarak veri D topla

  Her bir durum-aksiyon çifti için avantajları A hesapla

  θ'ya göre beklenen dönüşüm gradyanını g hesapla

  Bir güven bölgesi alt-problemini çözmek için adım yönünü d hesapla

  Bir çizgi araması veya başka bir yöntem kullanarak adım büyüklüğünü α hesapla

  Politika parametrelerini güncelle: θ = θ + α * d

Şekil 2.23.TRPO algoritması

2.5.6. PPO

Proksimal Politika Optimizasyonu (PPO), politika iyileştirmesini garanti altına almak için, TRPO(Schulman ve diğ., 2015), yeni politikanın bir optimizasyon kısıtlaması olarak eski politikanın ortalama performansından daha iyi olup olmadığını ölçmek için *KL* farklılığını getirmiştir. *KL* ayrışma kısıtlaması ile politikanın monoton olarak iyileştirilmesi garanti edilir. Ancak, TRPO'nun uygulanması zordur ve yürütülmesi için daha fazla hesaplama gerektirir.

PPO(Schulman ve diğ., 2017), sınırlı optimizasyondan hesaplamayı azaltan kırpılmış bir vekil amaç işlevi önerdi. TRPO'daki kayıp fonksiyonu Denklem 2.37'deki gibi verilir;

$$L(\theta) = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right] \tag{2.37}$$

Burada $\hat{\mathbb{E}}_t[\ldots]$sonlu bir örnek grubu üzerindeki ampirik ortalamayı gösterir, $\hat{A}_t$ avantaj fonksiyonu tahmincisidir $\hat{A}_t \coloneqq -V(s_t + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-r} V(s_T)$ ve $r_t(\theta)$, geçerli politikası ile eski politika$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ arasındaki olasılık oranını gösterir.

KL ayrışma kısıtlaması, eski politikadan yeni politikaya ciddi bir güncelleme yapılmasını yasaklar, PPO böyle büyük bir değişiklikten kaçınmak için bir ceza uygular. Kırpılmış vekil amaç fonksiyonu Denklem 2.38'deki gibi verilir;

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right] \quad (2.38)$$

TRPO ile karşılaştırıldığında, olasılık oranı $r_t(\theta)$,[1 - ε, 1 + ε] arasında kırpılır, pratikte epsilon ε= 0.2 olarak seçilir, yani yeni politika ne kadar iyi olursa olsun, $r_t(\theta)$en fazla %20 artar.

$\hat{A}_t \geq 0$, geçerli eylemin belirli bir durumda diğerlerinden daha iyi performans gösterdiği anlamına gelir. Yeni politika eskisinden daha iyiyse, daha iyi olanın seçilme olasılığı daha yüksek olacak şekilde $r_t(\theta)$artırılmalıdır. Buna karşılık, $\hat{A}_t \leq 0$ için, eylem caydırılmalı ve $r_t(\theta)$azaltılmalıdır.

PPO algoritması, Şekil 2.24'de özetlenmiştir.

PPO ALGORİTMASI

for episode= 0, 1, . . . **do**
  for iteration= 1, . . . ,N **do**
    T zaman adımı için ortamda π_{old} politikasını çalıştırın;
    Avantaj tahminlerini hesapla $\hat{A}_1, ., ., \hat{A}_T$
  end for
  $L^{CLIP}(\theta)$ θ'ya göre, K dönemleri ve mini parti boyutu M ≤ NT ile optimize edin.
  Update $\theta_{old} \leftarrow \theta$
end for

Şekil 2.24.PPO algoritması

2.5.7. TD3

İkiz Gecikmeli Derin Deterministik Politika Gradyanı **(**Twin Delayed Deep Deterministic Policy Gradient) algoritması (TD3), hem aktörün hem de eleştirmenin fonksiyon yaklaşımcıları olarak kullanılan derin sinir ağlarından oluştuğu Aktör-Kritik mimarisine dayanmaktadır (Fujimoto ve diğ., 2018). TD3, DDPG algoritmasının üzerine inşa edilmiştir(Lillicrap ve diğ., 2016b). DDPG, robotik gibi sürekli kontrol problemleri için mükemmel sonuçlarla yaygın olarak kullanılmasına rağmen bazı sınırlamaları vardır. DDPG, diğer politika dışı algoritmalar gibi kararsız olma eğilimindedir ve Kritik Ağ'da Q değerinin aşırı tahmin edilmesine neden olan hiperparametrelere duyarlıdır. Bu hatalar zaman içinde biriktikçe, ajan yerel optimale düşebilir ve bu da optimal olmayan performansla sonuçlanabilir. TD3, aşağıdaki 3 optimizasyon görevini yerine getirerek aşırı tahmin sorununu ele almaktadır.

Kırpılmış Çift Q-öğrenme: TD3, Q değerlerini tahmin etmek için bir yerine iki (ikiz) kritik ağı kullanır ve hedefi oluşturmak için ikisinden daha küçük olanı kullanır. Bu nedenle kırpılmış Çift Q-öğrenme olarak da adlandırılan bu yaklaşım, Q değerlerinin düşük tahmin edilmesine neden olur ve bu da kararlı bir yaklaşım sağlar. Q değerlerinin aşırı tahmin edilmesinin aksine, düşük değerler yayılmadığından düşük tahmin daha az sorun teşkil eder.

Gecikmeli Politika Güncellemesi: Aktör-Kritik yöntemlerinde politika (aktör) ve kritik (değer) ağları birbirine sıkı sıkıya bağlıdır. Politika aşırı tahmin nedeniyle zayıf olduğunda eğitim ajanının değer tahmini sapar ve dolayısıyla politika yanlış değer tahmini nedeniyle daha da kötüleşmeye devam eder. Bu sorunu çözmek ve iki ağ arasındaki bağlantıyı azaltmak için, politika ağı değer ağından daha az sıklıkta güncellenir. Bu, politika ağının yalnızca birkaç yinelemeden sonra değer hatası azaldıktan sonra güncellenmesi nedeniyle algoritmanın genel kararlılığını artırır.

Hedef Politika Düzgünleştirme: TD3 gibi deterministik politika yöntemleri değer fonksiyonundaki ani artışlara aşırı uyum sağlayabilir. Bu, eleştirmen güncellenirken yüksek varyanslı hedeflerle sonuçlanır. Bu sorunu çözmek için hedefe az miktarda kırpılmış rastgele gürültü eklenerek ve mini gruplar üzerinde ortalama alınarak

düzenlileştirme veya yumuşatma kullanılır. Gürültü kırpma işlemi, hedef değerin orijinal eyleme yakın olmasını sağlamak için yapılır.

Şekil 2.25'de TD3 algoritmasının sözde kodu verilmiştir.

TD3 ALGORİTMASI

1. Başlangıçta, gözlem ve eylem boyutlarını ve hiperparametreleri ayarlayın.
2. Rastgele başlangıç değerleriyle eleman tablolarını (Q, Q_target, policy) ve gürültü tablolarını (noise, noise_target) başlatın.
3. Hedef ağı güncelleme sıklığını ve hedef ağı güncelleme ağırlığını (tau) belirleyin.
4. Deneyim tamponunu (replay buffer) başlatın.
5. Epizod döngüsü başlatın:
   6. Başlangıç durumunu alın ve başlangıç durumu işleyicisini sıfırlayın.
   7. Zaman adımları döngüsü başlatın:
      8. Hareketi ajanın politikasına (policy) göre seçin ve çevreyle etkileşime girin.
      9. Gözlem, ödül, sonraki durum ve terminal durumu kaydedin ve deneyim tamponuna ekleyin.
      10. Deneyim tamponunda yeterli sayıda örnekleme varsa:
         11. Rastgele bir deneyim örneği seçin.
         12. Q ağını güncellemek için eylem değerini hesaplayın ve gradient inişini gerçekleştirin.
         13. Hedef Q ağı güncellemek için gradient inişini gerçekleştirin.
         14. Politika ağını güncellemek için gradient inişini gerçekleştirin.
         15. Hedef ağı güncellemek için yumuşak hedef ağı güncellemesini gerçekleştirin.
      16. Durumunu güncelleyin.
   17. Epizodu tamamlamışsanız döngüyü sonlandırın.
18. Eğitimi tamamladıktan sonra, ajanın politikasını kullanarak çevreyle etkileşime girebilirsiniz.

Şekil 2.25.TD3 algoritması

2.5.8. SAC

Soft Aktör Kritik-Soft Actor Critic (SAC)(Haarnoja ve diğ., 2018), stokastik bir politikayı politika dışı bir şekilde optimize eden ve stokastik politika optimizasyonu ile DDPG tarzı yaklaşımlar arasında bir köprü oluşturan bir algoritmadır. TD3'ün doğrudan halefi değildir, ancak kırpılmış çift-Q hilesini içerir ve SAC'deki politikanın doğal stokastikliği nedeniyle, hedef politika yumuşatmadan yararlanır. Şekil 2.26'da SAC algoritmasının sözde kodu verilmiştir.

SAC'nin merkezi bir özelliği entropi düzenlemesidir. Politika, beklenen getiri ile politikadaki rastgeleliğin bir ölçüsü olan entropi arasındaki dengeyi maksimize edecek şekilde eğitilir. Bunun keşif-kullanım değiş tokuşu ile yakın bir bağlantısı vardır: artan entropi daha fazla keşifle sonuçlanır ve bu da daha sonra öğrenmeyi hızlandırabilir.

Ayrıca politikanın zamanından önce kötü bir yerel optimuma yakınsamasını da önleyebilir.

SAC ALGORİTMASI

1. Başlangıçta, gözlem ve eylem boyutlarını ve hiperparametreleri ayarlayın.
2. Rastgele başlangıç değerleriyle politika ağını (policy network), değer ağlarını (value networks) ve hedef değer ağlarını başlatın.
3. Deneyim tamponunu (replay buffer) başlatın.
4. Hedef ağı güncelleme sıklığını (target update frequency) ve güncelleme ağırlığını (target update weight) belirleyin.
5. Entropi katsayısını (entropy coefficient) ayarlayın.
6. Epizod döngüsü başlatın:
   7. Başlangıç durumunu alın ve başlangıç durumu işleyicisini sıfırlayın.
   8. Zaman adımları döngüsü başlatın:
      9. Eylemi politika ağından seçin ve çevreyle etkileşime girin.
      10. Gözlemi, ödülü, sonraki durumu ve terminal durumunu kaydedin ve deneyim tamponuna ekleyin.
      11. Deneyim tamponunda yeterli sayıda örnekleme varsa:
         12. Rastgele bir deneyim örneği seçin.
         13. Hedef değer ağlarını kullanarak hedef değeri hesaplayın.
         14. Değer ağlarını güncellemek için değer kaybını hesaplayın ve gradient inişini gerçekleştirin.
         15. Politika ağını güncellemek için politika kaybını hesaplayın ve gradient inişini gerçekleştirin.
         16. Hedef ağları güncellemek için yumuşak hedef ağı güncellemesini gerçekleştirin.
      17. Durumunu güncelleyin.
   18. Epizodu tamamladıysanız döngüyü sonlandırın.
19. Eğitimi tamamladıktan sonra, politika ağını kullanarak çevreyle etkileşime geçebilirsiniz.

Şekil 2.26.SAC algoritması

TD3'te olduğu gibi, her iki Q fonksiyonuda tek bir ortak hedefe geriletilerek öğrenilir.TD3'te olduğu gibi, paylaşılan hedef, hedef Q-ağları kullanılarak hesaplanır ve hedef Q-ağları, eğitim süresince Q-ağ parametrelerinin çoklu ortalaması alınarak elde edilir.TD3'te olduğu gibi, paylaşılan hedef kırpılmış çift-Q hilesini kullanır.

TD3'ten farklı olarak hedef, SAC'nin entropi düzenlemesi kullanımından gelen bir terim de içerir.TD3'ten farklı olarak, hedefte kullanılan bir sonraki durum eylemleri hedef politika yerine mevcut politikadan gelir.TD3'ün aksine, açık bir hedef politika yumuşatması yoktur. TD3 deterministik bir politika eğitir ve bu nedenle sonraki durum eylemlerine rastgele gürültü ekleyerek yumuşatmayı gerçekleştirir. SAC stokastik bir politika eğitir ve bu nedenle bu stokastiklikten kaynaklanan gürültü benzer bir etki elde etmek için yeterlidir (URL-2).

2.6. Benzetim Ortamları (Simülatörler)

Robotikte simülatörler, oluşturulan prototipleri kolayca ve ekonomik bir şeklide test etmek için kullanılır. (De Melo ve diğ., 2019). Simülatörler ile fizik motorları kullanılarak gerçek dünyaya benzer ortamlar oluşturulur. Robot deneylerinde en büyük risk meydana gelecek kazalardır. Simülatörler kullanılarak hem kazaların önüne geçilir hem de ciddi bir maliyet düşüşü sağlanır.

Tool	Currently used, and its the main tool	Currently used, but not the main tool	Currently used, just to test it	Used once, just to test it	Used then aban-doned	Known, but never used	Never heard of
Gazebo	**13%**	7%	3%	18%	10%	34%	**15%**
ODE	**11%**	12%	5%	18%	**22%**	22%	**10%**
Bullet	5%	13%	7%	12%	10%	29%	24%
V-Rep	5%	3%	3%	18%	3%	29%	39%
Webots	4%	7%	1%	16%	**13%**	32%	27%
OpenRave	5%	3%	2%	7%	5%	29%	49%
Robotran	4%	0%	1%	4%	2%	13%	76%
XDE	5%	3%	0%	3%	1%	14%	74%
Blender	5%	17%	7%	22%	6%	28%	**15%**
MuJoCo	2%	0%	0%	4%	2%	21%	71%
iCub_SIM	4%	4%	2%	3%	3%	29%	55%
Nvidia PhysX	1%	1%	4%	12%	7%	43%	32%
OpenSIM	3%	4%	3%	8%	1%	41%	40%
HumanS	0%	0%	0%	1%	1%	10%	**88%**
Moby	2%	1%	0%	0%	2%	14%	81%
Vortex	3%	2%	0%	5%	5%	17%	68%
RoboRobo	3%	1%	0%	0%	1%	4%	**91%**

Şekil 2.27. Simülatörlerin bilinirlik ve kullanılma durumları (Ivaldi ve diğ., 2014)

Çağdaş robotik simülatörleri, çeşitli fizik motorları, geniş bir robot, sensör ve aktüatör kütüphanesi, gelişmiş programlama ve grafiksel arayüzler, robot hareketi ve sensör okumalarının simülasyonlarını sağlayan çoklu eklentiler sunar (Ivaldi ve diğ., 2014).

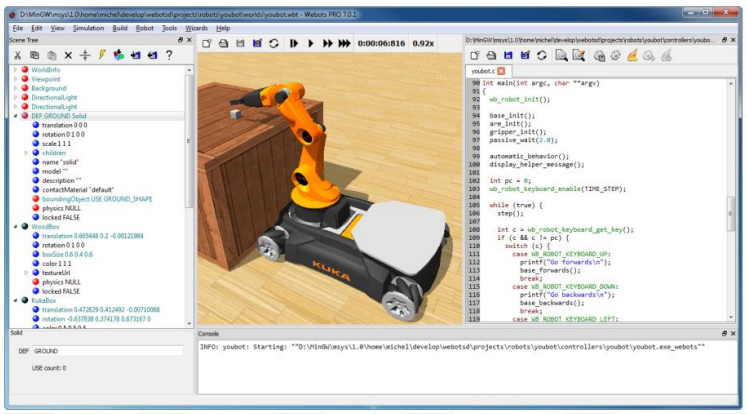
Simülatörler, kullanıcının robotlarla ve diğer bileşenlerle fiziksel olarak etkileşime girmek zorunda kalmadan test etmesine ve davranışlar oluşturmasına olanak tanıyan programlardır. Yazılıma ve robota bağlı olarak, simülatörde oluşturulan uygulamaların, örneğin Robot İşletim Sistemi (ROS) (Quigley ve diğ., 2009) aracılığıyla fiziksel robota aktarılmasına izin verebilir.

Bir simülatör kullanmanın faydaları, maliyetleri düşürmesi, zamandan ve paradan tasarruf etmesi, çeşitli alternatiflerin hiçbir maliyet veya risk ve arıza süresi olmadan test

edilmesine izin vermesi bakımından çok büyüktür. Şekil 2.27'de simülatörlerin bilinip bilinmediği ile ilgili yapılan bir anketin sonuçları verilmiştir.

2.6.1. Webots

Webots (Michel, 2004) yazılımı 1998 yılında Dr. Olivier Michael tarafından oluşturulmuştur ve çoğunlukla eğitim amaçlı kullanılmaktadır. Programa robotlar, sensörler ve aktüatörlerden oluşan geniş bir koleksiyon dahildir. Bir Webots sahnesi örneği, Şekil 2.28'de görülebilir.



Şekil 2.28. Örnek webots sahnesi

Robot hareket hızı üzerine yapılan araştırmalardan uyarlanabilir davranışların simülasyonundan öğretim ve robot programlama yarışmalarına kadar birçok alanda en sık kullanılan simülasyon yazılımlarından biridir.

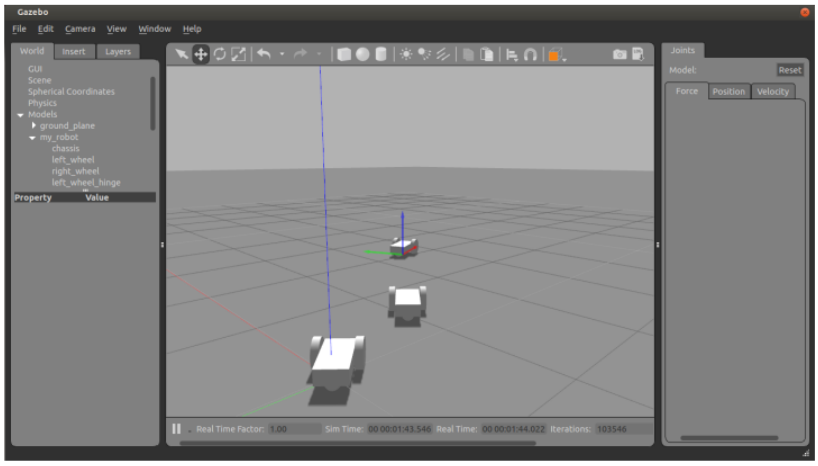
Webots çapraz platformdur ve C/C++, Java, Python ve Matlab gibi dilleri destekler. Oluşturma, OGRE motoru kullanılarak yapılır ve fizik motoru olarak ODE'nin özel bir sürümünü kullanır. Ayrıca, ROS'u desteklemenin yanı sıra dahili bir 3B modelleyici içerir (URL-3).

2.6.2. Gazebo

Gazebo (Koenig ve Howard, 2004), Open Source Robotics Foundation tarafından geliştirilen, 2002 yılında piyasaya sürülen bir robotik simülatördür. Karmaşık ortamlarda birden fazla robotu, nesneyi ve sensörü simüle etme yeteneğine sahip, dinamikleri olan çok robotlu bir simülatördür. Ayrıca, katı cisim fiziğini simüle etmenin yanı sıra nesneler arasında gerçekçi sensör geri bildirimi ve etkileşimler üretebilir.

Grafikler OGRE motoru kullanılarak oluşturulur. Dinamik simülasyonlar, dahil edilen dört fizik motorundan biri kullanılarak yapılabilir; ODE, Bullet, Simbody ve DART. Ana programlama dili C++'dır ve eklentiler kendi API'si kullanılarak geliştirilebilir. Simülasyonlar, TCP/IP kullanan uzak sunucularda veya bir bulut üzerinde çalıştırılabilir(Gong ve diğ., 2011).

Gazebo açık kaynak kodludur, tüm platformlar için mevcuttur ve çevrimiçi simülasyon modeli deposu, forum, wiki ve robot uygulamaları için kitaplık içeren çok aktif bir topluluğa sahiptir. Aynı şirket, robot yazılımı yazmak için bir çerçeve olan ROS'u da geliştirdi. Gazebo'daki bir mobil robot örneği, Şekil 2.29'da görülebilir.



Şekil 2.29. Örnek Gazebo sahnesi

2.6.3. V-rep (Copeliasim)

V-REP (Sanal Robot Deney Platformu), Marc Freese tarafından oluşturuldu ve ilk olarak 2010'da piyasaya sürüldü ve onu mevcut en modern simülatörlerden biri haline getirdi (URL-4). Hızlı prototipleme, otomasyon sistemlerinin simülasyonu ve öğretimi gibi birçok uygulama için kullanılabilir.

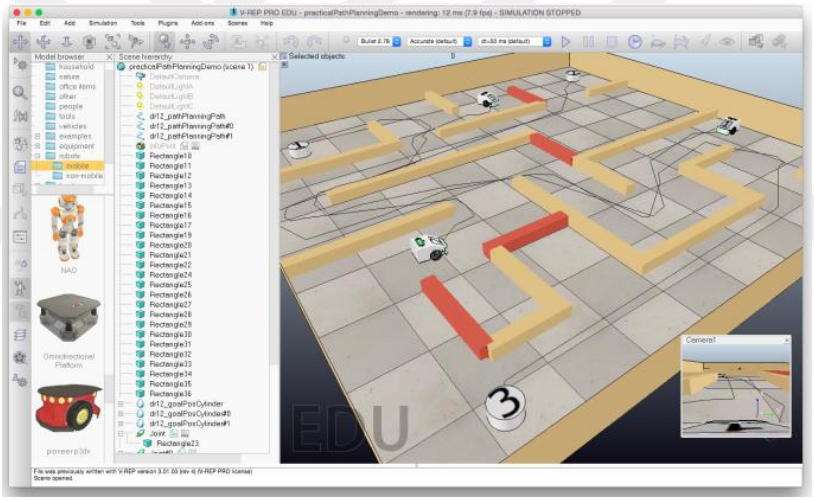
V-REP, Coppelia Robotics tarafından sağlanan genel amaçlı bir robot simülasyon çerçevesidir. Birçok özelliğinden bazıları şunlardır:

- Platformlar arası içerik (Linux, Mac ve Windows).

- Çerçeve ile çeşitli iletişim araçları (gömülü Lua komut dosyaları, C++ eklentileri, 6 dilde uzak API'ler, ROS vb.).
- Bir motordan diğerine hızlı bir şekilde geçiş yapabilme özelliğine sahip 4 fizik motoru (Bullet, ODE, Newton ve Vortex) desteği.
- Ters ve ileri kinematik.
- Hareket planlaması.
- Gömülü Lua komut dosyalarına dayalı dağıtılmış kontrol mimarisi.

Programa dahil edilen, piyasada bulunan robotların ve sensörlerin yanı sıra yeni modelleri içe aktarma veya entegre modelleme yeteneklerini kullanarak oluşturma yeteneğine sahiptir. ROS kullanarak gerçek robotlara da bağlanabilir.

Mobil robotların yer aldığı bir sahne Şekil 2.30'da görülebilir.



Şekil 2.30. Örnek Copeliasim sahnesi

2.6.4. Microsoft Robotics Developer Studio

Microsoft Robotics Developer Studio(Kang ve diğ., 2011) robot kontrolü için 3B simüle edilmiş ortam, sensör ve aktüatör verilerine kolay erişim, görsel bir programlama aracı ve web tabanlı arayüzler içerir. Araç, bir apartman, fabrika, ev ve dış mekan sahneleri dahil olmak üzere çeşitli simüle edilmiş ortamlarla birlikte gelir. Bir Microsoft aracı olan programlama, Python ve R ile karşılaştırıldığında genellikle makine öğrenimi ve pekiştirmeli öğrenme için popüler olmayan C# ile yapılır. Eylül 2014 itibariyle, Microsoft'un yeniden yapılandırma planının ardından araca verilen destek askıya alındı.

2.6.5. Robologix

Öncelikle öğretim için kullanılan Robologix (URL-5), programcıların kendi hareket dizilerini yazmalarını, ortamı değiştirmelerini ve mevcut sensörleri beş eksenli bir endüstriyel robotta kullanmalarını sağlar.

2.6.6. AnyKode Marilou

AnyKode Marilou (URL-6) mobil robotlar, insansı robotlar, eklemli kollar ve gerçek dünya koşullarında çalışan paralel robotlar için ortamları simüle eder. Sensörlerin ve aktüatörlerin fiziksel ortamdaki gerçek özelliklere göre davranışlarını son derece yüksek bir gerçeklik seviyesiyle yeniden üreten bir motor sunar. C/C++, VB, J# ve C# eklentileri sunar.

2.6.7. Graspit!

Graspit! (Miller ve Allen, 2004) kavramayı araştırmak için tasarlanmış bir araçtır. Bir dizi analiz ve geliştirme aracı eşliğinde robotik kavrama görevlerini simüle etmek için sanal bir ortamdır. Adından da anlaşılacağı gibi, bir uç manipülasyon görevinden ziyade kavramaya daha fazla önem verir ve mevcut modellerin seçimi Mico Arm gibi belirli robotlara odaklanır.

2.6.8. MuJoCo

MuJoCo (Todorov ve diğ., 2012) , hızlı ve doğru simülasyona ihtiyaç duyulan alanlarda araştırma ve geliştirmeyi kolaylaştırmayı amaçlayan bir fizik motorudur. Model tabanlı optimizasyon ve bağlantılar aracılığıyla optimizasyon için tasarlanmıştır.

2.6.9. OpenAI-Gym

OpenAI Gym (Brockman ve diğ., 2016), Makine Öğrenimi metodolojilerini doğrulamak için PÖ' nün epizodik tasarımına uygun olarak formüle edilen kıyaslama sorunlarının bir düzenlemesidir. OpenAI Gym platformu, araştırmacılara hazır ortamlar ve ajanlar sunarak PÖ algoritmları ile ilgili denemeler yapma imkanı sunar. Aşağıda kıyaslama ortamları hakkında bilgi verildi.

- PÖ literatüründe geçen küçük ölçekli görevler için ortamlar;
- Klasik Atari oyunlarda PÖ uygulamaları için ortamlar;

- Sürekli kontrol görevleri Box2D (URL-7) ortamı;
- MuJoCo fizik motoru kullanarak 2B ve 3B robot kontrolü(Todorov ve diğ., 2012) .

Bunları dışında daha birçok benzetim ortamı vardır. Otonom sürüş uygulamalarını test etmek için Carla ve Torcs, oyun geliştirme ortamı olan Unity, Python temelli robotik uygulamalar için Pybullet, insansız hava araçları için Airsim vb. gibi ortamlar mevcuttur.

Bu tezde, mevcut donanım kaynakları da göz önünde bulundurularak öncelikle iki boyutlu ortamda çalışmanın daha verimli olacağına karar verildi. Bundan dolayı ortam iki boyutlu bir grid dünya olarak matlab programında ve python minigrid kütüphanesinde ayrı ayrı tasarlanmıştır. Mobil robot ve ortam değişkenleri (engel, hedef, ödül, vs.) grid dünya üzerinde hücre olarak temsil edilmiştir. Pekiştirmeli öğrenme algoritmaları bu ortamda çalıştırılırmıştır. Böylece belli bir robot markasına bağlı kalmadan algoritmalara yoğunlaşıldı. Grid dünyada elde edilen tecrübeler (model, hiperparamtreler,vb.) 3 boyutlu dünyada kullanıldı.3 boyutlu robotik simülatörü olarak Gazebo platformu kullanıldı. Turtlebot3 robotu gazebo ortamlarında SAC, TD3, PPO algoritmaları ile çalıştırıldı.

2.7. Literatür İncelemesi

Pekiştirmeli öğrenme birçok farklı alana uygulanmaktadır. Türkiye'de son 10 yılda bu alanda yazılan tezler incelendiğinde finans, ulaşım ağları, trafik kontrolü, ağ saldırıları tespiti, enerji optimizasyonu, otonom araçlar, insansız hava araçları vb. gibi farklı alanlarda uygulamaları görülmüştür. Bu çalışmalardan yapılan çıkarımda bir ajanın etkileşimde bulunduğu çevrede dinamik, büyük, sürekli ve değişken veri akışı varsa pekiştirmeli öğrenme ile eğitilen ajanlar yeni durumlara kolayca uyum sağlayabilmekte ve en iyi kararı verebilmektedir. Bu tezde mobil robot navigasyonu için pekiştirmeli öğrenme çalışmaları incelenmiştir.

Khan, yaptığı çalışmada, mobil robot navigasyonu için TOSL informed-biased softmax regression (TOSL-iBSR) olarak adlandırılan yeni ve geliştirilmiş bir öğrenme süreci sunmuştur. Eylem seçimi rastgele bir süreç olarak değil, bunun yerine softmax regresyonu kullanılarak hesaplanan maksimum olasılık fonksiyonuna göre belirlenmiştir. Sunulan yaklaşımı kullanarak, robotun daha yüksek bir pozitif ödül ve daha az hesaplama maliyeti elde ederken navigasyon görevini tamamladığı gözlenmiştir. Simülasyon kullanılarak önerilen yaklaşımın Q-learning with softmax regression (Q-SR) ve true

online SARSA Q-biased softmax regression (TOSL-QBIASSR)'den daha iyi performans gösterdiği gösterilmiştir. Ajan olarak Pionner robot platformu kullanılmıştır. Python ve V-REP arasında çerçeve oluşturularak tüm fiziksel parametrelerle gerçek bir robot kullanılmıştır. Öğrenme sürecini daha bilinçli bir eylem seçme tekniğiyle birleştirerek bilinmeyen bir ortamda iki boyutlu gezinme gerçekleştirilmiştir(Khan, 2019).

Zhang ve diğerleri, kentsel arama ve kurtarma görevlerinde, mobil kurtarma robotlarının yürüteceği bir dizi yerel gezinme eylemini belirlemek için robotun yerleşik sensörlerinden gelen ham duyusal verileri kullanan bir DPÖ ağı geliştirmişlerdir. Optimum robot navigasyon eylemlerini belirlemek için girdi olarak derinlik görüntüleri, yükseklik haritaları ve 3B yönlendirmeyi kullanan Asynchronous Advantage Actor-Critic (A3C) mimarisine dayalı bir ağ eğitmişlerdir. Engebeli arazi bilinmediğinde DPÖ yaklaşımının bir ortamdaki bir robotu hedef konumuna başarılı bir şekilde yönlendirebileceğini göstermiştir(Zhang ve diğ., 2018).

Sung ve diğerleri, mobil robot navigasyonu için modelden bağımsız PÖ yaklaşımlarını tartışmışlardır. Ajan olarak Turtlebot3 robotu, fiziksel ortamı simule etmek için ise Gazebo ortamı kullanılmıştır. DQN algoritmasını farklı boyutlardaki gözlem uzaylarında denemişlerdir. Ayrık gözlem alanlarının sayısı artınca genelleme yeteneğinin kaybettiğini gözlemlemişlerdir. Öte yandan, ayrık gözlem alanları sayısını düşünce performansta yüksek varyanslara neden olunduğunu gözlemlemişlerdir. PÖ algoritmalarını ölçeklendirebilmek için birden fazla robotu simule etmişlerdir (Sung ve diğ., 2018).

Çetin, robot navigasyonu için Q-öğrenme algoritmalarının uygunluğunu incelemişlerdir. Bunun için 3 farklı labirent ortamında simülasyonlar yapmışlardır. Robotun labirentte hedefe gitmesini sağlayan algoritmanın başarısını; iterasyon sayısı, öğrenme katsayısı ve Q tablosunun matris boyutunun belirlediğini gözlemlemişlerdir. Q matrisinin kararlı bir yapıya erişmesini beklemeden iterasyon miktarı küçük tutularak doğru sonuçlar elde edebilmişlerdir(Çetin, 2014).

Muhammad ve Bucak, mobil robot navigasyonu için geleneksel Q-öğrenme algoritması yerine yeni bir algoritma önermişlerdir. Önerdikleri algoritmada gezinme sırasında, tüm durum-eylem çiftlerinin yörüngesi saklanır ve rafine edilmiş Q değerlerini herhangi bir durumdan bir hedef durumuna yaymak için geriye doğru bir yönde yeniden oynatılır.

Simülasyonlardan elde ettikleri sonuçlar ile geleneksel Q-Öğrenmeye kıyasla çok daha iyi bir performans gözlemlemişlerdir. Q-tablosunun yakınsama oranını büyük ölçüde azaltılmışlardır(Muhammad ve Bucak, 2013).

Güçkıran ve Bolat, yaptığı çalışmada TORCS ortamı için Soft Actor-Critic-LSTM (SAC-LSTM) ve Rainbow DQN algoritmalarını, keşif ve genelleme tekniklerini kullanarak en uygun DPÖ ajanlarını araştırmıştır. TORCS ortamında SAC-LSTM ve Rainbow algoritmalarını yarış araçlarına uygulamıştır. Otonom yarışlar simule edildikten sonra SAC-LSTM algoritmasının daha başarılı olduğunu gözlemlemiştir. Bunun sebebi olarak keşif yöntemleri ve sürekli eylem alanı nedeniyle olduğunu iddia etmiştir. SAC, entropiyi maksimize etmeye çalışır ve bu ajanın eylem alanının belirsiz bölgelerini keşfetmesine izin verir. Ek olarak, SAC'ın politika ağı sürekli eylemler içerdiğinden, Rainbow DQN'nin gizli 27 eyleminden farklı olarak frenleme, hızlanma ve yönlendirme sürekli eylemlerle kontrol edilebilir(Guckiran ve Bolat, 2019).

Altuntaş, popüler PÖ algoritmalarından Sarsa(λ) ve Q(λ) algoritmalarını seçerek mobil robot navigasyonu probleminin çözümü için bir sistem önermiştir. MATLAB ile geliştirilen sistem, hem simülasyon hem gerçek ortamda, yüksek bir başarı oranıyla gezgin robotu engellerden kaçırarak istenen hedefe yönlendirebilmiştir. Robot platformu olarak Robotino kullanılmıştır. Ayrıca, sistem sayesinde PÖ metotlarında kullanılan başlangıç parametrelerinin, örneğin λ, öğrenmeye olan etkisini gözlemlemiş ve Sarsa(λ) ve Q(λ) algoritmalarının performanslarında karşılaştırmalar yapmıştır. Sonuç olarak, SARSA'nın% 90 ila% 70 öğrenme oranıyla Q-öğrenmeden daha hızlı optimal değerlere yakınlaştığını bulmuştur. (Altuntaş, 2013).

Engin, Q-öğrenme, SARSA, DQN ve Bulanık Kural Interpolasyonu Temelli Q tipi öğrenme (FRIQ) gibi farklı pekiştirmeli öğrenme algoritmalarının, bir robot modelinin iki boyutlu bir labirent ortamında başlangıç noktasından hedefe ulaşana kadar geçen toplam zaman ve adım sayısına bağlı olarak performans kıyasını içeren bir çalışma yapmıştır. DQN ve FRIQ öğrenmenin, daha az sayıda bölümle hedefe ulaşmak için en kısa yol anlamına gelen optimal politikayı bulma açısından Q-öğrenme ve SARSA'ya üstün olduğu görülmüştür.

FRIQ-Öğrenmeye göre, DQN, daha az sayıda bölümle en uygun politikayı bulma açısından biraz daha iyi bir performansa sergilemiştir. Öte yandan, FRIQ öğrenme, labirentin duvarlarıyla çevrili köşelerde ve dar bölgelerde daha iyi bir performansa sahip olduğu görülmüştür.(Engin, 2019) .

Demir, otonom forkliftler gibi değişken yükler altında çalışacak olan robotların hareket planlama ve kontrolü problemlerine farklı bir çözüm önermiştir. Önerilen yöntem ile robotların üzerlerindeki yükler altında nasıl hareket edebildiklerini derin pekiştirmeli öğrenme yöntemi ile öğrenmeleri sağlanmıştır. Ardından robotlar daha önceden kendilerine öğretilmiş olan görevi, öğretim zamanı kendilerine verilmemiş yük miktarlarında da tekrarlamış ve başarılı olmuşlardır. Ajanlar DDPG algoritması kullanılarak eğitilmiştir(Demir, 2019).

Lei Tai ve Ming Liu tarafından yapılan çalışmada(Tai ve Liu, 2016), Kinect RGB-D kamerasından elde edilen ham derinlik görüntülerini işleyen bir CNN ile işletilen mobil platform, farklı senaryolarda çarpışmasız bir şekilde ortamı keşfetmeyi başarı ile öğrenmiştir. Aynı araştırmacılar bir başka çalışmada (Tai ve diğ., 2017) bir robot üzerine monte edilmiş tek bir SICK TiM570 lazer kullanarak haritasız bir hareket planlayıcı yaklaşımı önerdiler. On adet seyrek lazer bulgusu ve göreli hedef konumundan oluşan bir durum modeli ile mobil platform, herhangi bir engele çarpmadan istenilen hedeflere ulaşmayı başardı. Eğitim rutinini sürdürmek için kullanılan DPÖ yöntemi, eşzamansız DDPG tabanlı bir algoritmaydı(Lillicrap ve diğ., 2016).

Liang ve ark. yaptıkları çalışmada (Liang ve diğ., 2020), bir mobil robotun yoğun ve kalabalık ortamlarda gerçek zamanlı çarpışma önleme işlemini gerçekleştirmesini sağlayan bir uygulama sunmaktadır. Ajan, PPO (Schulman ve diğ., 2017) adlı politika tabanlı bir DPÖ algoritması kullanarak dinamik ve statik engellerle girdiği farklı etkileşim türlerinden dolaylı olarak öğrenir.

Xie ve ark. (Xie ve diğ., 2017) ve Ruan ve ark. (Ruan ve diğ., 2019), dinamik engellerden kaçınma ile uçtan uca bir mobil robot navigasyonu oluşturmak için son teknoloji D3QN mimarisini kullanarak karşılaştırılabilir çalışmalar önermektedir. Ruan ve ark., ilgili doğrulamaları gerçekleştirmek için hem simüle edilmiş hem de gerçek alanlarda Kinect

RGB-D kamera ile donatılmış bir platform kullanırken, Xie ve ark. bu tür sensörleri yalnızca sanal ortamlarda kullanmaktadır.

Chen ve ark. (Chen ve diğ., 2017) , sırayla, DPÖ değer tabanlı bir yaklaşım olan V-Öğrenme'yi kullanarak tamamen özerk bir robotik navigasyon sağlayan yöntem önermektedir. Kalabalık bir ortamda insan yürüme hızında hareket eden platform, sosyal olarak bilinçli bir hareket planlaması yürütürken diğer üç ajanı tespit etmeyi ve izlemeyi başarıyor. Yöntemin ödül modelini ayarlayarak, robot sağ veya sol sosyal normları benimseyebildiğini kanıtlıyor.

Yukarıda belirtilen ajanın tespit ve takip numarası sınırlamasıyla yüzleşmek için, Everett ve arkadaları (Everett ve diğ., 2018) rasgele sayıda ajanı gözlemlemek için yinelenen sinir ağları (RNN) mimarisine sahip bir çözüm önermiştir. Karar verme ajanları arasında sunulan DPÖ tabanlı hareket planlaması, politika tabanlı bir GPU / CPU Asenkron Advantage ActorCritic (GA3C) (Babaeizadeh ve diğ., 2017) öğrenme yaklaşımı, bir kuyruk sistemi kullanan bir strateji ve Derin Yapay Sinir Ağlarını eğitmek için dinamik bir zamanlama tekniği kullanmaktadır.

Bu tez çalışmasının amacı, mobil robotların pekiştirmeli öğrenme algoritmalarını kullanarak dinamik iç ortamlarda yörünge planlamasını yapmasını sağlamaktır. Mobil robotik araştırmalarındaki otonom navigasyon konusuna yıllar içerisinde pek çok çözüm bulunmuş ve geliştirilmiştir. Bu çalışmada bu zamana kadar mobil robotların otonom navigasyon probleminin çözümüne nasıl yaklaşıldığına ve pekiştirmeli öğrenme yönteminin seçilmesinin sebebine odaklanılacaktır. Pekiştirmeli öğrenme algoritmalarının değişen çevre şartlarına göre performans testleri yapılacaktır. Ayrıca öğrenme sürelerine göre de testler yapılıp algoritmalar kıyaslanacaktır.

3. MALZEME VE YÖNTEM

3.1. İşletim Sistemi

Ubuntu(Sobell, 2015), ücretsiz, açık kaynak kodlu, güvenlikli, geliştiricilere imkan sağlayan bir işletim sistemdir. Çoğu donanım ve yazılım sürümüyle uyumlu olması ve robot çerçeveleri, simülatörler ve Entegre Geliştirme Ortamları (IDE'ler) gibi çeşitli modülleri kolayca birleştirmek için gerekli araçları sunması nedeniyle robotik uygulamaları ve simülasyonu yapmak için geleneksel işletim sistemi haline gelmiştir.

Bu çalışmada, Gazebo platformunu ROS Noetic ile kullanmak için Ubuntu 20.04, sürümü kullanıldı.

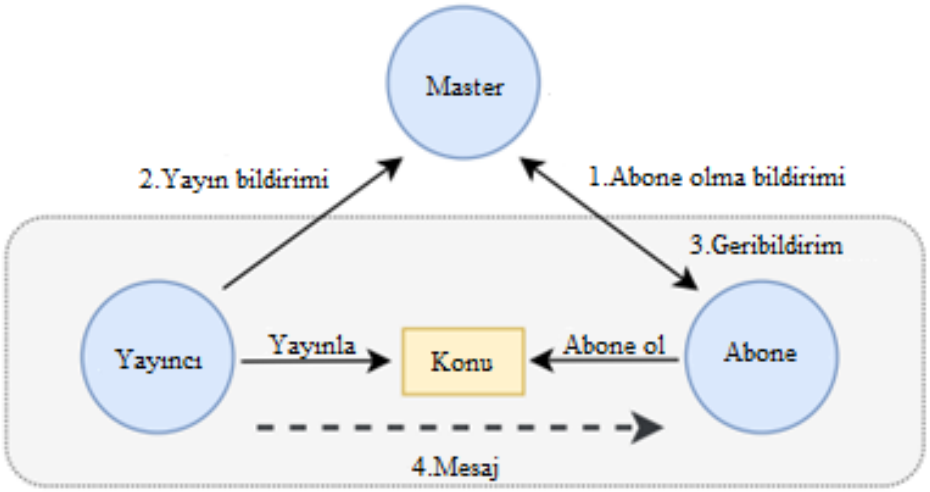
3.2. Robot İşletim Sistemi (ROS)

Robot İşletim Sistemi (Quigley et al. 2009), robotların ortak platformda çalışabilmesi için çeşitli protokoller, kütüphaneler ve araçlar sunan açık kaynak kodlu bir meta işletim sistemdir. C++, Python gibi çeşitli yazılım dillerini destekler. ROS yapısı dört ana etmenden meydana gelir. Bunlar; konular, hizmetler, düğümler ve mesajlardır. Düğümler ana yazılım birimleridir. Aralarında mesajları gönderir ve alırlar. ROS'da yayıncı-abone mantığı vardır. Bir düğüm bir konuya kaynağa olur, başka bir düğümde aboneye yayın yapar. Örneğin; Anlık nem bilgisini yayınlayan ve ona abone olan düğümler gibi.

- Düğümler arasındaki iletişimi konular ya da hizmetler üzerinden sağlanır.
- Konular: Yayıncı-Abone protokolüne göre çalışır. Mesajı yayınlayan bir düğüm vardır. Başka bir düğümde ona abone olur. Hizmetler: İstemci-Sunucu mantığı ile çalışır. Bir düğüm istek yapar diğer düğüm isteğe cevap vererek hizmet sunar.

Şekil 3.1'de ROS 'un çalışma yapısını gösterilmektedir. Her konunun yayıncılarını ve abonelerini, konu adreslerini, hizmetleri ve yayınlanan mesajları izleyen bir ana sunucu (Master) vardır. Kayıt ve iletişim işlemleri sırasıyla aşağıdaki adımlara uyar:

1. Abone düğümler Master'a bir konuya abone olmak isteklerini bildirir.
2. Yayıncı düğümler, Master'a aynı konuda yayın yaptığını bildirir;
3. Abone düğümler, Master düğüm tarafından geri bildirim alır;
4. Abone düğümler yayıncı düğümle iletişime geçer ve mesajı alırlar.



Şekil 3.1.ROS yayıncı-abone iletişimi

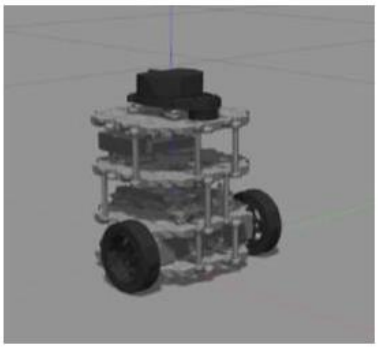
3.3. Gazebo

Robotik simülatörler, fizik motorları, sensör ve aktuatör destekleri ve çeşitli yazılım dili destekleri araştırmacılara güzel imkanlar sunar(Ivaldi ve diğ., 2014). Birkaç seçeneği (Gong ve diğ., 2011) analiz ettikten sonra, Gazebo (Zamora ve diğ., 2016), geliştirilen navigasyon çerçevesi deneylerine temel olarak gerçekleştirmek üzere seçilen robot simülatörüydü. Gazebo, mobil, insansı ve hizmet robotu araştırma alanlarında (Ivaldi ve diğ., 2014) en yaygın yazılımdır.Bu tezde üç boyutlu ortamlarda PPO,SAC,TD3 algoritmalarının eğitimi Gazebo'nun sunduğu hazır ortamlar ile yapıldı.

3.4. Turtlebot

Şimdiye kadar DPÖ tabanlı algoritmayı geliştirmek için, robot çerçeveleri, ortam görselleştirme araçları ve navigasyon simülatörleri açıklandı. Bununla birlikte, her türlü veriyi toplayan ve yazılım kontrol modüllerine ileten, tüm navigasyon akışının uyum içinde çalışmasını sağlayan sistemin merkezi unsuru, mobil robottur. Bu tezde Gazebo ortamında yapılan deneylerde Turtlebot kullanıldı. Şekil 3.2'de Turtlebot Burger gösterildi.

Turtlebot, kullanıcıların robotlar veya sanal ortamlar oluşturmaya ihtiyaç duymadan robotik uygulamaları hızlı bir şekilde geliştirmelerini sağlayan bir paket olan ROS 'da mevcuttur. Turtlebot'un ana donanım özellikleri (URL-8) Tablo 3.1'de sunulmaktadır ve yerleşik lazer özellikleri Tablo 3.2'de listelenmiştir.



Şekil 3.2. Turtlebot Burger

Tablo 3.1. Turtlebot donanım özellikleri

Maksimum hız	0.22 m/s
Maksimum dönüş hızı	2.84 rad/sn(167.72 derece/sn)
Maksimum hız	15 kg
Boyut(Uzunluk x Genişlik x Yükseklik)	138mm x 178mm x 192 mm
Lazer Mesafe Sensörü	360 Lazer Mesafe Sensörü LDS-01

Tablo 3.2.Lazer Mesafe Sensörü LDS-01

Mesafe Aralığı	120-3500mm
Mesafe doğruluğu(120mm-499mm)	±15mm
Mesafe doğruluğu(500mm-3500mm)	±5.0%
Mesafe hassasiyeti(120mm-499mm)	±15mm
Mesafe hassasiyeti (500mm-3500mm)	±3.5%
Tarama hızı	300±10 rpm
Açısal aralık	360°
Açısal çözünürlük	1°

3.5. Programlama Dili

Bu çalışmada ROS'da Gazebo ortamında çalışacak yazılım modüllerini geliştirmek için Python (URL-9) programlama dili seçildi. Sürüm olarak da Python 3.8 kullanıldı. Python, makine öğrenimi uygulamaları oluşturmak için tercih edilen bir programlama dilidir, geliştiricilere yardımcı olmak için çeşitli araçlara ve kapsamlı bir paket kitaplığına sahiptir. Bu tezde makine öğrenme kütüphanesi olarak PyTorch(Paszke ve diğ., 2019) ve TensorFlow (Abadi ve diğ., 2016) kullanıldı. Veri görselleştirme için TensorBoard, ROS bağlantısını kurmak için RosPy kullanıldı. Grid dünya oluşturmak için Minigrid kütüphanesi (URL-10) kullanıldı.

3.6. Matlab Pekiştirmeli Öğrenme Araç Kutusu

Pekiştirmeli Öğrenme Araç Kutusu (URL-11), DQN, PPO, SAC ve DDPG dahil olmak üzere pekiştirmeli öğrenme algoritmalarını kullanarak eğitim politikaları için bir uygulama, işlevler ve bir Simulink bloğu sağlar. Bu politikaları, kaynak tahsisi, robotik ve otonom sistemler gibi karmaşık uygulamalar için denetleyiciler ve karar verme algoritmaları uygulamak üzere kullanabilirsiniz.

Pekiştirmeli Öğrenme Araç Kutusu, derin sinir ağları veya arama tabloları kullanarak politikları ve değer işlevlerini temsil etmenize ve bunları MATLAB veya Simulink'te modellenen ortamlarla etkileşimler yoluyla eğitmenize olanak tanır. Araç kutusunda sağlanan tek veya çok ajanlı pekiştirmeli öğrenme algoritmalarını değerlendirebilir veya kendinizinkini geliştirebilirsiniz. Hiper parametre ayarlarıyla denemeler yapabilir, eğitim ilerlemesini izleyebilir ve eğitilmiş ajanları uygulama aracılığıyla etkileşimli olarak veya programlı olarak simüle edebilirsiniz. Eğitim performansını artırmak için simülasyonlar birden fazla CPU, GPU, bilgisayar kümesi ve bulutta (Paralel Bilgi İşlem Araç Kutusu ve MATLAB Paralel Sunucu ile) paralel olarak çalıştırılabilir.

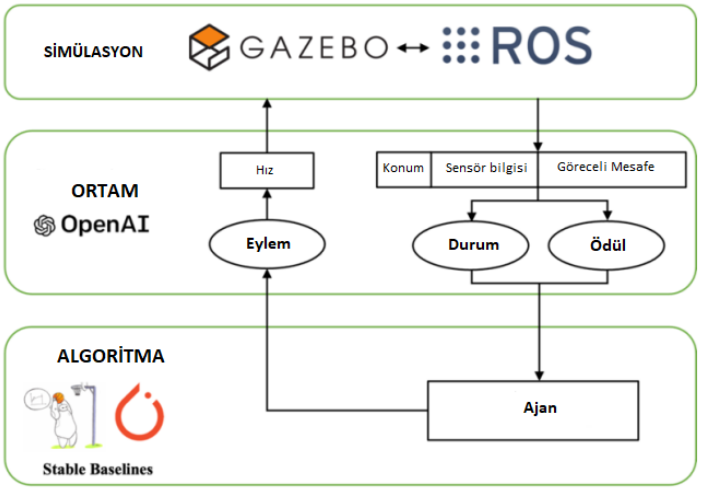
3.7. OpenAI Gym ve Stable Baselines

OpenAI gym(Brockman ve diğ., 2016b), pekiştirmeli öğrenme ortamlarının ve algoritmalarının geliştirilmesi ve kıyaslanması için yaygın olarak kullanılan bir python kütüphanesidir. Kullanım kolaylığı nedeniyle, PÖ öğrenmek için bir başlangıç noktası haline gelmiştir. Atari video oyunlarından, robotik hareket kontrolüne kadar çeşitli kullanıcılara sunar. Bu tezde kullandığımız özel ortamın oluşturulmasını destekler.

OpenAI gym, çevre ve öğrenme algoritmaları arasındaki iletişim için standart API'ler sağlar. DPÖ algoritmalarını kullanarak mobil robotu eğitmek için OpenAI gym ile birlikte stable baselines3 (SB3) kütüphanesini kullanıldı. SB3, PyTorch kullanarak son teknoloji DPÖ algoritmalarının yüksek kalitede uygulanmasını sağlayan açık kaynak kodlu bir çerçevedir (Raffin ve diğ., 2021).

Derin pekiştirmeli öğrenme kullanan mobil robot haritasız navigasyon sisteminin genel mimarisi, Şekil 3.3’de gösterilmektedir. Bu sistemin üç temel bileşeni vardır: Simülasyon, OpenAI Gym ortamı ve Stable Baselines temelli DPÖ algoritmalarının

uygulanması. Modüllerde kullanılan tüm araçlar, kütüphaneler ve çerçeveler açık kaynaklıdır.



Şekil 3.3. Gazebo-Ros-OpenAI-Stable Baselines Mimarisi

3.8. Derin Pekiştirmeli Öğrenme Temsilcileri

Bu tezde son teknoloji DPÖ algoritmaları olan DQN, SAC, A2C, TRPO, TD3, PPO karşılaştırıldı. Bu algoritmalar, DPÖ araştırma topluluğundaki mevcut literatüre ve popülerliğe göre seçilmiştir (Fujimoto ve diğ., 2018; Raffin ve diğ., 2021). A2C, DQN, TRPO, PPO gibi ayrık eylem uzayında çalışabilen farklı pekiştirmeli öğrenme algoritmalarının performansları grid dünya ortamında kıyaslanmıştır. Sürekli eylem uzaylarında çalışan TD3, SAC, PPO algoritmaları Gazebo ortamında kıyaslandı. DPÖ ajanlarına ait hiperparemetreler Bölüm 4'de verilmiştir.

3.9. Çevre-Ortam Temsilleri

Pekiştirmeli öğrenmede çevre, girdi/çıktı veri reaksiyonlarının, model görselleştirmesinin ve ödül işlevinin açıklamasıdır. Yani bir PÖ ajanı çevre ile etkileşime girerek çevreden ödül, durum, yeni durum bilgilerini alır.

Araştırmacılar, yol planlama problemini çözmek için kullanılan çeşitli yöntemleri çevre tipi ve yol planlama algoritmaları olarak iki faktöre göre ayırmaktadırlar. Çevre tipi, statik ve dinamik olmak üzere ikiye ayrılır. Statik ortam, robottan başka hareketli nesneler içermeyen ortam olarak tanımlanır; dinamik ortam ise dinamik hareketli nesnelere (yani

insanlar, hareketli makineler ve hareketli robotlar) sahip olan ortamdır(Al-Taharwa ve diğ., 2008).

Bu tezde, pekiştirmeli öğrenme ajanları önce 2 boyutlu ortamlarda daha sonra 3 boyutlu ortamlarda eğitildi. 3 boyutlu ortamda eğitim, donanım kaynaklarını fazlasıyla zorlamıştır. Bu nedenle kapalı bir oda ortamı öncelikle grid dünya temsil edildi. Böylece mevcut kaynaklar daha verimli kullanılmıştır. Eğitim süreleri arasındaki fark Bölüm 4'te verildi. 3 boyutlu ortamda algoritmaları eğitmek için Gazebo Empty World ve Gazebo Maze World ortamları kullanıldı. 2 boyutta ise Matlab'ta dinamik ve statik grid dünya, MiniGrid kütüphanesinde ise MiniGrid-FourRooms ve MiniGrid-Dynamic-Obstacles ortamları kullanıldı.

3.10. Ödül Modelleri

Ödül fonksiyonları veya ödül şekillendirme, geçerli politikayı ve optimizasyon hedefini dolaylı olarak belirlediğinden, geçerli bir politikanın başarılı bir şekilde öğrenilmesinde kritik bir rol oynar. Optimal bir ödül işlevi tasarlamak zorlu bir iştir (Abbeel ve Ng, 2004). Bir ödül fonksiyonu, her t adımında ödülü hesaplar. Bir ajana, istenen şekilde eylem oluşturma için pozitif bir ödül verilirken, aksi halde cezalandırılır. Bu nedenle, ödül işlevi, ajana navigasyon politikasını / eğitim modelini öğrenirken veya model devreye alındığında performansının iyileşip iyileşmediğini bildiren bir geri bildirim sinyali sağlar. Bu çalışma için farklı ödül fonksiyonlarını ve parametrelerini araştırıldı ve denendi. Bölüm 4'te her ortamda kullanılan ödül yaklaşımlarına yer verilmiştir.

3.11. Parametre Optimizasyonu

Bu tezde iki aşamalı bir yöntem izlenmiştir.2 boyutlu dünyada elde edilen tecrübeler(hiperparamterler,modeller,vb.) 3 boyutlu ortamlara aktarıldı. Parametre optimizasyonu için RL Baselines 3 Zoo (URL-12) kütüphanesinden yararlanıldı. RL Baselines3 Zoo, PyTorch'taki pekiştirmeli öğrenme algoritmalarının güvenilir uygulamaları olan Stable Baselines3'ü kullanan PÖ için bir eğitim çerçevesidir.

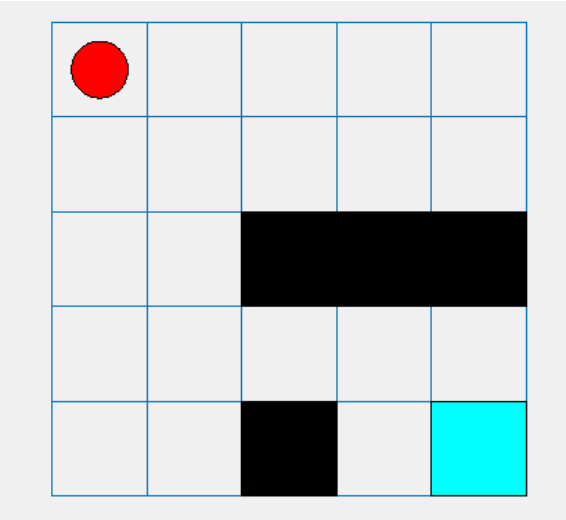
3.12. Donanım

Bu tezde çalışmalar, işlemci olarak, Intel(R) Core(TM)2 Duo CPU P8400 @ 2.26GHz, 2267 Mhz, 2 Çekirdek, 2 Mantıksal İşlemci, Yüklü Fiziksel Bellek (RAM) 4.00 GB bir

bilgisayarda yapılmıştır. Mevcut donanım kaynakları ile 3 boyutlu render işlemi çok uzun süre almıştır ve kaynakları ciddi anlamda zorlamıştır.

4. BULGULAR VE TARTIŞMA

4.1. Grid Dünya



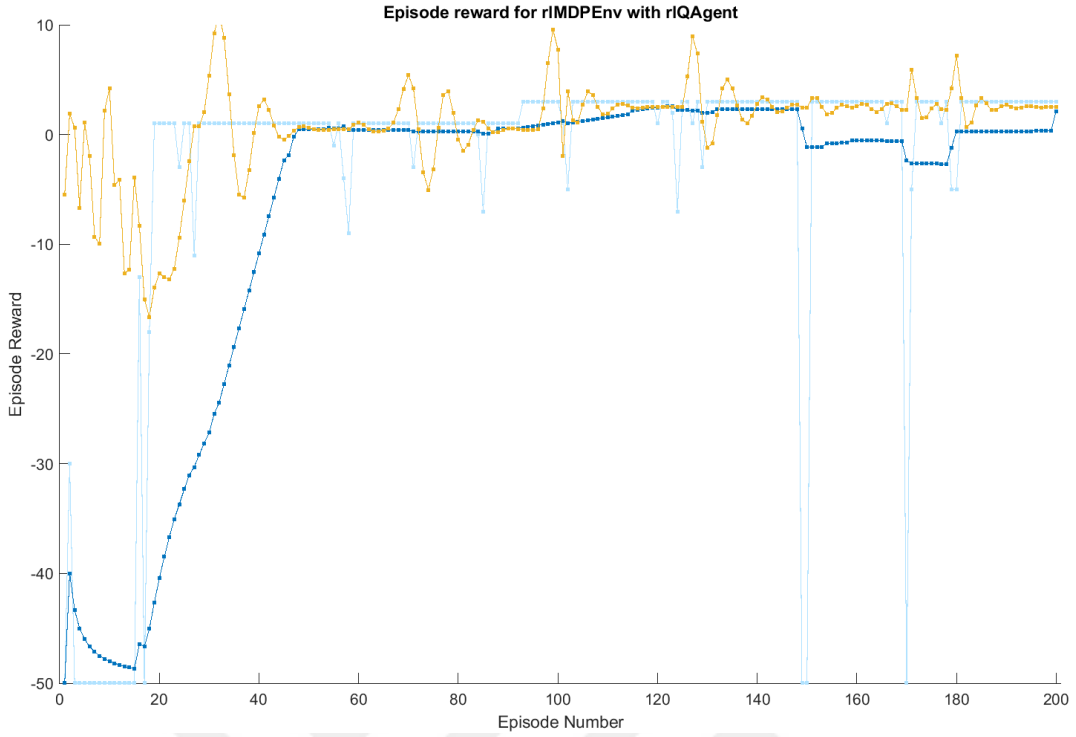
Şekil 4.1. 5x5 grid dünya

Şekil 4.1'de kapalı bir oda ortamının grid dünya temsili gösterilmektedir. 5x5 matris yapısındadır. Her hücre bir metrekare olarak varsayılmıştır. Böylece 25 metrekarelik bir oda oluşturuldu. [[3,3],[3,4],[3,5],[5,3]] noktaları duvar olarak tasarlandı. [5,5] noktası varılacak hedef nokta olarak tasarlandı. [1,1] noktası robotun başlangıç noktasıdır. Durum uzayı 25x1, eylem uzayı 4x1 vektörlerden oluşur. Yukarı, aşağı, sağa ve sola olmak üzere 4 eylem vardır. Ödül modeli olarak ise robotun hedefe ulaşamadığı her an -1 ödül puanı alır. Hedefe ulaşınca ise 10 puan alır. Q ajanı ve Sarsa ajanı bu ortamda eğitildi. Kullanılan hiperparametreler ve eğitim seçenekleri Tablo 4.1 'de verilmiştir.

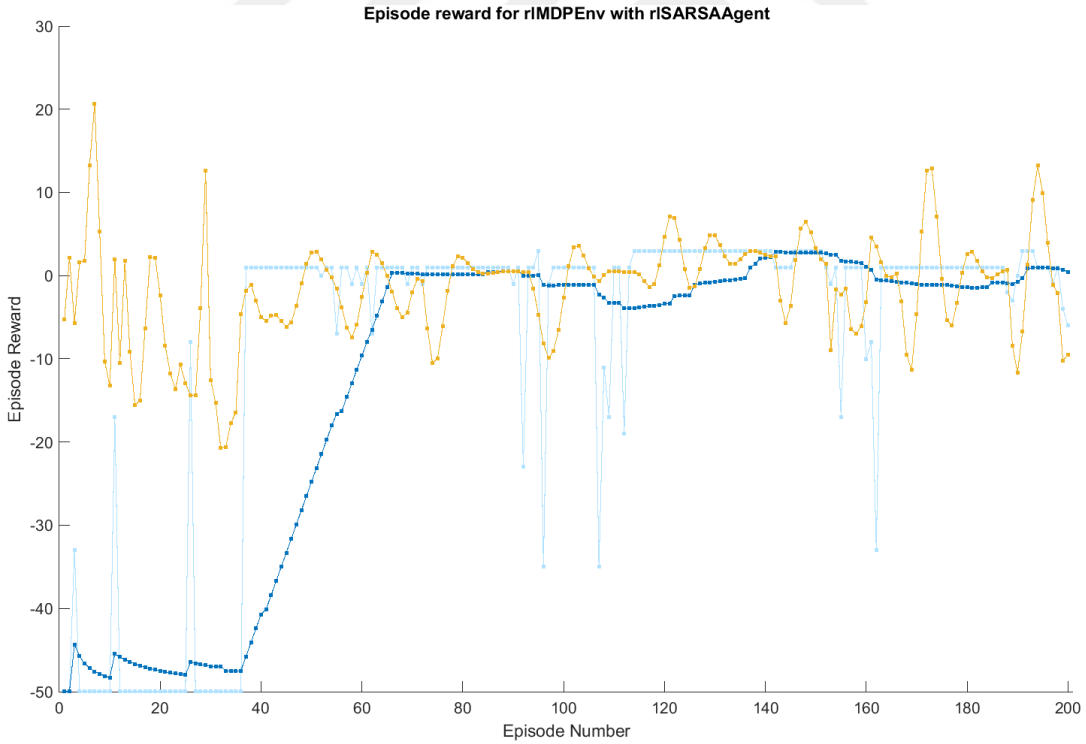
Tablo 4.1.Q ve Sarsa ajanları için kullanılan hiperparametreler ve eğitim seçenekleri

Öğrenme Oranı-α	0.99
İndirim Faktörü-γ	0.95
Epsilon-ε	0.4
Bölüm Sayısı	200
Bölüm Başına Adım Sayısı	50
Durdurma Kriteri	Ortalama Ödül
Durdurma Değeri	11
Pencere Uzunluğu	30

Şekil 4.2 ve 4.3'te sırasıyla Q ajanının ve Sarsa ajanın bölüm başına aldığı ödül miktarlarının grafiği verilmiştir. Q ajanı öğrenmeyi daha hızlı gerçekleştirmiştir. Her iki ajanda hedefe başarıyla ulaşmıştır.

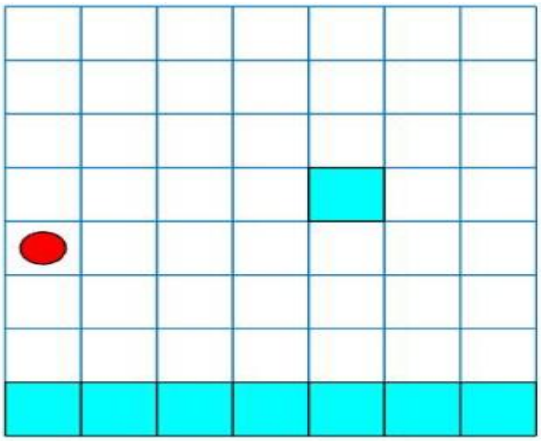


Şekil 4.2. Q ajanın ödül performansı



Şekil 4.3. Sarsa ajanın ödül performansı

Şekil 4.4'te hastane odasının grid dünyada temsili verilmiştir. Bu sefer oda daha büyük ve stokastik bir şekilde tasarlanmıştır. [4,5] noktasındaki hücre insan olarak düşünülebilir. Ortamın diğer özellikleri ise şöyledir;



Şekil 4.4. 8x7 grid dünya

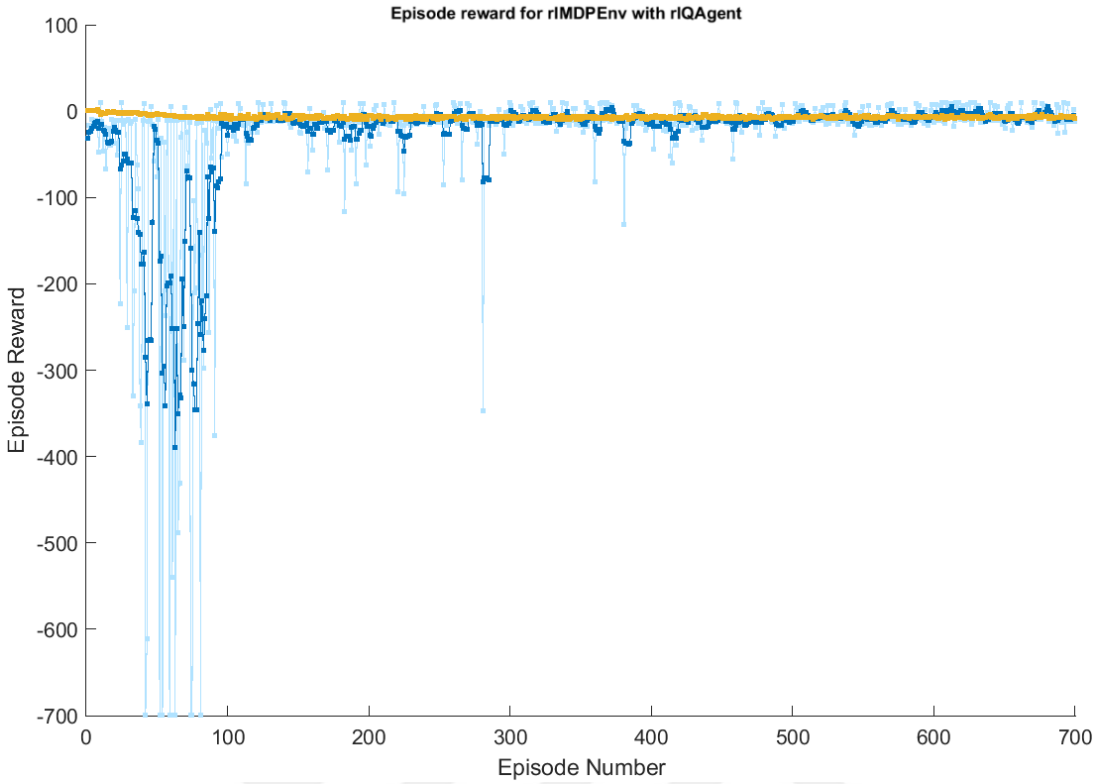
- Hedef: Mümkün olan en kısa sürede en yakın pozitif terminal durumlarına ulaşmaktır.
- Eylemler: Temsilci 4 olası yönde hareket edebilir. Sağ, sol, yukarı, aşağı
- Durumlar: 7 Pozitif Terminal (8. Sıra) ve 1 Negatif Terminal durumu (4,5) ile 56 durum vardır.
- Ödül: Tüm terminal olmayan durumların küçük bir negatif ödülü (-1) ve terminal durumlarının büyük bir pozitif ödülü (10) vardır.

Ajan hareket etmek zorundadır, bir yerde duramaz. Ortamdaki stokastiklik ajanların hareketini etkiler. Ortam, ajanı belirli bir yoğunlukta grid'in altına doğru iter. Eğer ajan [4,2] durumundan yukarı çıkarsa, [6,2] durumuna inecektir.

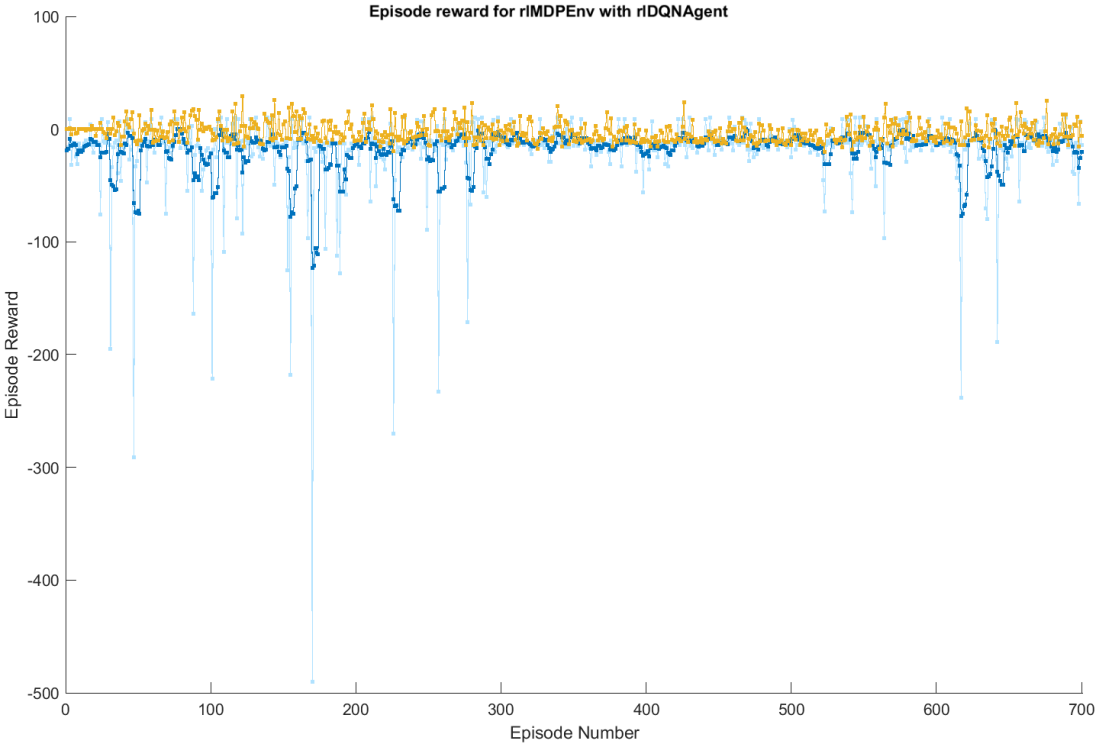
Bu ortamda, Q ajanı ve DQN ajanı eğitilmiştir. Şekil 4.5 ve 4.6'da sırasıyla Q ajanının ve DQN ajanın bölüm başına aldığı ödül miktarlarının grafiği verilmiştir. DQN ajanı öğrenmeyi daha hızlı gerçekleştirmiştir. Bunun sebebi olarak fonksiyon yaklaştırıcı olarak YSA'nın kullanılması söylenebilir. Kullanılan hiperparametreler ve eğitim seçenekleri Tablo 4.2 'de verilmiştir.

Tablo 4.2.Q ve DQN ajanları için kullanılan hiperparametreler ve eğitim seçenekleri

Öğrenme Oranı-α	1
İndirim Faktörü-γ	0.95
Epsilon-ε	0.4
Bölüm Sayısı	700
Bölüm Başına Adım Sayısı	700
Durdurma Kriteri	Bölüm Sayısı
Durdurma Değeri	700
Pencere Uzunluğu	5

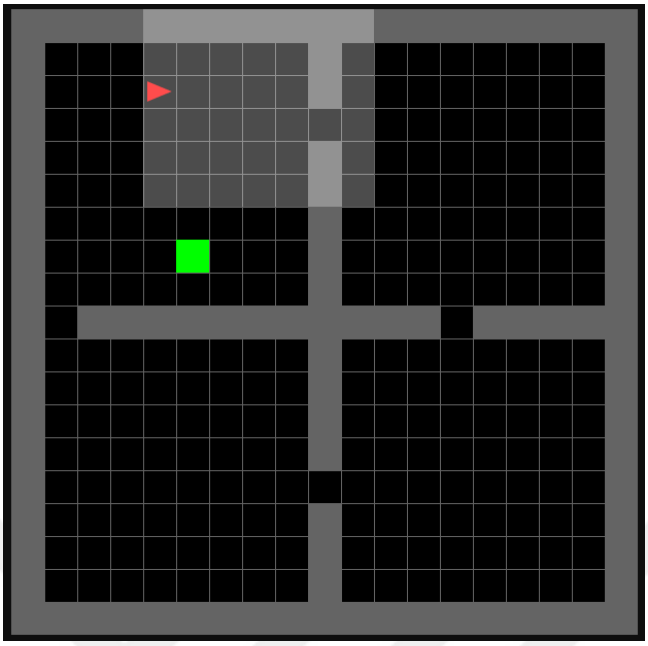


Şekil 4.5. Q ajanın 8x7 grid dünyada ödül performansı



Şekil 4.6. DQN ajanın 8x7 grid dünyada ödül performansı

4.2. MiniGrid-FourRooms

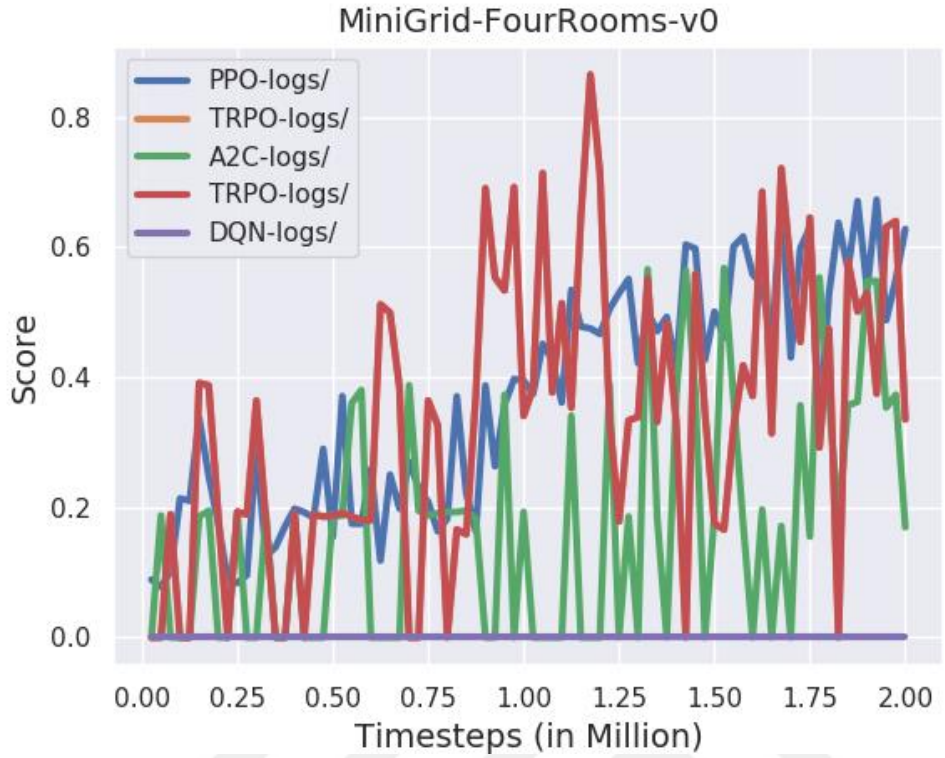


Şekil 4.7. MiniGrid-FourRooms-v0

MiniGrid-FourRooms-v0 ortamı dört odalı bir grid ortamını temsil eder. Şekil 4.7'de ortamın örnek bir temsili verilmiştir. Ajan, duvarlardaki 4 boşlukla birbirine bağlanan dört odadan oluşan bir labirentte gezinmelidir. Ajan, bir ödül elde etmek için yeşil hedef karesine ulaşmalıdır. Hem ajan hem de hedef kare dört odadan herhangi birine rastgele yerleştirilir. Durum uzayı 8x8x4, eylem uzayı 4x1 vektörlerden oluşur. Yukarı, aşağı, sağa ve sola olmak üzere 4 eylem vardır. Ödül modeli olarak ise başarı için '1- 0,9 * (adım sayısı / toplam adım)' ve başarısızlık için '0' ödülü verilir. Ajan hedefe ulaştığında yada maksimum adım sayısına ulaşıldığında bölüm sona erer.

Şekil 4.8'de DPÖ Ajanlarının bölüm başına aldığı ödül miktarlarının grafiği verilmiştir. Algoritmalar eylem ve durum uzayına göre belirlenmiştir. MiniGrid-FourRooms-v0 ortamındaki eylem ve durum uzayı ayrıktır. Buna göre A2C, DQN, PPO, TRPO algoritmaları ayrık eylem uzaylarına uygun olduğu için seçilmiştir. Ajanlar 2 milyon bölüm eğitilmiştir. DQN ajanı ödül alamamıştır. A2C, PPO, TRPO ajanlarından ise PPO daha kararlı görülmektedir ve doğru eylem seçimlerini kararlı bir şekilde yapmıştır.

Kullanılan hiperparametreler ve eğitim seçenekleri Tablo 4.3 'de verilmiştir.

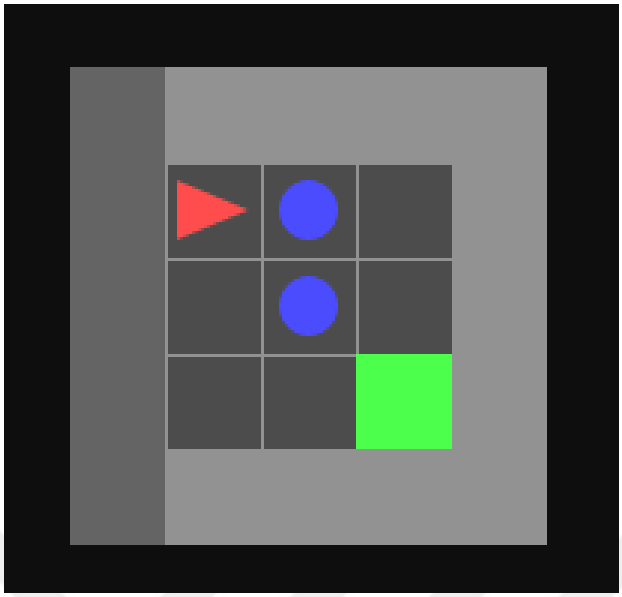


Şekil 4.8. DPÖ Ajanlarının MiniGrid-FourRooms-v0 Ortamındaki Ödül Performansı

Tablo 4.3. A2C, DQN, PPO, TRPO ajanı için kullanılan hiperparametreler ve eğitim seçenekleri

	A2C	**DQN**	**PPO**	**TRPO**
normalize	true	true	true	true
n_envs	8	8	8	8
n_timesteps	4000000	4000000	4000000	4000000
policy	'MLP'	'MLP'	'MLP’	'MLP'
n_steps	512	-	512	512
gae_lambda	0.95	-	0.95	0.95
gamma	0.99	0.99	0.99	0.99
ent_coef	0.0	-	0.0	0.0
learning_rate	2.5e-4	2.5e-4	2.5e-4	0.001
batch size		64	64	64
clip_range	-	-	0.2	-
n_epochs	-	-	10	-

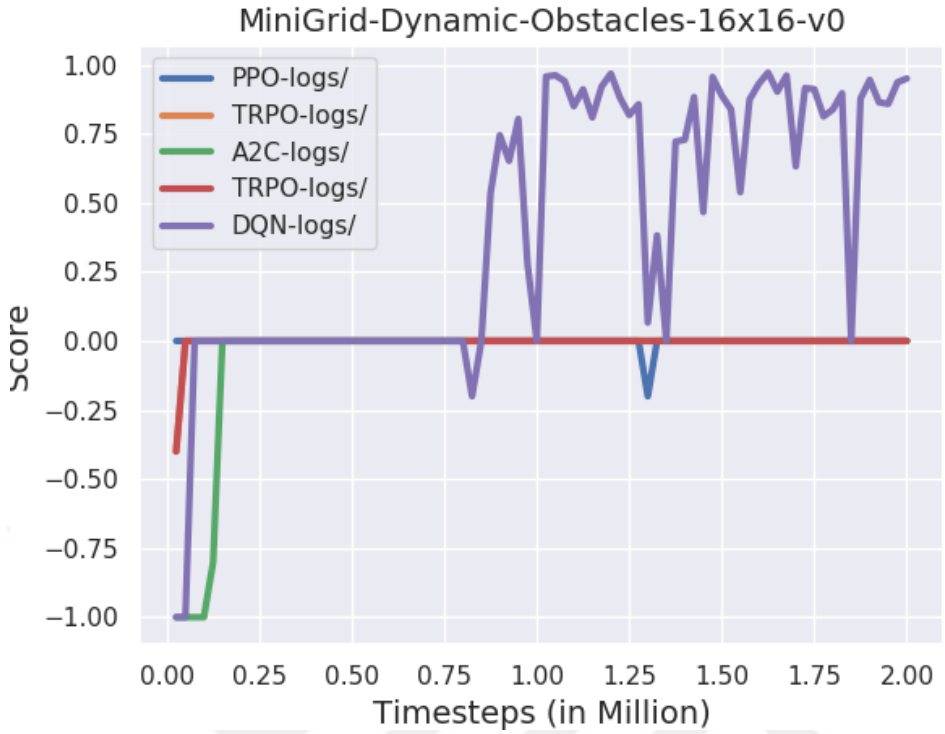
4.3. MiniGrid-Dynamic-Obstacles



Şekil 4.9. MiniGrid- Dynamic-Obstacles-16x16-v0

MiniGrid-Dynamic-Obstacles-16x16-v0 ortamı, hareketli engellerin bulunduğu boş bir odadır. Şekil 4.9'da ortamın örnek bir temsili verilmiştir. Ajanın amacı herhangi bir engele çarpmadan yeşil hedef karesine ulaşmaktır. Ajan bir engelle çarpışırsa büyük bir ceza kesilir ve bölüm sona erer. Bu ortam, Kısmi Gözlenebilirlikte Takviyeli Öğrenme ile mobil robotlar için Dinamik Engelden Kaçınma'yı test etmek için kullanışlıdır. Durum uzayı 16x16, eylem uzayı 4x1 vektörlerden oluşur. Yukarı, aşağı, sağa ve sola olmak üzere 4 eylem vardır. Ödül modeli olarak ise başarı için '1- 0,9 * (adım sayısı / toplam adım)' ve başarısızlık için '0' ödülü verilir. Ajan bir engelle çarpışırsa '-1' cezası verilir. Ajan hedefe ulaştığında, bir engele çarptığında ya da maksimum adım sayısına ulaşıldığında bölüm sona erer.

Şekil 4.10'da DPÖ Ajanlarının bölüm başına aldığı ödül miktarlarının grafiği verilmiştir. Algoritmalar eylem ve durum uzayına göre belirlenmiştir. MiniGrid-Dynamic-Obstacles-16x16-v0 ortamındaki eylem ve durum uzayı ayrıktır. Buna göre A2C, DQN, PPO, TRPO algoritmaları ayrık eylem uzaylarına uygun olduğu için seçilmiştir. Ajanlar 2 milyon bölüm eğitilmiştir. Tablo 4.3'teki hiperparametreler kullanılmıştır. A2C, DQN PPO, TRPO ajanlarından ise PPO daha kararlı görülmektedir ve doğru eylem seçimlerini kararlı bir şekilde yapmıştır.



Şekil 4.10. DPÖ Ajanlarının MiniGrid-Dynamic-Obstacles-16x16-v0 Ortamındaki Ödül Performansı

4.4. Gazebo Empty World

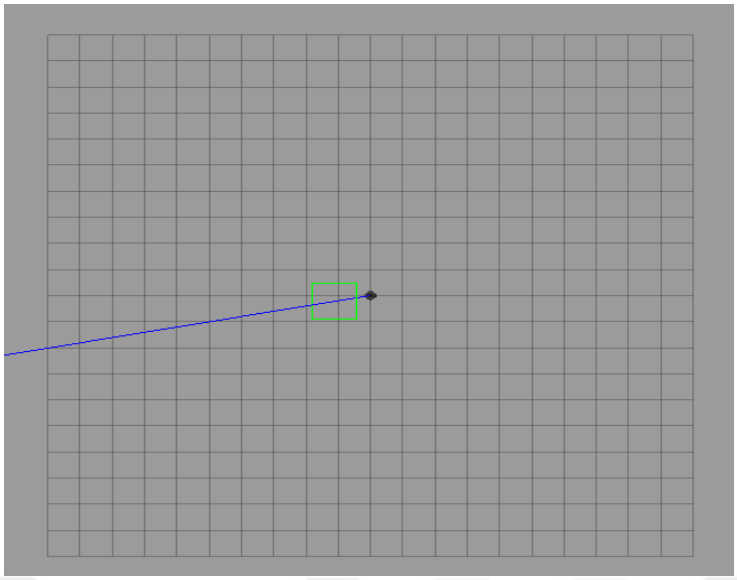
Gazebo'da boş dünya içinde hiçbir engel bulunmayan boş bir alana sahiptir. Gözlemlerin sadece uzaklık ve açı olduğu robotu hedefe götürür. Uygulanan tüm ortamlarda, ajan eylemleri doğrusal ve açısal hızlardır. Durum uzayının boyutu 38'dir. Durum uzayı, lidar sensöründen gereken 36 değerine (örnekleme miktarı), hedefe uzaklığa ve hedefe açıya sahiptir. Ajan hem açısal hem de doğrusal hızlar için normalleştirilmiş girişler kullanır, yani ajanın eylemleri -1 ile 1 arasında skalerdir. Bu dünyada sürekli eylem alanında çalışabilen TD3, PPO ve SAC algoritmaları denenmiştir. Şekil 4.11'de ortamın örnek bir temsili verilmiştir.

Ödül, robotun bir bölüm sırasındaki performansının ve yörüngesinin nicel bir ölçüsünü sağlar. Ajanın amacı bunu maksimize etmektir. Gazebo ortamında yapılan eğitimlerde bir bölüm yalnızca bir çarpışma veya zaman aşımı olduğunda sonlanır, aksi takdirde bir hedefe ulaşıldığında, robota mevcut konumundan ulaşması gereken yeni bir hedef konumu verilir. Dolayısıyla, daha yüksek kümülatif ödül, bir robotun çarpışmadan veya zaman adımı sınırını aşmadan daha fazla sayıda hedefe ulaşabildiği anlamına gelir. Ödül

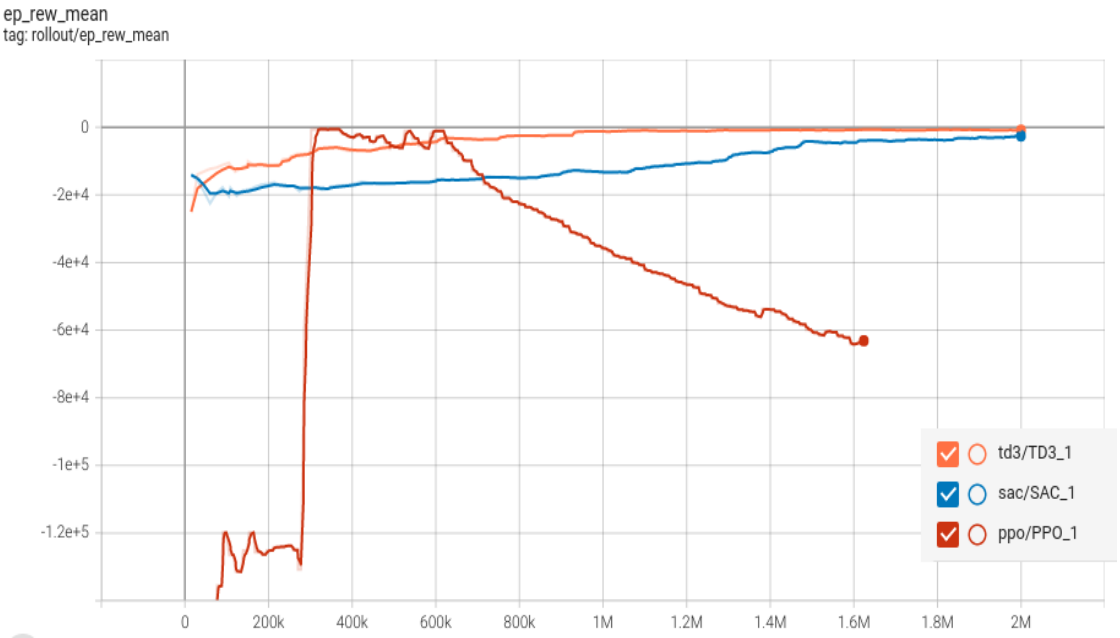
modeli olarak; hedefe varınca 100 puan, çarpışma olursa -120 puan, zaman sınırı aşılınca 0 puan belirlenmiştir. Zaman sınırı 200 zaman adımından sonra aşılır. Bu yalnızca robot hedefe ulaşamadığında veya çarpıştığında gerçekleşir. Bu, dairesel davranışı önlemek veya robotun bir yere takılıp kalmasını önlemek içindir. Kullanılan hiperparametreler ve eğitim seçenekleri Tablo 4.4 'de verilmiştir.

Tablo 4.4. TD3, PPO ve SAC ajanı için Gazebo Empty World ortamında kullanılan hiperparametreler

	PPO	**SAC**	**TD3**
training_timesteps	2000000	2000000	2000000
policy	MLP	MLP	MLP
activation_fn	relu	relu	relu
optimizer	Adam	Adam	Adam
n_steps	100	-	-
gae_lambda	0.95	-	-
gamma	0.99	0.99	0.99
ent_coef	0.0	auto	
tau	-	0.005	0.005
learning_rate	0.0003	0.0003	0.001
batch size	100	256	15000
clip_range	0.2	-	-
n_epochs	5	-	-
vf_coef	0.5	-	-
max_grad_norm	0.5	-	-
buffer_size	-	1000000	2000000
learning_starts	-	100	25000
gradient_steps	-	-1	-1
target_update_interval	-	1	-
target_entropy	-	auto	-
policy_delay	-	-	2
target_policy_noise	-	-	0.2
target_noise_clip	-	-	0.5



Şekil 4.11. Gazebo Empty World



Şekil 4.12. TD3, PPO ve SAC ajanı için Gazebo Empty World ortamında ödül performansı

TD3, PPO ve SAC ajanıları için Gazebo Empty World ortamında ödül performansı Şekil 4.12'de verildi. TD3 ajanının diğerlerinden daha başarılı olduğu görüldü. Turtlebot3 robotu çarpışmasız bir şeklide hedefe ulaşmayı başardı.

4.5. Eğitim Süreleri

Tablo 4.5'de ajanların eğitim ortamlarına göre eğitilme süreleri verildi. Basit ortam tasarımlarında eğitim süresi kısa iken tasarım karmaşıklaştıkça eğitim süresi artmaktadır. Gazebo ortamlarında eğitim sürelerinin uzunluğu dikkat çekicidir. Bu tezde,2 boyutlu ortam kullanılmasının ana sebeplerinden biri, eğitim ortamının karmaşıklığını azarlatarak donanım kaynaklarını verimli bir şeklide kullanama ve algoritmalara yoğunlaşmaktı.

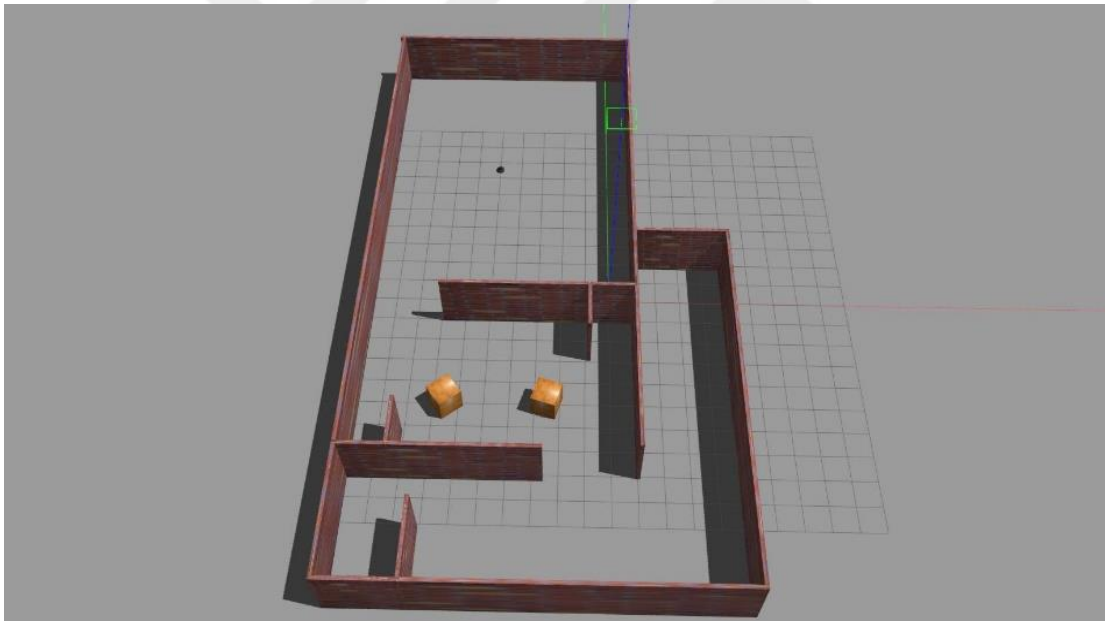
Tablo 4.5. Ajanların eğitim ortamlarına göre eğitilme süreleri

Ortam - Çevre	**Ajan**	**Eğitim Süresi**
Grid Dünya (5x5)	Q-öğrenme	55 dakika
Grid Dünya (5x5)	Sarsa	58 dakika
Dinamik Grid Dünya (8x7)	Q-öğrenme	2 saat 10 dakika
Dinamik Grid Dünya (8x7)	DQN	2 saat 30dakika
MiniGrid-FourRooms	A2C	6 saat
MiniGrid-FourRooms	DQN	4 saat
MiniGrid-FourRooms	PPO	6 saat
MiniGrid-FourRooms	TRPO	6 saat
MiniGrid-Dynamic-Obstacles	A2C	7 saat
MiniGrid-Dynamic-Obstacles	DQN	5 saat
MiniGrid-Dynamic-Obstacles	PPO	7 saat
MiniGrid-Dynamic-Obstacles	TRPO	7 saat
Gazebo Empty World	PPO	72
Gazebo Empty World	SAC	74
Gazebo Empty World	TD3	80

4.6. Test

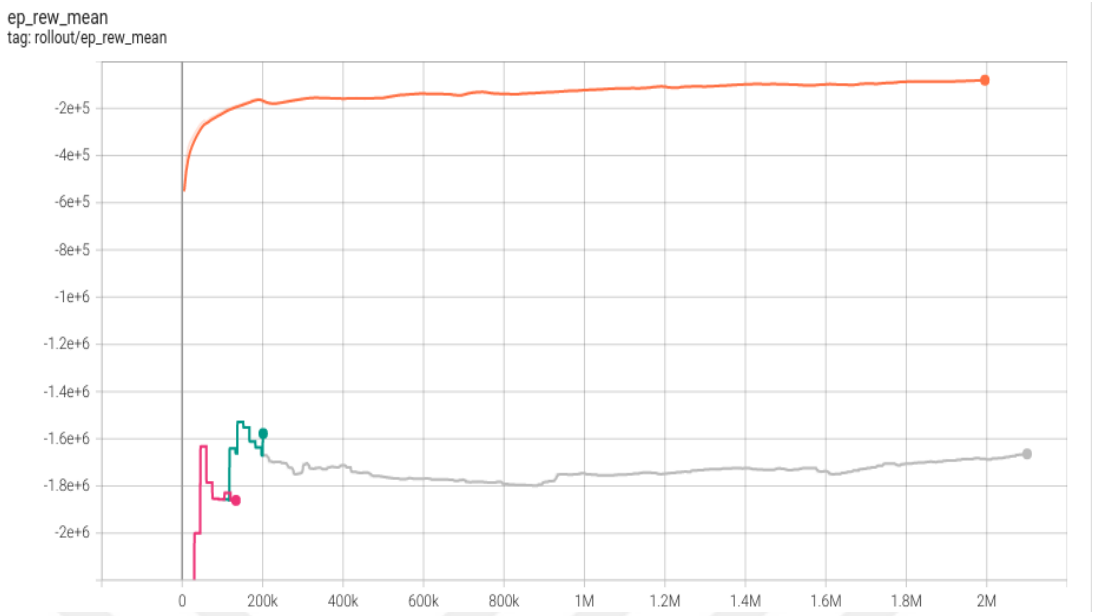
Robotik haritalama, gerçek bir ortamın bir robot veya bir grup robot tarafından dijital bir modele dönüştürüldüğü bir süreçtir. Robot nesnesinin bir sensörü, konumu ve hız parametreleri vardır. Robot ilk konumundan koşmaya başlar. Simülasyonu çalıştırırken, bu konum haritadaki boş alanın herhangi bir x-y koordinatı olabilir. Gerçek dünya deneyinde, robot o anda odanın neresinde olursa olsun, başlangıç konumunun harita üzerinde sıfır değeri vardır. Lazer sensörü okumaları, ajan tarafından yapılan gözlemler olarak kabul edilir. Uygulamada lazer sensörü okumalarının açısal konumlarını, maksimum aralığı ve gürültü parametreleri tanımlandı. Ajanın eylemi, robotun doğrusal ve açısal hızlarının bulunduğu ve olduğu iki boyutlu bir vektördür. Ajan, hem açısal hem de doğrusal hızlar için normalleştirilmiş girişler kullanır, Yani ajanın eylemleri -1 ile 1 arasında skalerdir. Ajan, en kötü durum senaryosunu en aza indiren, en yakın engelden

kaçınması için ödüllendirilir. Ek olarak, ajana daha yüksek doğrusal hızlar için pozitif bir ödül verilir ve daha yüksek açısal hızlar için negatif bir ödül verilir. Bu ödüllendirici strateji, ajanın daireler çizme davranışını caydırır. Şekil 4.13'de gösterilen Gazebo ortamı, iki dinamik engelin bulunduğu bir labirenttir. Bu ortamda, haritasız navigasyon mobil robotikte kullanılan yerel bir planlayıcı, önceden planlanmış bazı yörüngeler robota sağlanır. PÖ haritasız navigasyon algoritması önceden planlanmış noktalardan beslenir yörünge istenen hedefler olarak; robot bir hedefe ulaştığında noktasından sonra, yörüngedeki bir sonraki nokta son noktaya kadar gönderilir yörünge tespit edilir. Ortam gözlemleri, robottan hedefe olan mesafe ve açı ile LIDAR ölçümlerinden oluşmaktadır. Uygulanan tüm ortamlarda, ajan eylemleri doğrusal ve açısal hızlardır. Ödül modeli Gazebo Empty World ile aynıdır. Tek fark çarpışma durumunda verilen cezadır. Hedefe varınca 100 puan, çarpışma olursa -200 puan, zaman sınırı aşılınca 0 puan belirlenmiştir. Zaman sınırı 100 zaman adımından sonra aşılır.



Şekil 4.13. Gazebo Maze World

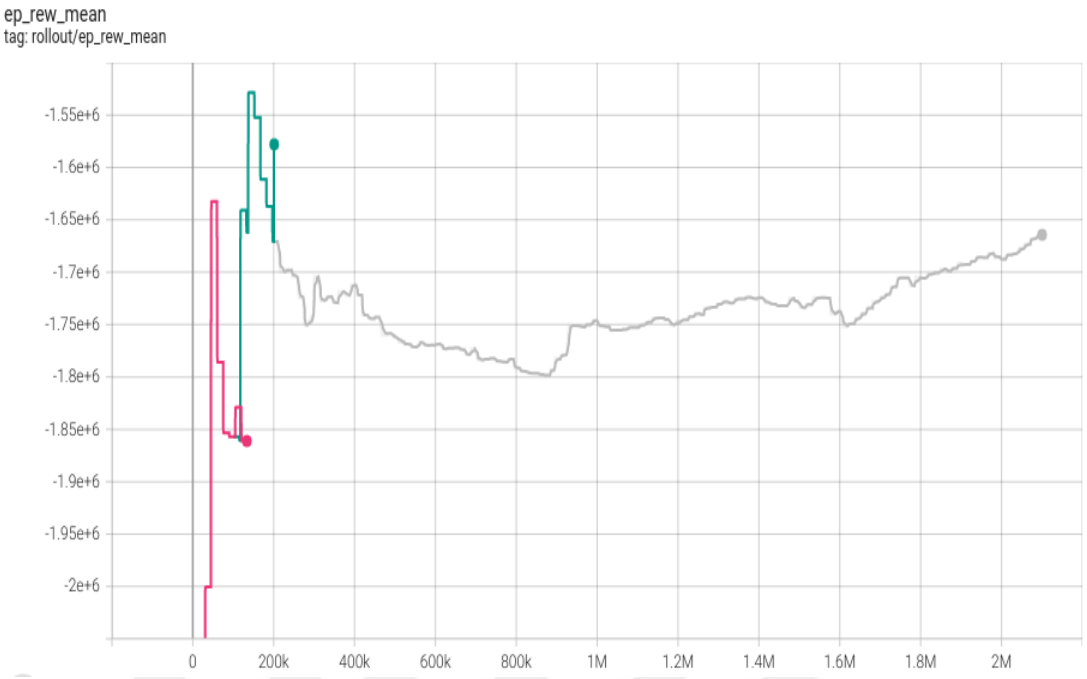
Daha önceki deneylerde PPO ve TD3 algoritmasının başarısı ön plana çıkmıştır. PPO hem ayrık hem de sürekli eylem uzaylarında çalışabilirken TD3 sadece sürekli eylem uzaylarında çalışabilir. PPO algoritması TRPO algoritmasının geliştirilmiş hali iken TD3 algoritması DDPG algoritmasının geliştirilmiş halidir. Bundan dolayı test ortamında PPO ve TD3 algoritmaları karşılaştırıldı.



Şekil 4.14. TD3 ve PPO ajanı için Gazebo Maze World ortamında ödül performansı

Şekil 4.14'de TD3 ve PPO ajanı için Gazebo Maze World ortamında ödül performansı verilmiştir. Her iki ajanda 2 milyon bölüm eğitilmiştir. Hiperparametre olarak Gazebo Empty ortamı ve grid dünyalarda kullanılan parametreler kullanıldı.TD3 ajanını daha yüksek ödüllere daha hızlı ulaştığı görüldü. Robotun başarılı bir şekilde hedefe ulaştığı görüldü.

Şekil 4.15. TD3 ajanı için Gazebo Maze World ortamında ödül performansı



Şekil 4.16. PPO ajanı için Gazebo Maze World ortamında ödül performansı

5. SONUÇLAR VE ÖNERİLER

Bu tezde, bilinmeyen ortamlarda otonom, haritasız mobil robot navigasyonu için Derin Pekiştirmeli Öğrenme yaklaşımı incelenmiştir. Yol planlama ve otonom navigasyon problemlerine çözüm olarak son yıllarda üzerine oldukça araştırma yapılan pekiştirmeli öğrenme algoritmaları incelendi. Farklı DPÖ algoritmaları farklı ortamlarda çalıştırıldı ve performansları bölüm başına ödül miktarı olarak ölçüldü. Önerilen ödül modeli robotun dairesel davranış yapmasını ve bir yere takılıp kalmasın engellemek için ve hedefe çarpışma olmadan varmasını sağlamak için tasarlandı.

Son teknoloji DPÖ algoritmaları olan DQN, DDPG, A2C, TRPO, TD3, PPO karşılaştırıldı. Bu algoritmalar, DPÖ araştırma topluluğundaki mevcut literatüre ve popülerliğe göre seçilmiştir(Fujimoto ve diğ., 2018; Raffin ve diğ., 2021)

Ortam tasarımı olarak mevcut donanım kaynaklarını verimli kullanmak adına öncelikle 2 boyutlu ortam tasarımında deneyimler elde edildi.2 boyutlu ortam olarak Matlab'ta grid dünya ve ayrıca Minigrid kütüphanesinden 2 adet ortam kullanıldı.3 boyutu ortam için ise Gazebo benzetim ortamı kullanıldı.

Parametre optimizasyon yöntem olarak 2 boyutlu ortamda denemeler yapıldı. Elde edilen tecrübeler 3 boyutlu dünyaya aktarıldı. Hiperparamtere optimizasyonu için RL Baselines 3 Zoo kütüphanesinde yararlanıldı.

Grid dünyası, çalışma ortamını görsellerden kolayca tanımlayabilmesi ve pekiştirmeli öğrenme girdisine giren durumun ve bu durumdaki davranışın basit olması avantajlarından dolayı yol arama probleminde sıklıkla kullanılan ortamı ifade eden bir tekniktir. Ancak, çalışma ortamını ifade etme işi sadece görüntüyü grid bir dünyaya dönüştürmekle bitmiyor. Uygun bir ödül fonksiyonu kurma, pekiştirmeli öğrenme ile çözülecek bir problem oluşturma, bir pekiştirmeli öğrenme algoritması seçme ve son olarak pekiştirmeli öğrenme algoritmasının öğrenme parametrelerine değer atama süreci tasarımcının sorumluluğundadır. Bu tezde, problemi çözmek için kullanılan derin pekiştirmeli öğrenme algoritmasının model mimarisi ve parametreleri ve grid dünyasında yol arama probleminde grid dünyası çevre tasarımı örneği sunuldu. Ardından, çeşitli derin takviyeli öğrenmenin öğrenme algoritmaları farklı ortamlarda çalıştırıldı ve performansları gözlemlendi.

Basit ortamlarda Q-öğrenme yaklaşmışının başarılı olduğu gözlendi. Durum uzayı büyüdükçe Q tablolarının yetersiz kaldığı tespit edildi. Bu noktada fonksiyon yaklaştırıcısı yöntemi incelendi. Yapay sinir ağı modelleri incelendi. DQN ajanının getirdiği yaklaşımın Q-öğrenmeden daha başarılı olduğu görüldü. Grid dünyada hastane modellemesinin stokastik versiyonunda DQN ajanının Q-öğrenme ajanından daha başarılı olduğu tespit edildi.

MiniGrid ortamlarında A2C, DQN, TRPO, PPO ajanları eğitildi. PPO ajanının daha başarılı olduğu görüldü. Bunun sebebi olarak politika tabanlı bir yaklaşım kullanılması söylenebilir.

Sürekli eylem uzaylarında çalışan TD3, SAC, PPO algoritmaları Gazebo Empty ortamında kıyaslandı.TD3 ajanı daha başarılı sonuçlar aldı.

Test ortamı olarak Gazebo Maze World kullanıldı. Bu ortamda Turtlebot robotu kullanıldı. PÖ modelleri, lazer sensör gözlemleri kullanılarak eğitildi. Lazer sensör gözlemleri, daha az işlem gerektirdiği için kamera gözlemlerine kıyasla modelleri eğitmek daha hızlı olduğu için kullanıldı. Sürekli eylem uzaylarında çalışabilen ve daha önceki deneylerde başarıları ile ön plana çıkan TD3, PPO algoritmaları kıyaslandı. TD3 ajanı daha başarılı sonuçlar aldı.

Sonuç olarak; sonsuz olası durumlara sahip birçok durumda, doğrusal yaklaşım ve Q tablosu kullanmanın doğru yaklaşım olmadığı görüldü. DQN algoritmasının sürekli durumlar ve ayrık eylemler üzerinde başarılı olduğu ve sinir ağı kullanan doğrusal olmayan fonksiyon yaklaşımının çok güçlü bir yaklaşım olduğu sonucuna varıldı. Sürekli eyleme ve sürekli durum uzayında sahip ortamlarda ise PPO, TD3 algoritmalarını kullanılabileceği ve başarılı olunabileceği görüldü. Bunun sebebi olarak fonksiyon yaklaştırıcısı olarak aktör-kritik ağ modelinin kullanılması söylenebilir.

Bu tezde, bir mobil robotun 2 boyutlu grid ortamda ve 3 boyutlu bir oda ortamında yol planlaması yapması için çeşitli DPÖ algoritmaları çalıştırıldı. Kullanılan algoritmaların yol planlama problemini çözebileceği görüldü.

Gelecekte ise bu çalışmada kullanılan ve başarıları gözlenen PPO, TD3 gibi algoritmaların ağ ve ödül modeli üzerine çalışarak bunların öğrenmeye olan etkisinin

tespit edilmesi planlandı. Reward Shaping(Ödül fonksiyonu modelleme) pekiştirmeli öğrenme konusunun en zorlu araştırma konularından biridir. Bu konu üzerine özellikle çalışılarak algoritmalar toplam ödül, başarı oranı, en kısa yolu bulma gibi ölçütler üzerinden kıyaslanacaktır. Ayrıca geleneksel yol planlama algoritmaları (A*,RRT, vb.) ile DPÖ algoritmaları hibrit bir şekilde kullanılarak yol planlama problemlerine çözüm aranacaktır. Bu anlamda global bir yol planı klasik algoritmalar ile oluşturulup hareket planlaması DPÖ ajanları ile yapılabilir, kapalı ortamlar segmentlere ayrılıp her segmentte DPÖ ajanı çalışabilir ve segmentler arası bağlantı A*, RRT gibi algoritmalarla sağlanabilir ya da bir mobil robotun bir hedefe tam anlamıyla varması için belirli bir güvenli mesafeye (çarpışma olamayacak mesafe) kadar DPÖ ajanı o noktadan sonra geleneksel yol planlama algoritmaları kullanılabilir. Nihai hedef ise en iyi performans veren algoritmanın, simülasyon ortamından gerçek robota aktarılmasıdır.

KAYNAKLAR

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *IEEE Symposium on Security and Privacy*, 265-283. https://doi.org/10.1109/SP.2016.29.

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship Learning via Inverse Reinforcement Learning. *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004), 1–8.* https://doi.org/10.1145/1015330.1015430

Abel, D. (2020). A Theory of Abstraction in Reinforcement Learning. Phd Thesis, Department of Computer Science at Brown University.

Aghaei, V., Onat, A., & Yıldırım, S. (2018). A Markov Chain Monte Carlo Algorithm For Bayesian Policy Search. *Systems Science & Control Engineering, 6*(1), 438-455. Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A. S., & Asari, V. K. (2019). A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics (Switzerland), 8*(3), 1–67. https://doi.org/10.3390/electronics8030292

Al-Taharwa, I., Sheta, A., & Al-Weshah, M. (2008). A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment. *Journal of Computer Science, 4*(4), 341–344. https://doi.org/10.3844/jcssp.2008.341.344

Altunaş, N., Imal, E., Emanet, N., Öztürk, C. N. (2016). Reinforcement Learning-Based Mobile Robot Navigation. Turkish Journal of Electrical Engineering and Computer Sciences, 24(3), 1747-1767.

Aradi, S. (2020). Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems, 23*(2), 740–759. https://doi.org/10.1109/TITS.2020.3024655

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine, 34*(6), 26-38. https://doi.org/10.1109/MSP.2017.2743240

Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., & Kautz, J. (2017). Reinforcement Learning Through Asynchronous Advantage Actor-Aritic on a GPU. *In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017),* Toulon, France, April 24-27, 2017.

Bellman, R. (1957a). A Markovian Decision Process. *Indiana University Mathematics Journal, 6*(4), 679–684. https://doi.org/10.1512/iumj.1957.6.56038

Bellman, R. (1957b). *Dynamic Programming* (6th ed.). Princeton, NJ: Princeton University Press.

Bölük, N., Ucar, O., Inner, A.B. (2019). Mobil Robotlarda Navigasyon Problemi için Pekiştirmeli Öğrenme. *Türkiye Robotbilim Konferansı*, İstanbul, Türkiye,26-28 Haziran 2019.

Brémaud, P. (1999). *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. New York: Springer-Verlag.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arxiv preprint arXiv:1606.01540.*

Celebi, M.E., Aydin, K. (2016). *Unsupervised Learning Algorithms.* Switzerland : Springer International Publishing.

Çetin, H., Durdu, A. (2014). Path Planning of Mobile Robots with Q-learning. *22nd Signal Processing and Communications Applications Conference (SIU), Trabzon, Turkey*, 2162-2165. https://doi.org/10.1109/SIU.2014.6830691

Chance P. (1999). Thorndike's Puzzle Boxes And The Origins Of The Experimental Analysis Of Behavior. *Journal of Experimental Analysis of Behavior, 72*(3), 433-440. https://doi.org/10.1901/jeab.1999.72-433

Chen, Y. F., Everett, M., Liu, M., How, J. P. (2017). Socially Aware Motion Planning With Deep Reinforcement Learning. *In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1343-1350).* Vancouver, Canada, doi:10.1109/IROS.2017.820231

Cheng, J., Cheng, H., Meng, M. Q. H., & Zhang, H. (2018, December). Autonomous navigation by mobile robots in human environments: A survey. IEEE Transactions on Robotics, 34(6), 1309-1332. doi:10.1109/TRO.2018.2867592

Cheng, J., Cheng, H., Meng, M. Q.-H. ve Zhang, H. (2018). Autonomous Navigation by Mobile Robots in Human Environments: A Survey. *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Kuala Lumpur, Malaysia, 12-15 December 2018 https://doi.org/10.1109/ROBIO.2018.8665075

Demir, A. (2019). Derin Pekiştirmeli Öğrenme Kullanarak Rastgele Yükler ile Hareket Planlama ve Kontrol. Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 550541

Ejaz, M. M., Tang, T. B. ve Lu, C. -K. (2019). Autonomous Visual Navigation using Deep Reinforcement Learning: An Overview. *2019 IEEE Student Conference on Research and Development (SCOReD),* Bandar Seri Iskandar, Malaysia, 294-299. https://doi.org/10.1109/SCORED.2019.8896352

Engin, C. D. (2019). Otonom Mobil Robotların Pekiştirmeli Öğrenme ile Kontrolü. Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 604642

Everett, M., Chen, Y. F. ve How, J. P. (2018). Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3052-3059. https://doi.org/10.1109/IROS.2018.8593871

Everett, M., Chen, Y. F. ve How, J. P. (2018). Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. 2018 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 3052-3059. https://doi.org/10.1109/IROS.2018.8593871

Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., & Mizutani, K. (2017). State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. IEEE Communications Surveys and Tutorials, 19(4), 2432-2455. [7932863]. https://doi.org/10.1109/COMST.2017.2707140

Fox, D., Burgard, W., Thrun, S. (1997). Collision Avoidance For Mobile Robots: A Dynamic Window Approach. *IEEE Robotics & Automation Magazine, 4*(1), 23-33. doi:10.1109/100.580977

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G. ve Pineau, J. (2018). An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning, 11*(3-4), 219-354. https://doi.org/10.1561/2200000

Fujimoto, S., van Hoof, H. ve Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, Stockholmsmässan, Stockholm, Sweden, 1582-1591. Proceedings of Machine Learning Research (PMLR), 80, 2018

Garrote, L., Premebida, C., Silva, M., & Nunes, U. (2014). An RRT-based navigation approach for mobile robots and automated vehicles. *Robotics and Autonomous Systems*, *62*(11), 1506-1517. doi:10.1016/j.robot.2014.07.003

Gong, Z., Liang, P., Feng, L., Cai, T., Xu, W. (2011). Comparative Analysis Between Gazebo And V-REP Robotic Simulators. *Robotics and Computer-Integrated Manufacturing, 27(*6), 957-966. doi: 10.1016/j.rcim.2011.07.005

Güçkıran, K. ve Bolat, B. (2019). Autonomous Car Racing in Simulation Environment Using Deep Reinforcement Learning. *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1-6, İzmir, Turkey, 31 October 2019- 02 November 2019

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning With A Stochastic Actor. *In Proceedings of the 35th International Conference on Machine Learning (ICML) (pp. 1861-1870). PMLR*. doi:10.1561/2200000071

Harnad, S. (2006). *The Annotation Game: On Turing (1950) on Computing, Machinery, and Intelligence.* In Epstein, R. ve Peters, G. (Eds.), [Book Chapter] (in Press). Kluwer Academic Publishers.

Hu, Y., Yang, S. X. (2004). A Knowledge Based Genetic Algorithm For Path Planning Of A Mobile Robot. *IEEE Transactions on Robotics and Automation, 20*(6), 1059-1068. doi:10.1109/TRO.2004.836473

Goodfellow, I., Bengio, Y., Courville, A. C. (2016). *Deep Learning*. Massachusetts: MIT Press.

Ivaldi, S., Padois, V., Nori, F. (2014). Tools For Dynamics Simulation Of Robots: A Survey Based On User Feedback. *ArXiv, abs/1402.7050.*

James, S. (2016). 3D Simulated Robot Manipulation Using Deep Reinforcement Learning. Phd Thesis, Imperial College London.

Kang, S.-C., Chang, W.-T., Gu, K.-Y., Chi, H.-L. (2011). *Robot Development Using Microsoft Robotics Developer Studio*. New York: Chapman and Hall/CRC.

Kardell, S., Kuosku, M. (2017). Autonomous Vehicle Control Via Deep Reinforcement Learning. Master's thesis, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden.

Keller, M., Hoffmann, F., Bertram, T., Hass, C., Seewald, A. (2014). Planning Of Optimal Collision Avoidance Trajectories With Timed Elastic Bands. *IFAC Proceedings Volumes, 47*(3), 9822-9827. doi:10.3182/20140824-6-ZA-1003.01143

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Jaśkowski, W. (2016, June). Vizdoom: A Doom-Based AI Research Platform For Visual Reinforcement Learning. *In 2016 IEEE Conference on Computational Intelligence and Games (CIG) (pp. 1-8).* Santorini, Greece, 20-23 September 2016. doi:10.1109/CIG.2016.7860433

Kersandt, K. (2018). Deep Reinforcement Learning As Control Method For Autonomous Uavs. Master's thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

Khan, M. U. (2019). Mobile Robot Navigation Using Reinforcement Learning in Unknown Environments. *Balkan Journal of Electrical and Computer Engineering, 7*(3), 235–244. https://doi.org/10.17694/bajece.532746

Koenig, N., Howard, A. (2004, October). Design And Use Paradigms For Gazebo, An Open-Source Multi-Robot Simulator. *In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566) (pp. 2149-2154 vol.3).* Sendai, Japan, 28 September 2004- 02 October 2004 doi:10.1109/IROS.2004.1389727

Lefrançois, G. R. (2000). *Theories of human learning : What The Old Man said* (4th ed.). Wadsworth: Thomson Learning.

Levine, S., Finn, C., Darrell, T., Abbeel, P. (2016). End-To-End Training Of Deep Visuomotor Policies. *The Journal of Machine Learning Research, 17*(39), 1-40.

Lewis, F. L., Vrabie, D., Vamvoudakis, K. G. (2012). Reinforcement Learning And Feedback Control: Using Natural Decision Methods To Design Optimal Adaptive Controllers. *IEEE Control Systems Magazine, 32*(6), 76-105. doi:10.1109/MCS.2012.2214134.

Li, Y. (2018). Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv:1801.07240.*

Liang, D., Patel, A. K., Siegwart, R. (2020). Real-Time Collision Avoidance For Mobile Robots İn Dense Crowds Using İmplicit Multi-Sensor Fusion And Deep Reinforcement Learning. *In Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020) (pp. 1087-1094)*, Auckland, New Zealand, May 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. (2016). Continuous Control With Deep Reinforcement Learning. *4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, May 2-4, 2016.

Liu, X., Gong, D. (2011). A Comparative Study Of A-Star Algorithms For Search And Rescue İn Perfect Maze. *2011 International Conference on Electric Information and Control Engineering*, Wuhan, 15-17 April 2011.doi: 10.1109/ICEICE.2011.5777723.

Marin-Plaza, P., Hussein, A., Martin, D., Escalera, A. (2018). Global And Local Path Planning Study İn A ROS-Based Research Platform For Autonomous Vehicles. *Journal of Advanced Transportation*, February 22, 2018,.https://doi.org/10.1155/2018/6392697

Puterman, M. L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ: John Wiley & Sons.

Michel, O. (2004). Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems, 1*(1), 5-20. doi:10.5772/5618

Miller, A. T., Allen, P. K. (2004). Graspit: A Versatile Simulator For Robotic Grasping. *IEEE Robotics & Automation Magazine, 11*(4), 110-122. doi:10.1109/MRA.2004.1371616

Mitchell, T. M. (1997). *Machine Learning*. New York, NY: McGraw-Hill.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Petersen, S. (2013). Playing Atari With Deep Reinforcement Learning. *Nature, 518*(7540), 529-533. doi:10.1038/nature12271

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Petersen, S. (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature, 518*(7540), 529-533. doi:10.1038/nature14236

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., Kavukcuoglu, K. (2016). Asynchronous Methods For Deep Reinforcement Learning. *In Proceedings of the 33rd International Conference on Machine Learning (pp. 1928-1937)*, San Francisco, California, USA. June 21-26 2016

Muhammad, J., & Bucak, I. O. (2013). An İmproved Q-Learning Algorithm For An Autonomous Mobile Robot Navigation Problem. *In 2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE) (pp. 239-243)*, Konya, Turkey. June 14-16 2013. doi:10.1109/TAEECE.2013.6557278

Nasteski, V. (2017). An Overview Of The Supervised Machine Learning Methods. *Journal of Strategic Innovation and Sustainability, 4*(1), 1-17. doi:10.2139/ssrn.32814611

Pandey, A. (2017). Mobile Robot Navigation And Obstacle Avoidance Techniques: A Review. *International Robotics & Automation Journal, 2*(2), 96-108. doi:10.15406/iratj.2017.02.00023

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An İmperative Style, High-Performance Deep Learning Library. *In Advances in Neural Information Processing Systems (NIPS) 2019 (pp. 8024-8035)*. Montreal, Quebec, Canada , 6-12 December, 2019.

Patle, B. K., Babu L, G., Pandey, A., Parhi, D. R. K., Jagadeesh, A. (2019). A Review: On Path Planning Strategies For Navigation Of Mobile Robot. *Defence Technology, 15*(4), 582-606. doi: 10.1016/j.dt.2019.04.011.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A. (2009). ROS: An Open-Source Robot Operating System. *In Proceedings of the ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5)*. Kobe, Japan,May 2019.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning İmplementations. *Journal of Machine Learning Research, 22*(268), 1-8.

Reibman, A., Smith, R., & Trivedi, K. (1989). Markov And Markov Reward Model Transient Analysis: An Overview Of Numerical Approaches. *European Journal of Operational Research, 40*(2), 257-267. doi: 10.1016/0377-2217(89)90052-5

Risald, A. E., Mirino, A. E., & Suyoto. (2017). Best Routes Selection Using Dijkstra And Floyd-Warshall Algorithm. *In 2017 11th International Conference on Information & Communication Technology and System (ICTS) (pp. 155-158). IEEE*, June 2-4 2017. doi: 10.1109/ICTS.2017.822647365

Rothman, D. (1984). Monte Carlo Techniques: An Overview. *Statistical Science, 1*(3), 235-247. doi: 10.1214/ss/1177692200

Ruan, X., Ren, D., Zhu, X., Huang, J. (2019). Mobile Robot Navigation based on Deep Reinforcement Learning. *2019 Chinese Control And Decision Conference (CCDC)*, Nanchang, China, 3-5 June 2019. DOI: 10.1109/CCDC.2019.8832393

Santos Pessoa de Melo, M., Gomes da Silva Neto, J., Jorge Lima da Silva, P., Natario Teixeira, J. M. X. ve Teichrieb, V. (2019). Analysis and Comparison of Robotics 3D Simulators. *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, Rio de Janeiro, Brazil, 242-251. https://doi.org/10.1109/SVR.2019.00049

Schmidhuber, J. (2015). Deep Learning İn Neural Networks: An Overview. *Neural Networks, 61*, 85-117. doi: 10.1016/j.neunet.2014.09.003

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust Region Policy Optimization. *In International Conference on Machine Learning (pp. 1889-1897). PMLR.* San Francisco, CA., 22-26 June 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347.*

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., ... & Dieleman, S. (2016). Mastering The Game Of Go With Deep Neural Networks And Tree Search. *Nature, 529*(7587), 484-489. doi:10.1038/nature16961

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815.*

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815.*

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Dieleman, S. (2016). Mastering The Game Of Go Without Human Knowledge. *Nature, 529*(7587), 484-489. doi:10.1038/nature16961

Skinner, B. F. (1938). *The Behavior of Organisms: An Experimental Analysis.* New York, NY: Appleton-Century-Crofts.

Sobell, M. G. (2015). *A Practical Guide to Ubuntu Linux* (5th ed.). Sebastopol, CA: O'Reilly Media.

Stentz, A., Hebert, M. (1995). The Focussed D* Algorithm for Real-Time Replanning. *In Proceedings of the 14th International Joint Conference on Artificial Intelligence (pp. 1652-1658).* San Francisco, CA, 18-22 March 1995.

Şucan, I. A., Moll, M., Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine, 19*(4), 72-82. doi:10.1109/MRA.2012.2205651.

Sung, T. T., Kim, C., Lee, K., Sohn, C. B. (2018). Exploring Navigation using Deep Reinforcement Learning. *International Journal of Applied Engineering Research, 13*(19), 14447-14450.

SunWoo, Y., Lee, W. (2021). Comparison Of Deep Reinforcement Learning Algorithms: Path Search İn Grid World. *In 2021 International Conference on Electronics, Information, and Communication (ICEIC) (pp. 1-3),* Jeju, Korea(South), 31 Jan.-3 Feb. 2021. doi: 10.1109/ICEIC51217.2021.9369800.

Sutton, R. S. (1988). Learning To Predict By The Methods Of Temporal Differences. *Machine Learning, 3*(1), 9-44. doi:10.1007/BF00115009

Sutton, R. S., Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge: MIT press.

Sutton, R. S., Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). Cambridge: MIT Press.

Tai, L., Liu, M. (2016). Towards Cognitive Exploration Through Deep Reinforcement Learning For Mobile Robots. *Autonomous Robots, 38*(3), 405-420. doi:10.1007/s10514-016-9496-8

Tai, L., Paolo, G., & Liu, M. (2017). Virtual-To-Real Deep Reinforcement Learning: Continuous Control Of Mobile Robots For Mapless Navigation. *In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (p. 31-36).* Vancouver, BC, Canada, Semptember 2017. doi:10.1109/IROS.2017.8202134

Taïga, A. A., Courville, A., Bellemare, M. G. (2018). Approximate Exploration Through State Abstraction. *arXiv preprint arXiv:1808.09819*.

Talabis, M. R. M., McPherson, R., Miyamoto, I., Martin, J. L., Kaye, D. (2015). Chapter 1 - Analytics Defined. In M. R. M. Talabis, R. McPherson, I. Miyamoto, J. L. Martin, & D. Kaye (Eds.), *Information Security Analytics* (pp. 1-12). Syngress. Boston. DOI: 10.1016/B978-0-12-800207-0.00001-0.

Thorndike, E. L. (1898). Animal İntelligence: An Experimental Study Of The Associative Processes İn Animals. *The Psychological Review: Monograph Supplements, 2*(4), i–109. https://doi.org/10.1037/h0092987

Thrun, S. (1998). Learning Metric-Topological Maps For İndoor Mobile Robot Navigation. *Artificial Intelligence, 99*(1), 21-71.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A Physics Engine For Model-Based Control. *In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5026-5033).* Seville, Spain, September 13-17, 2012. doi:10.1109/IROS.2012.6224805

Tomatis, N., Nourbakhsh, I., Siegwart, R. (2001). Combining Topological And Metric: A Natural İntegration For Simultaneous Localization And Map Building. *Proceedings of the Fourth European Workshop on Advanced Mobile Robots (EUROBOT)*, 207-214.

Tsitsiklis, J. N., Van Roy, B. (1997). An Analysis Of Temporal-Difference Learning With Function Approximation. *IEEE Transactions on Automatic Control, 42*(5), 674-690.

URL-1:https://www.deepmind.com/, (Ziyaret Tarihi:10 Nisan 2022).

URL-2: https://spinningup.openai.com/en/latest/algorithms/sac.html/, (Ziyaret tarihi:23 Mart 2022).

URL-3:http://www.cyberbotics.com/, (Ziyaret tarihi:23 Mart 2022).

URL-4:https://coppeliarobotics.com/, (Ziyaret tarihi:30 Mart 2022).

URL-5:https://www.robologix.com/, (Ziyaret tarihi:30 Nisan 2022).

URL-6:http://www.anykode.com/index.php, (Ziyaret tarihi:2 Mayıs 2022).

URL-7:https://box2d.org/documentation/, (Ziyaret tarihi:23 Mart 2022)

URL-8:https://emanual.robotis.com/ Turtlebot3features, (Ziyaret tarihi:10 Temmuz 2022)

URL-9:https://www.python.org/, (Ziyaret tarihi:10 Eylül 2022).

URL-10: https://minigrid.farama.org/ (Ziyaret tarihi:12 Mart 2023).

URL-11:https://www.mathworks.com/help/reinforcementlearning/,(Ziyaret tarihi:15 Eylül 2022).

URL-12: https://github.com/DLR-RM/rl-baselines3-zoo / (Ziyaret tarihi:22 Şubat 2023).

Van Hasselt, H., Guez, A., Silver, D. (2016). Deep Reinforcement Learning With Double Q-Learning. *In Proceedings of the AAAI conference on artificial intelligence (Vol. 30, No. 1).* Palo Alto, CA: AAAI Press, March 2016.

Van Seijen, H., & Sutton, R. S. (2015). A Deeper Look At Planning As Learning From Replay. Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, 2015.

Vermorel, J., Mohri, M. (2005). Multi-armed Bandit Algorithms and Empirical Evaluation. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds) *Machine Learning: ECML 2005. ECML 2005. Lecture Notes in Computer Science()*, vol 3720. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11564096_42

Vinyals, O., Babuschkin, I., Czarnecki, W.M., et al. (2019). Grandmaster Level İn Starcraft II Using Multi-Agent Reinforcement Learning. *Nature, 575*, 350–354. https://doi.org/10.1038/s41586-019-1724-z

Wang, L., Cai, Q., Yang, Z., & Wang, Z. (2020). Neural Policy Gradient Methods: Global Optimality And Rates Of Convergence. In International Conference on Learning Representations (ICLR), Montreal, Canada, May 31-June 5 2020.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Frcitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. *33rd International Conference on Machine Learning, ICML 2016, 4*(9), 2939–2947.

Watkins, C. J. C. H. (1989). Learning from Delayed Rewards. Phd Thesis, King's College, Cambridge, UK. Erişim adresi: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

Watkins, C. J. C. H., Dayan, P. (1992). Q-Learning. *Machine Learning, 8*(3), 279-292. doi:10.1007/BF00992698

Werbos, P.J. (1989). Neural Networks For Control And System İdentification. *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, FL, USA, December 13-15 1989, doi: 10.1109/CDC.1989.70114.

Xie, L., Wang, S., Markham, A., & Trigoni, N. (2017). Towards Monocular Vision Based Obstacle Avoidance Through Deep Reinforcement Learning. *arXiv preprint arXiv:1706.09829.*

Zamora, I., Lopez, N. G., Vilches, V. M., Cordero, A. H. (2016). Extending The Openai Gym For Robotics: A Toolkit For Reinforcement Learning Using Ros And Gazebo. *arXiv preprint arXiv:1608.05742.*

Zhang, H.-y., Lin, W.-m., & Chen, A.-x. (2018). Path Planning for the Mobile Robot: A Review. *Symmetry, 10*(10), 450. https://doi.org/10.3390/sym10100450

Zhang, K., Niroui, F., Ficocelli, M., Nejat, G.. (2018). Robot Navigation of Environments with Unknown Rough Terrain Using deep Reinforcement Learning. *In 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (pp. 1-7).* Philadelphia, PA, USA,06-08 August 2018 https://doi.org/10.1109/SSRR.2018.8468643.

Zhang, S., Sutton, R. S. (2017). A Deeper Look At Experience Replay. *arXiv preprint arXiv:1712.01275.*

Zhu, Q., Yan, Y., Xing, Z. (2006). Robot Path Planning Based on Artificial Potential Field Approach with Simulated Annealing. *Sixth International Conference on Intelligent Systems Design and Applications (pp. 622-627),* Jian, China, 16-18 October 2006. https://doi.org/10.1109/ISDA.2006.253908.

KİŞİSEL YAYIN VE ESERLER

Apaydın E., İnner A.B. (2022). Hastane Ortamında Bir Mobil Robotun Navigasyonu İçin Derin Pekiştirmeli Öğrenme Algoritmalarının Karşılaştırılması, *IMASCON Uluslararası Marmara Fen Bilimleri Kongresi*, Kocaeli, Türkiye,13-14 Mayıs 2022.

ÖZGEÇMİŞ

İlkokul, ortaokul ve lise öğrenimini Giresun'da tamamladı. 2011 yılında Kocaeli Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Öğretmenliği bölümünden mezun oldu. 2018 yılında Sakarya Üniversitesi Bilgisayar ve Bilişim Bilimleri Fakültesi Bilgisayar Mühendisliği bölümünden mühendislik tamamlama programı ile mezun oldu.2012 yılından beri Milli Eğitim Bakanlığı'nda Bilişim Teknolojileri Öğretmeni olarak görev yapmaktadır.