T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ROBOT KOLLARININ CİSİMLERİN TUTULABİLİRLİĞİNİ FEDERE DERİN PEKİŞTİRMELİ ÖĞRENME YÖNTEMİYLE ÖĞRENMESİ

YÜKSEK LİSANS TEZİ

Murat Uğur GÜLLE

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

AĞUSTOS, 2022

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ROBOT KOLLARININ CİSİMLERİN TUTULABİLİRLİĞİNİ FEDERE DERİN PEKİŞTİRMELİ ÖĞRENME YÖNTEMİYLE ÖĞRENMESİ

YÜKSEK LİSANS TEZİ

Murat Uğur GÜLLE

(Y1813.010007)

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Peri GÜNEŞ

AĞUSTOS, 2022

ONAY FORMU

ONUR SÖZÜ

Yüksek Lisans Tezi olarak sunduğum “Robot Kollarının Cisimlerin Tutulabilirliğini Federe Derin Pekiştirmeli Öğrenme Yöntemiyle Öğrenmesi” adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya’da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (04/08/2022)

Murat Uğur GÜLLE

ÖNSÖZ

Yapay zekâ algoritmalarına ve bu algoritmalar uygulanılarak kontrol edilebilen mobil robotlara farklı alanlarda olan gereksinim son yıllarda en popüler konuların arasında yer almaktadır. Daha önce yapmış olduğum yüksek lisans çalışmalarında robotlar ve onların yazılımı ile ilgili yaptığım araştırmalar ve çalışmaların sayesinde de "Robot Kollarının Cisimlerin Tutulabilirliğini Federe Derin Pekiştirmeli Öğrenme Yöntemiyle Öğrenmesi" başlıklı tez konusunu seçtim. Bu çalışmanın gerçekleştirilmesindeki destek ve katkılarından dolayı tez danışmanım Dr. Öğr. Üyesi Peri GÜNEŞ'e teşekkürlerimi sunarım. Tez çalışma süresinde her zaman yanımda olan aileme ve eşime teşekkürü bir borç bilirim.

Ağustos 2022 Murat Uğur GÜLLE

ROBOT KOLLARININ CİSİMLERİN TUTULABİLİRLİĞİNİ FEDERE DERİN PEKİŞTİRMELİ ÖĞRENME YÖNTEMİYLE ÖĞRENMESİ

ÖZET

Robot kollar son yıllarda endüstride en çok kullanılan robotik sistemlerden birisi haline gelmiştir. Bu sayede üretim hatları gelişmiş, hızlanmış ve otomatik hale getirilmiştir. Robotların belirlenen cisimleri belirli yerlerden alıp tekrar belirlenen yere taşınması, robotların belirli cisimleri tanıması ve ona göre hareket etmesi, robot kolların eklem bölgelerinin hareket performansları, robot kollarının tork kontrolleri araştırmacıların çalıştığı konuların başında gelmektedir. Robot kolların otomatik hale getirmedeki en büyük problemlerden birisi kullanım yerine göre eğitilmesidir. Geçtiğimiz yıllarda bu eğitim uzun sürmekteyken son yıllarda yöntemlerin ve eğitim algoritmalarının gelişmesiyle birlikte oldukça hızlı gerçekleşmektedir. Önerilen ve geliştirilen algoritmalar bahsedilen problemleri çözmektedir. Son yıllarda çeşitli araştırmacıların önerdiği federe öğrenme yöntemi ile derin pekiştirmeli öğrenme yaklaşımı birleştirilmiştir. Federe öğrenme yaklaşımı alt sistemlerin yani robot kolların aynı anda eğitilmesini sağlamaktadır. Bu eğitim gerçekleştirilirken alt sistemler hatalarını ve sinir ağlarının parametrelerini sunucuya gönderir ve sunucuda tekrar hesaplanan bu ağırlıklar ve hatalar güncellenmiş bir şekilde tekrar alt sistemlere gönderilir. Federe öğrenmenin bir faydası da veri güvenliği olarak karşımıza çıkar. Federe öğrenmede alt sistemler kendi aralarında haberleşmez ve sunucuya herhangi bir veri göndermezler. Tüm transferler sadece alt sistemlerin sinir ağlarının hataları ve ağırlıkları üzerinden gerçekleştirilir. Sunulan eğitim algoritmasının testi federe derin öğrenme yaklaşımıyla eğitilen bir robot kolun eğitim performansı ile karşılaştırılmıştır. Geliştirilen federe derin öğrenme yaklaşımı yaklaşık 6 saat eğitimden sonra %90 seviyesinde öğrenme gerçekleştirmiştir. Ancak sunulan federe derin pekiştirmeli öğrenme algoritması aynı eğitim düzeyine yaklaşık 5 saatte ulaşmıştır. Sunulan eğitim algoritması standart federe derin öğrenme algoritmasından daha iyi performans göstermiş ve daha hızlı sonuç vermiştir. Aynı

zamanda sunulan bu algoritma herhangi bir veriye ihtiyaç duymadan öğrenmeyi sağlar.

Anahtar kelimeler: Robot Kollar, Federe Öğrenme, Federe Derin Öğrenme, Pekiştirmeli Öğrenme, Federe Pekiştirmeli öğrenme

LEARNİNG THE GRİP OF OBJECTS BY THE FEDERATED DEEP REİNFORCEMENT LEARNİNG METHOD OF ROBOT ARMS

ABSTRACT

Robotic arms have become one of the most used electronic systems in the industry in recent years. In this way, production lines have been developed, accelerated and automated.Precise objects identification and moving, the motion performances of the joint areas of the robot arms, and torque controls of the robot arms are some of the favorite subjects that researchers are working on. One of the biggest problems in automating robot arms is they are being trained for needs in certain fields. While this training took a long time in the past years, it has been taking place very quickly with the development of methods and training algorithms in recent years. Suggested and developed algorithms are able to solve the mentioned problems. In recent years, the federated learning method recommended by various researchers has been combined with a deep reinforcement learning approach. Federated learning method enables training of multiple robot arms at the same time. While performing this training, subsystems send their errors and neural network parameters to the server, and these recalculated and updated weights and errors are sent back to the subsystems. Another benefit of federated learning is data security. In federated learning, subsystems do not communicate with each other and do not send any data to the server. All transfers are performed only on the faults and weights of the subsystems of neural networks.The testing of the developed training algorithm is compared with the training performance of a robot arm trained with a federated deep learning approach. The developed federated deep learning approach achieved 90% of learning after approximately 6 hours of training. However, the presented federated deep reinforcement learning algorithm reached the same education level in approximately 5 hours. The presented training algorithm outperformed the standard federated deep learning algorithm and provided faster results. At the same time, this algorithm provides learning without the need for any data.

Keywords: Robot arms, federated learning, federated deep learning, reinforcement learning, federated reinforcement learning

İÇİNDEKİLER

ONUR SÖZÜ ..... **iii**
ÖNSÖZ ..... **v**
ÖZET ..... **vii**
ABSTRACT ..... **ix**
İÇİNDEKİLER ..... **xi**
KISALTMALAR LİSTESİ ..... **xiii**
ÇİZELGELER LİSTESİ ..... **xv**
ŞEKİLLER LİSTESİ ..... **xvii**
I. GİRİŞ **1**
A. Robot Kolların Tarihçesi 1
B. Endüstriyel Robot Kollar 3
1. Kartezyen Robot Kolu 3
2. Silindirik Robot Kolları 4
3. Küresel Robot Kolları 4
4. Scara Robot Kol 5
5. Mafsallı Robot Kollar 5
6. Tutucular 6
II. ÖĞRENME ALGORİTMALARI **9**
A. Yapay Sinir Ağları 9
1. Lineer Olmayan Kısıtsız Optimizasyon 10
2. Optimallik İçin Gerekli Koşullar 12
3. Gradyan Yöntemler 14
4. Gradyan Azalan Algoritması 15
5. Tek Katmanlı Yapay Sinir Ağları 16
6. Tek Katmanlı Yapar Sinir Ağı Eğitimi 16
7. Çok Katmanlı Yapay Sinir Ağları 18
8. Çok Katmanlı Yapay Sinir Ağının İleri Modeli 18
B. Pekiştirmeli Öğrenme 20

1. Rastgele Süreçler ve Markov Süreci 22
2. Markov Karar Süreci 24
3. Markov Karar Süreci Çözüm 28
4. Dinamik Programlama 29
5. Bellman Denklemi 31
6. Pekiştirmeli Öğrenmede Politika Tabanlı Çözümler 32
7. Pekiştirmeli Öğrenmede Değer Tabanlı Çözümler 34
8. Q-öğrenme Algoritması 36
9. Sarsa Algoritması 38
10. Derin Pekiştirmeli Q-öğrenme Algoritması 39
11. Derin Deterministik Politika Gradyan Algoritması 42
C. Federe Öğrenme 44
III. ROBOTLARDA PEKİŞTİRMELİ ÖĞRENME 51
A. Öğrenen Robotlar 52
B. Robotlarda Eylem Seçim Metodları 54
IV. İLGİLİ ÇALIŞMALAR 57
V. YÖNTEM 63
A. V-REP (Coppelia Robotics Virtual Robotic Experiment Platform) 68
B. Eğitim Öncesi 69
C. Mimari 70
D. Ödül İşlevi 70
E. Keşif 71
F. Ajan 72
G. Konum 73
H. Kavrama 74
I. Tensorflow 75
İ. Hiper parametre Optimizasyonu 75
VI. SİMÜLASYON SONUÇLARI VE DEĞERLENDİRMESİ 79
A. Simülasyon Sonuçları ve Kavrama Deneyleri 79
VII. SONUÇ VE GELECEK ÇALIŞMALAR 87
VIII. KAYNAKÇA 89
ÖZGEÇMİŞ 99

KISALTMALAR LİSTESİ

ADP : Uyarlamalı Dinamik Programlama

MDP : Nörodinamik Programlama

ZM : Zamansal Fark

MC : Monte Carlo

DP : Dinamik Programlama

PDF : Rastgele Sürece İlişkin Ortak Olasılık Yoğunluk Fonksiyonu

JPDF : Rastgele Sürece İlişkin Basit Olasılık Yoğunluk Fonksiyonu

IID : Bağımsız ve Özdeş Dağıtılmış Rastgele Değişken

ADHDP : Eyleme Bağlı Buluşsal Dinamik Programlama

FedRL : Federe Pekiştirmeli Öğrenme

MLP : Çok Katmanlı Algılayıcı

MEC : Mobil Uç Bulutu

NAF : Normalleştirilmiş Avantaj Fonksiyonu

NAC : Doğal Aktör Kritik

İHA : İnsansız Hava Aracı

CNN : Evrişimsel Sinir Ağı

SGD : Olasılıksal Dereceli Azalma

FEDAVG : Federe Ortalama

API : Uygulama Programlama Arayüzü

GPU : Grafik İşlem Birimi

CPU : Merkezi İşlem Birimi

SAC : Yumuşak Aktör Kritik

VRAM : Video Rastgele Erişimli Bellek

RAM : Rastgele Erişimli Bellek

DFQN : Derin Federe Q Ağı

FDRL : Derin Federe Pekiştirmeli Öğrenme

RL : Pekiştirmeli Öğrenme

DRL : Derin Pekiştirmeli Öğrenme

DL : Derin Öğrenme

V-REP : Araç Hazırlık Geliştirme Programı

ÇİZELGELER LİSTESİ

Çizelge 1 Hessian Tablosu 14
Çizelge 2 Örnek Veri Seti 17
Çizelge 3 rexp = 7 ayarlandığı ve her bölümün en fazla 100 zaman adımına sahip olduğu kavrama görevi için bu çalışmada kullanılan ödül fonksiyonuna genel bakış. 71
Çizelge 4 Robotik Kol İçin Verilen D-H Parametreleri 71
Çizelge 5 Oluşturulan Sinir Ağı İçin Hiper Parametreleri 77
Çizelge 6 Hedef pozisyonuna göre nihai pozisyonun standart sapması ile ortalama başarı oranı ve ortalama mesafe. 700 denemelik ön eğitim kullanılarak ortalama 10 deneme ve toplu öğrenme için bir arabellek ve iki aracı kullanılarak 300 deneme için fizik simülatörü kullanılarak eğitilmiştir. 6 eklem ve 3 parmak kullanılmıştır. 82
Çizelge 7 Hedef Pozisyonuna göre nihai pozisyonun standart sapması ile ortalama başarı oranı ve ortalama mesafe. Ortalama 10 deneme, 1500 deneme için fizik simülatörü kullanılarak eğitilmiş, toplu öğrenme için bir arabellek ve iki aracı; 6 eklem ve 3 parmak kullanılmıştır. 82
Çizelge 8 Aynı türden birden çok robot kol cihazının dinamikleri için homojenlik testinin sonuçları. 84

ŞEKİLLER LİSTESİ

Şekil 1 Von Kempelen'in Turk satranç oyuncu robotu 2
Şekil 2 Unimate robot kolu 3
Şekil 3 Kartezyen robot kolu 4
Şekil 4 Silindirik Robot Kolu 4
Şekil 5 Küresel Robot Kolu 5
Şekil 6 Scara Robot Kol 5
Şekil 7 Mafsallı Robot Kolları 6
Şekil 8 Elektrikli Tutucu 7
Şekil 9 Pünomatik Tutucu 7
Şekil 10 Makine Öğrenmesi Algoritma Ağacı 9
Şekil 11 Tam Bağımlı Yapay Sinir Ağı 10
Şekil 12 Konveks Olmayan $f(\boldsymbol{x})$ fonksiyonu 12
Şekil 13 Gradyan Azalan Algoritması 15
Şekil 14 Gradyan Azalan Algoritması Grafiği 15
Şekil 15 Tek Katmanlı Yapay Sinir Ağı 16
Şekil 16 Yapay Sinir Ağlarının Gradyan Azalan Algoritmasıyla Eğitimi Algoritması 18
Şekil 17 Çok Katmanlı Yapay Sinir Ağı 18
Şekil 18 Çok Katmanlı Çok Girişli Çok Çıkışlı Yapay Sinir Ağı 19
Şekil 19 Adam Algoritması 20
Şekil 20 Çeşitli Pekiştirmeli Öğrenme Algoritmaları Grafiği 21
Şekil 21 Markov Süreci İçin Örnek Bir Akış 23
Şekil 22 Optimal Yol Problemi Örnek Grafik 29
Şekil 23 Optimal Yol Problemi Çelişkisi 30
Şekil 24 Dinamik Programlama Modeli Genel Çalışma Prensibi 31
Şekil 25 Basit Politika Arama Algoritması 33
Şekil 26 Politika Gradyanı ve Takviye Algoritması 34
Şekil 27 Zamansal Fark Algoritması 35

Şekil 28 Q-Öğrenme Algoritması 38
Şekil 29 Sarsa Algoritması 39
Şekil 30 Derin Q-Öğrenme Algoritması Akış Diyagramı 40
Şekil 31 Derin Q-Öğrenme Algoritması 42
Şekil 32 Derin Deterministik Politika Gradyanı Algoritması 44
Şekil 33 Kullanılacak Federe Derin Öğrenme Algoritmasının Sözde Kodu (Yerel) . 49
Şekil 34 Kullanılacak Federe Derin Öğrenme Algoritması Sözde Kodu (Sunucu) ... 48
Şekil 35 Akıllı Robot Ortam İlişkisi 52
Şekil 36 Akıllı Robot Öğrenme Süreci 53
Şekil 37 Ödülün Ortamdan Durum-Eylem İkilisi Sonrası Nasıl Geldiğinin İlişkisi .. 54
Şekil 38 Örnek Bir Robot Kol 63
Şekil 39 Kullanılacak Federe Derin Öğrenme Algoritması Sözde Kodu (Sunucu) ... 64
Şekil 40 Kullanılacak Federe Derin Öğrenme Algoritmasının Sözde Kodu (Yerel) . 65
Şekil 41 Kullanılacak Federe Derin Pekiştirmeli Öğrenme Algoritması (İşçi) 66
Şekil 42 Kullanılacak Federe Derin Pekiştirmeli Öğrenme Algoritması (Sunucu) ... 67
Şekil 43 Robot Kollarının Tutucularının Kullanabileceği Katezyen Uzay 68
Şekil 44 Robot Kol 1'in derin pekiştirmeli öğrenme sonuçları 79
Şekil 45 Robot Kol 2'nin derin pekiştirmeli öğrenme sonuçları 80
Şekil 46 Robot Kol 3'ün derin pekiştirmeli öğrenme sonuçları 80
Şekil 47 Federe derin öğrenme yöntemi ile öğrenme 80
Şekil 48 Federe derin pekiştirmeli öğrenme yöntemi ile öğrenme 81
Şekil 49 Federe derin pekiştirmeli öğrenme 2 ve 3 robot kolu ile öğrenme sonucunun karşılaştırması 81

I. GİRİŞ

Son yıllarda gelişmiş ve gelişmekte olan ülkelerin üretim bantlarına bakıldığında otomasyon sistemlerinin önemli ölçüde kullanıldığı göze çarpmaktadır. Üretim sistemlerindeki otomasyon süreci önem kazandıkça endüstriyel robot kolların kullanımı ve geliştirilmesi önemli bir konu haline gelmektedir.

Günümüzde insanlar fiziksel yapılarının getirdiği zafiyetlerden dolayı, gücünün yetmediği koşullarda kullanmak üzere çeşitli makineler geliştirmiştir. İlk çağlarda yeterince gelişmiş ve kullanım alanları çok sınırlı olan bu makineler, gelişen teknoloji ile birlikte insanlar tarafından geliştirilmiş ve insanların fiziksel yapılarına erişebilecek seviyede makineler ortaya çıkmıştır. Orta çağlarda geliştirilen ilk makineler insan yardımı ile çalışmaktaydı, ancak zaman içerisinde geliştirilerek herhangi bir insan müdahalesine gerek kalmayacak şekilde çalışır hale getirilmiştir.

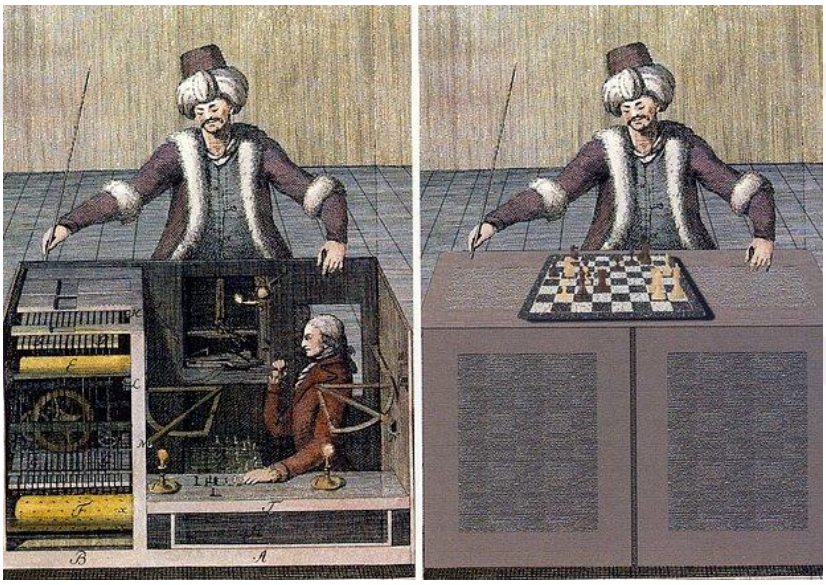
Sanayide kullanılmak için geliştirilen birçok robot ve robot kol bulunmaktadır. Bu robotlar ile birlikte genel itibariyle, üretim maliyetini düşürerek ve insan gücünü ortadan kaldırarak daha hızlı ve daha kaliteli üretim yapılabilmektedir. Ayrıca insan sağlığının el vermediği (afetler, nükleer enerji, yüksek ısı, dar alanlar vb.) durumlarda kullanılmaktadır.

A. Robot Kolların Tarihçesi

Robot fikrini ilk olarak Da Vinci 1495'te dört serbestlik dereceli, güç ve programlanabilirlik sağlan, analog yerleşik kontrolörlü bir robot kol tasarlayarak ortaya koymuştur (Rosheim, 2022). Bu robot iki bağımsız sistemden oluşmaktadır. Alt eklemler üç serbestlik derecesine sahiptir: Kalçalar, dizler, ayak bilekleri ve bacaklar. Üst eklemler ise dört serbestlik derecesine sahiptir: Kollar, dirsekler, bilekler ve omuzlar (Rosheim, 2022).

Uzun yıllar sonra Wolfgang Von Kempelen'in Türk olarak adlandırılan satranç oyuncusu robotu ortaya çıkmıştır (Şekil 1) (Standage, 2002). Robot 1769 yılında Kraliçe Maria Threse için üretilmiştir. Türk, satranç tahtasının altındaki dolabın içerisinde gizlenmiş bir insan tarafından kullanılmaktadır. Otomat, satranç

oyuncusunun kolunu otomatik hale getiren bir mekanizmaya sahiptir. Satranç oyuncusu, Türk-Osmanlı kıyafeti gitmiş ahşap bir sandığın arkasına oturan bir kukladır. Bu kuklanın başı, gözleri ve kolları hareket edebilmektedir. Aynı zamanda kuklanın sol kolu ve eli muhteşem bir şekilde düzenlenmiştir. Kol mekaniği, oyunların insan kontrollü olduğunu bilenlerin verdiği isim olan "yönetici" tarafından kontrol edilmektedir. Uzuv önce kaldırılır, ardından sol el, hareket ettirilmesi istenen satranç taşının üzerine ortalanır. Kol taşa doğru indirildikten sonra çark çevrildiğinde Türk'ün elindeki kaldıracın satranç taşını kaldırılması sağlanır. Otomatın kolları tahtadan oluşmaktadır ve satranç karşılaşması sırasında taşların daha kolay kavranması için el bir eldivenin içerisine yerleştirilmiştir.

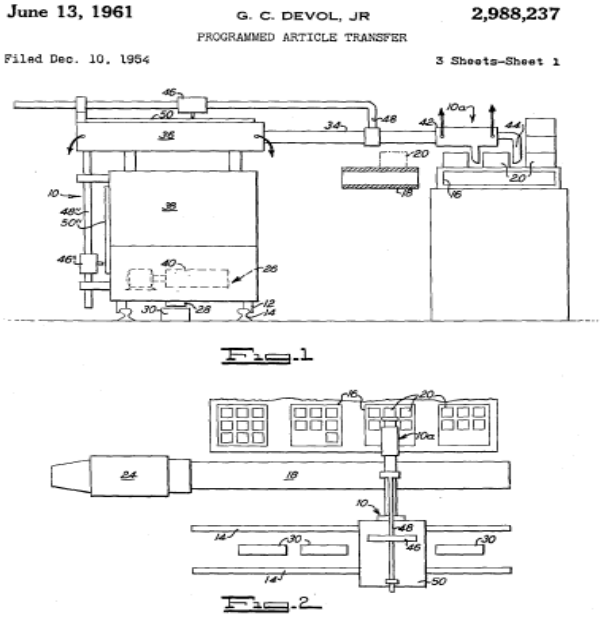


Şekil 1 Von Kempelen'in Turk satranç oyuncu robotu

İlk "konum kontrol aparatı" 1938'de Williard Pollard tarafından geliştirildi ve patentlendi (ABD Patent No. 2,286,571, 1942).

Bu beş serbestlik derecesine ve bir elektrik kontrol sistemine sahip sprey cilalama robot koludur (ABD Patent No. 2,286,571, 1942). Harold A. Roselund başka bir sprey cilalama robot kolu geliştirdi (ABD Patent No. 4,344,108, 1944). Her iki kol da kendi zamanları için çok karmaşıktır ve elektronik kontrol sistemleri, onları kullanılabilir hale getirmek için yeterince geçişmiş değildir. Modern robotik çağı, 1930'ların sonlarında geliştirilen, az bilinen bu iki robot kolun ortaya çıkmasıyla başlamıştır.

Unimate şirketi, ilk robot kolunu 1962 geliştirdi (Şekil 2) (Ellis, 2000). Kol George Devol tarafından icat edilmiştir. Geliştirilen bu robot kol ilk endüstriyel kol olarak kabul edilmektedir.



Şekil 2 Unimate robot kolu

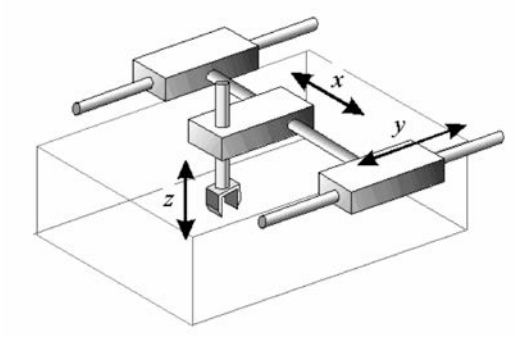
ISO 8373 tarafından tanımlanan sanayi robotu tanımı şu şekildedir: Üç veya daha fazla programlanabilir ekseni olan, otomatik kontrollü, çok amaçlı, bir yerde sabit duran veya tekerlekleri olan endüstriyel uygulamalarda kullanılan manipülatördür (Yücel, 1991).

B. Endüstriyel Robot Kollar

1. Kartezyen Robot Kolu

Kartezyen Robotların sadece tutma ve taşıma yetenekleri bulunmaktadır ve 3 eksende hareket etme kabiliyetlerine sahiptirler. Basit bir yapıya sahip olduklarından dolayı hareketlerinin ve kontrollerinin planlanması yeterince kolaydır. Kartezyen robotlarda; pozisyonların hesaplamaları, robotun bulunduğu pozisyon ve mafsalların aynı yerde bulunmasından dolayı kolaydır (Şekil 3).

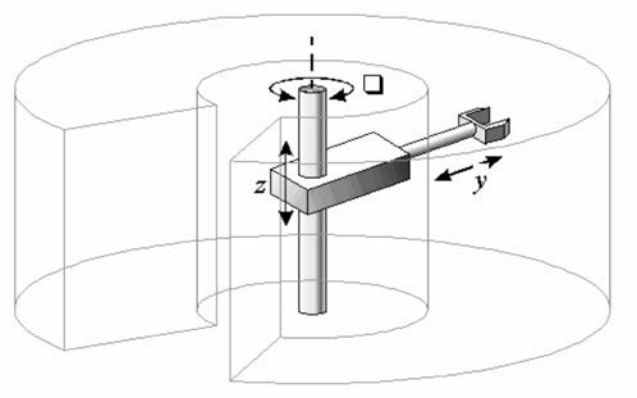
Kartezyen robotlar, eğilme ve bükülme işlemlerini gerçekleştiremez. Yük taşıma işlemleri için kullanılmakta ve genellikle insan gücünü aşan yüklerin taşınmasında kullanılmaktadır. Bu nedenden dolayı genellikle fabrikalarda yükleme ve boşaltma işlemlerinde kullanıldığından fabrikaların tavan bölümlerine monte edilmektedir.



Şekil 3 Kartezyen robot kolu

2. Silindirik Robot Kolları

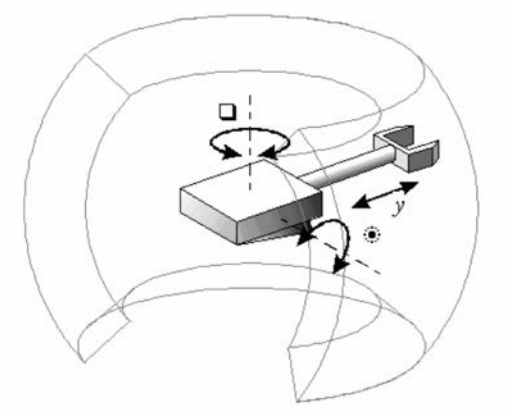
Silindirik robot kolları kendi etrafında dönebilecek şekilde geliştirilmiş ve aynı kartezyen robot kolunda olduğu gibi 3 eksende hareket etme kabiliyetine sahiptirler. Şekil 4'te görüldüğü üzere esnek yapıda değil ve aynı zamanda Kartezyen robot kollarından daha fazla alana hareket edebilmektedirler. Robot kolunun çalışabileceği alan silindirik koordinat sisteminde hareket edecek kolların uzunluğuna göre değişmektedir.



Şekil 4 Silindirik Robot Kolu

3. Küresel Robot Kolları

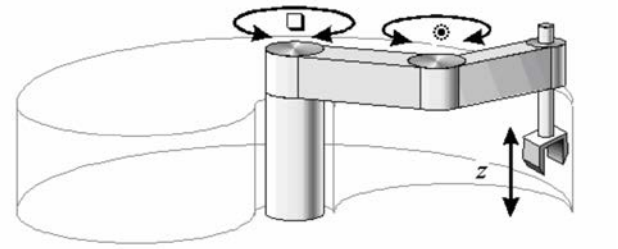
Küresel robot kolları omuz, dirsek ve gövdeden oluşmaktadır. Gövde ve omuz kendi etrafında dönebilir, kol ise dirsek bölümünden uzayıp kısalabilir. Hareket çerçevesi şekil 5'te gösterilmiştir. Küresel robot kolları silindirik bir dönme sistemine sahiptir. Yapıları genel itibariyle kartezyen ve silindirik robot kollarına göre daha karmaşıktır. Çalışma alanı kolların dirseklerden uzayıp kısalma boyutuna göre değişmektedir. Küresel robot kolları sarkaç robot kol olarak da isimlendirilebilmektedir. Genel itibariyle endüstriye kaynak yapımları ve yapıştırma işlemlerinde kullanılmaktadır.



Şekil 5 Küresel Robot Kolu

4. Scara Robot Kol

Scara robot kol diğer robot kollardan farklı olarak eklem bölgesi yerine elektrik motoru, yukarı ve aşağı hareket edebilen bir koldan oluşmaktadır. Eklemler elektrik motorlarından destek alarak kendi etrafında dönebilirler. Tutucu ağız sadece z ekseninde yani yukarı ve aşağı yönde hareket edebildiğinden yeterinde hız ve performans sağlamaktadır. Scara robot kol da aynı silindirik robot kolunda olduğu gibi orta eksende kendi etrafında dönebilmektedir (Şekil 6). Kolun programlanması kolay ve hızlı hareket kabiliyeti olmasından dolayı endüstride elektronik sanayinde, elektronik kartlara yapılacak olan ekleme ve lehim işlemlerinde kullanılmaktadır. Scara robot kol hali hazırda endüstride en çok kullanılan robot konumundadır.



Şekil 6 Scara Robot Kol

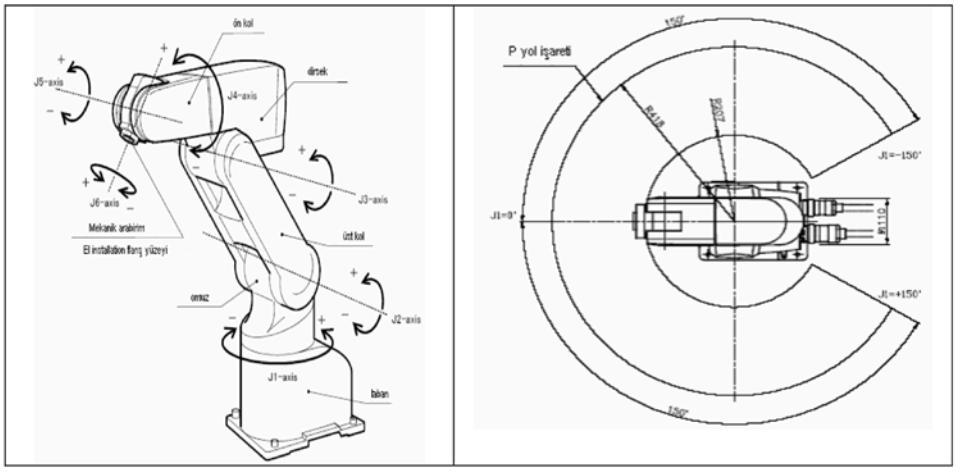
5. Mafsallı Robot Kollar

Mafsallı robot kollar insan koluna en yakın şekilde hareket edebilen robot koldur. Öncesinde açıklanan robot kollarının hareket yeteneklerinin yeterince iyi olmamasından dolayı endüstride kullanmak amacıyla eklem sayısı 7'e kadar çıkabilecek mafsallı robot kollar geliştirilmiştir.

Hareket kabiliyeti en iyi olan robot koldur. Kol belirli bir alana monte edildikten sonra, kol X,Y ve Z ekseninde üç boyutlu hareket yapılabilmektedir. Ancak öncesinde bahsedilen robotlardan daha karmaşık olduğundan dolayı programlanması da diğer robot kollardan daha zordur.

Her eklem bölgesi şekil 7'de gösterildiği gibi programlandığı şekilde rahat hareket edebilmektedir. Bu da robotun istenen noktaya daha hızlı ve güvenli bir şekilde ulaşmasını sağlar. Yapılacak olan uygulamanın şekline göre robot kolunun eklem ve eksen sayısının tercihi yapılmalıdır.

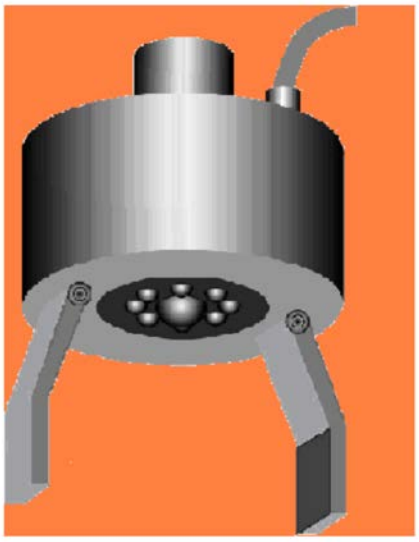
Hali hazırdaki çalışmada mafsallı robot kollar üzerinde bir eğitim çalışması gerçekleştirilmiştir.



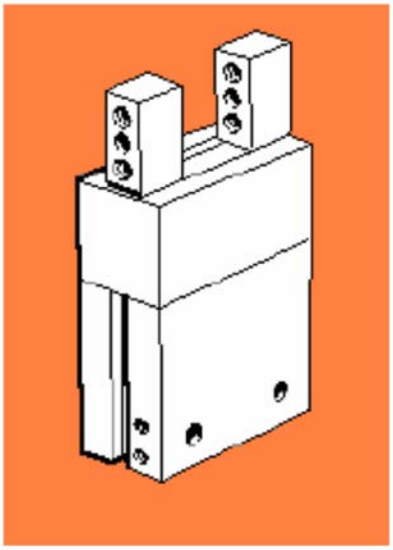
Şekil 7 Mafsallı Robot Kolları

6. Tutucular

Tutucular robotun bir nesneyi tutması amacıyla çeşitli büyüklükte ve biçimlerde tasarlanmış eklem bölgesidir. Tutma işlemi robotun üzerine işlem gerçekleştireceği nesneye bağlı olacak şekilde geliştirilmiştir. Şekil 8'de elektrikli tutucu ve şekil 9'da pünomatik tutucunun şekli görülmektedir. Çalışmada elektrikli tutucu kullanılacaktır.



Şekil 8 Elektrikli Tutucu

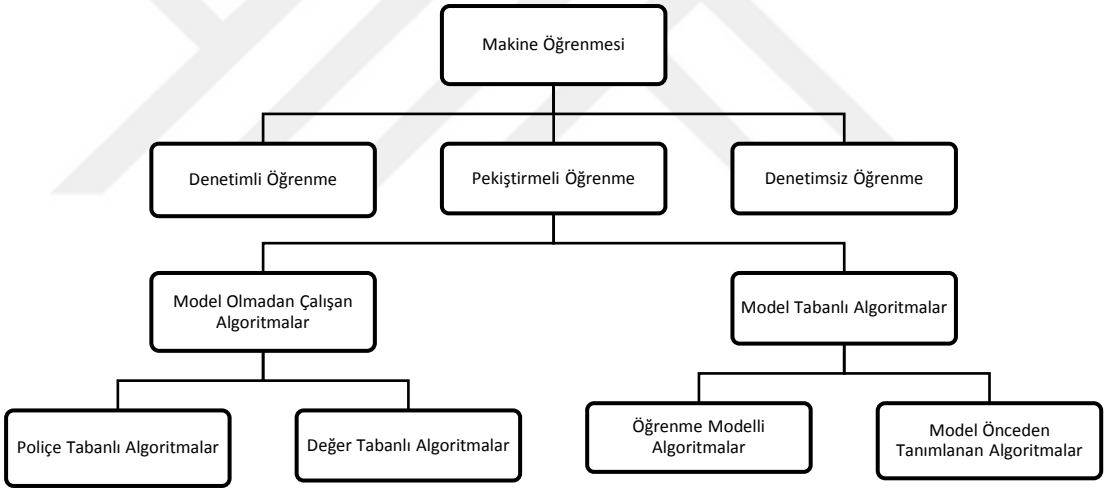


Şekil 9 Pünomatik Tutucu

II. ÖĞRENME ALGORİTMALARI

Bu bölümde robotların çeşitli yapay zeka yöntemleriyle eğitilmesi, bu yapay zeka yöntemlerinin içerikleri ve çalışma mekanizmalarındaki farklardan bahsedilecektir. Sonrasında çeşitli federe öğrenme yaklaşımlarından bahsedilecektir.

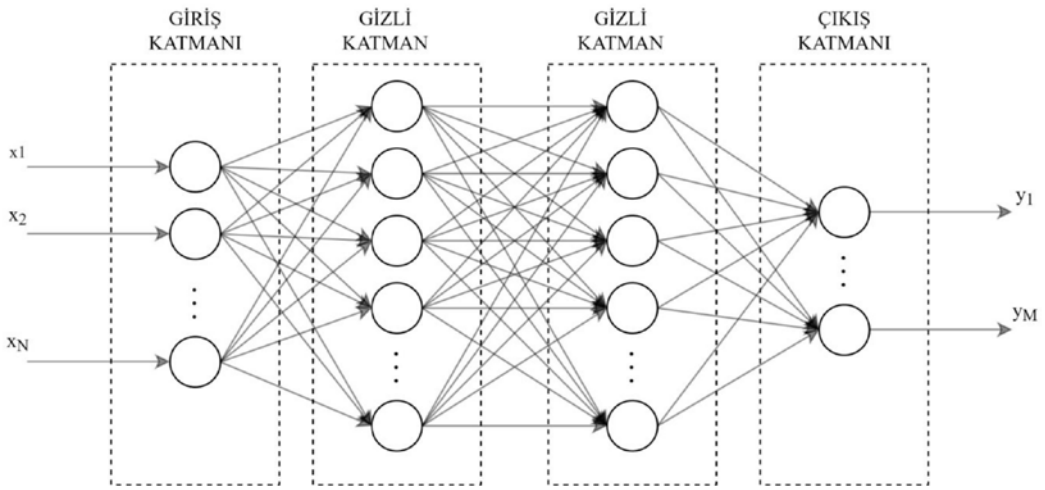
Son yıllarda makine öğrenimi yöntemleri gerçek ya da simülasyon ortamlarında çeşitli problemleri çözmek için yaygın bir biçimde kullanılmaktadır (El-Ghazali, 2020). Makine öğrenimi algoritmaları şekil 10'da da görüldüğü üzere 3 farklı bölüme ayrılmıştır. Bu bölümler; Denetimli Öğrenme, Denetimsiz Öğrenme ve Pekiştirmeli Öğrenme olarak adlandırılmıştır.



Şekil 10 Makine Öğrenmesi Algoritma Ağacı

A. Yapay Sinir Ağları

Makine öğrenimi yöntemleri yapay zekanın alt kolu olarak karşımıza çıkmaktadır. Araştırmacılar son yıllarda yaptıkları geliştirmelerle makine öğreniminin alt kolu olan yapay sinir ağlarını ortaya koymuşlardır. Yapay sinir ağları insan beynindeki nöronların matematiksel bir karşılığı olarak tasarlanmıştır. Klasik bir tam bağımlı yapay sinir ağının şekli şekil 11'de görüldüğü gibidir (Hagan, 2014).



Şekil 11 Tam Bağımlı Yapay Sinir Ağı

Şekil 11'den de görüldüğü üzere klasik bir yapay sinir ağında 3 katman bulunmaktadır. Giriş katmanı girdi verilerini alan ve sonraki katmana taşıyan katman olarak nitelendirilebilir. Giriş katmandaki nöronların sayısı verilerin sınıf sayısına göre belirlenir. Veriler giriş katmanına eklendikten sonra matematiksel bir fonksiyon ile hesaplaması yapıldıktan sonra gizli katmanlara iletilir. Bir yapay sinir ağında birden fazla gizli katman varsa derin (deep) yapay sinir ağı olarak isimlendirilir. Çalışmada çok katmanlı yapay sinir ağı ortaya konan pekiştirmeli öğrenme yönteminde kullanılacaktır.

Yapay sinir ağında herhangi bir modelin eğitilebilmesi için bir optimizasyon algoritması kullanılmak zorundadır. Optimizasyon kısaca yapılan işlemin eniyilenmesi olarak da adlandırılabilir. Çalışma kapsamında yapay sinir ağını oluştururken lineer olmayan kısıtsız sayısal optimizasyon kullanılacaktır.

1. Lineer Olmayan Kısıtsız Optimizasyon

Optimizasyon teknikleri günümüz makine öğrenmesi yöntemlerinde oldukça sık kullanılmaktadır. Optimizasyon yöntemleri; matematiksel programlama, sezgisel optimizasyon ve sayısal optimizasyon şeklinde alt dallara ayrılabilir. En sade haliyle optimizasyon problemi denklem 1'de belirtilmiştir (İplikçi, 2017).

$$\begin{aligned} &\min_{x} f\,(x) \\ &g_i(x) = 0 i \,\epsilon\, E \\ &g_i(x) \leq 0 i \,\epsilon\, J \end{aligned} \qquad (1)$$

Denklem 1'de $f(\boldsymbol{x})$ optimize edilmek istenen amaç fonksiyonunu temsil etmektedir. $\boldsymbol{x}$ tasarım değişkenlerini gösterir. Aynı zamanda $\boldsymbol{x} \in \mathrm{R}^n$ olmak üzere $f(\boldsymbol{x})$: $\mathrm{R}^n \rightarrow \mathrm{R}$ tanımlıdır. $g_i(\boldsymbol{x}) = 0$ değişkenlerin eşitlik kısıtını $g_i(\boldsymbol{x}) \leq 0$ ise değişkenin eşitsizlik kısıtını temsil etmektedir. E ve J sırasıyla eşitlik kısıtları kümesini ve eşitsizlik kısıtları kümesini temsil etmektedirler.

Denklem 1'de ifade edilen optimizasyon problemi, parametrelerine göre doğrusal olmayan bir fonksiyon olduğundan dolayı lineer olmayan kısıtlı optimizasyon problemi olarak adlandırılmıştır.

Kısıtsız lineer olmayan optimizasyon problemleri için denklem 2'de ki fonksiyon kullanılmaktadır.

$$\min_x f\,(x) \tag{2}$$

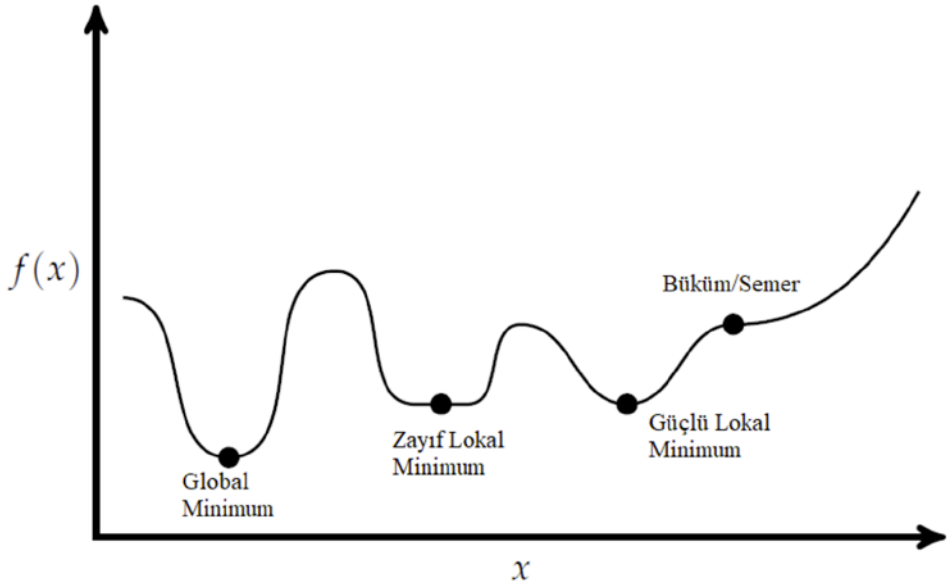
Denklem 2'de $f(\boldsymbol{x})$: $\mathrm{R}^n \rightarrow \mathrm{R}$ koşuluyla birlikte sürekli ve türevlenebilir olmak zorundadır. $f(\boldsymbol{x})$ fonksiyonu için optimal değer her bir $\boldsymbol{x}^*$ değeri için $e > 0$ olduğunda denklem 3 sağlanıyorsa bu durumda $\boldsymbol{x}^*$ yerel minimumu ifade etmektedir.

$$f(x^*) \leq f(x) \qquad \forall x \parallel x - x^* \parallel < e \tag{3}$$

Eğer $\boldsymbol{x}^*$ parametresi için denklem 4 sağlanıyorsa, nokta global minimum noktasını ifade etmektedir.

$$f(x^*) \leq f(x) \qquad \forall x \in \Re^n \tag{4}$$

Denklem 3 ve 4'te ki ifadelerdeki eşitsizlik değeri ($\leq \rightarrow <$) kaldırıldığında kesin yerel ve kesin global minimum noktaları ortaya çıkmaktadır. Optimize edilmek istenen $f(\boldsymbol{x})$ fonksiyonu konveks olmayan bir biçimde karşımıza çıkabilir ve fonksiyonda birden fazla yerel minimum ve maksimum noktaları bulunabilir. Konveks olmayan $f(\boldsymbol{x})$ fonksiyonunun grafiği aşağıdaki gibidir (Şekil 12).



Şekil 12 Konveks Olmayan $f(\boldsymbol{x})$ fonksiyonu

2. Optimallik İçin Gerekli Koşullar

Denklem 2'nin optimum noktasını bulabilmek için $f(\boldsymbol{x})$ fonksiyonunun $\Delta \boldsymbol{x}$ değişimiyle azalma sürecinin incelenmesi ve fonksiyonun davranışlarının analiz edilmesi gerekmektedir. Denklem 2'de $f(\boldsymbol{x})$ fonksiyonu $\boldsymbol{x}^*$ noktasında $f(\boldsymbol{x}^*)$ değerini alıyorken $\boldsymbol{x}^* + \Delta \boldsymbol{x}^*$ noktasında hangi değeri aldığının Taylor açılımı ile incelenmelidir. Denklem 2'nin birinci dereceden Taylor açılımı denklem 5'te ki gibidir

$$f(x^* + \Delta x) = f(x^*) + \nabla f(x^*)' \Delta x \, HOT \qquad (5)$$

Denklem 5'te HOT Taylor açılımında yüksek dereceden terimleri ifade eder. Eğer Denklem 5'te $HOT = 0$ alınırsa denklem 6 elde edilmektedir.

$$f(x^* + \Delta x) - f(x^*) \approx \nabla f(x^*)' \Delta x \qquad (6)$$

Denklem 6'ya eğer ikinci dereceden taylor açılımı uygulanırsa Denklem 7 elde edilecektir

$$f(x^* + \Delta x) - f(x^*) \approx \nabla f(x^*)' \Delta x + \frac{1}{2} \Delta x' \Delta^2 f(x^*) \Delta x \qquad (7)$$

Eğer $\boldsymbol{x}^*$ kısıtsız yerel minimum noktası ise Taylor açılımının birinci türevindeki $\Delta\boldsymbol{x}$ noktasının pozitif olması gerekmektedir. Bu durum denklem 8'de gösterilmiştir

$$\nabla f(x^*)'\Delta x = \Sigma_{i=1}^{n} \frac{\partial f(x^*)}{\partial x_i} \Delta x_i \geq 0 \tag{8}$$

Denklem 8'in optimallik için birinci şartı, denklem 9'daki gibidir.

$$\nabla f(x^*) = 0 \tag{9}$$

Benzer şekilde optimallik için ikinci dereceden koşul için denklem 7 incelenerek denklem 10 elde edilir.

$$\nabla f(x^*)'\Delta x + \frac{1}{2} \Delta x' \nabla^2 f(x^*) \Delta x \geq 0 \tag{10}$$

Denklem 9 ve denklem 10 birleştirilerek denklem 11 elde edilir.

$$\Delta x' \nabla^2 f(x^*) \Delta x \geq 0 \tag{11}$$

Denklem 11'de $\forall \boldsymbol{x}$ değeri için eşitlik sağlanacağından dolayı, eşitliğin oluşması için ikinci dereceden koşul denklem 12'deki şekilde elde edilir

$$\nabla^2 f(x^*) \geq 0 \tag{12}$$

Denklem 12'de ∇^2 ifadesi amaç fonksiyonunun hessian matrisini oluşturmaktadır. $f(.)$: $R^n \rightarrow R$ olmak üzere amaç fonksiyonunun hessian ve gradyan matrisi denklem 13'deki gibi hesaplanır.

$$\nabla^2 f(\boldsymbol{x}) = H = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial {x_1}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{bmatrix} \quad (13)$$

$$\nabla f(\boldsymbol{x}) = G = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Denklem 9'da bulunan $\boldsymbol{x}^*$ kritik noktanın yorumu denklem 11 ile yapılmalıdır. Böylece uygun hessian matrisinin durumuna göre $\boldsymbol{x}^*$ noktasının durumu değişmektedir. Bu durum çizelge 1'de görüldüğü gibidir.

Çizelge 1 Hessian Tablosu

Olasılık	$\boldsymbol{x}^*$ Noktasının Durumu	Açıklama
1	$\nabla^2 f(\boldsymbol{x}^*) \geq 0$	$\boldsymbol{x}^*$ noktası yerel minimum
2	$\nabla^2 f(\boldsymbol{x}^*) > 0$	$\boldsymbol{x}^*$ noktası kesin yerel minimum
3	$\nabla^2 f(\boldsymbol{x}^*) \leq 0$	$\boldsymbol{x}^*$ noktası yerel maksimum
4	$\nabla^2 f(\boldsymbol{x}^*) < 0$	$\boldsymbol{x}^*$noktası kesin yerel maksimum
5	$\nabla^2 f(\boldsymbol{x}^*) = 0$	$\boldsymbol{x}^*$semer noktası

3. Gradyan Yöntemler

Denklem 3 amaç fonksiyonunu minimum yapan $\boldsymbol{x}^*$ parametresinin değerini bulmak için $f(\boldsymbol{x})$ fonksiyonunun birinci türevini $\nabla f(\boldsymbol{x})$ gradyan vektörü ve ikinci türevi olan $\nabla^2 f(\boldsymbol{x})$ hessian matrisini kullanmaktadır. Belirlenen bir $\boldsymbol{x}_0$ başlangıç noktasından $\boldsymbol{x}^*$ noktasının bulunması için arttırılabilir bir şekilde parametrelerin güncellenmesi gerekmektedir. Bu yöntem denklem 14'te gösterilmiştir.

$$x_{i+1} = x_i + s_i p_i \quad (14)$$

Denklemde $\boldsymbol{p_i}$ iniş yönünü ifade etmekte ve birinci dereceden yöntemler için fonksiyonun gradyan vektörünün tersine eşittir ve Denklem 15'te gösterilmiştir.

$$p_i = -\nabla f(x_i) \quad (15)$$

Denklem 14'te s_i adım aralığını belirtmektedir. Bu şekilde denklem 14'te ki güncelleme ile denklem 2'yi minimum yapan noktalar belirlenir. Denklem 14'teki ifadede ise $\boldsymbol{p}_i$ amaç fonksiyonunun, birinci dereceden türevini ya da ikinci dereceden türevini temsil etmektedir (İplikçi, 2017).

4. Gradyan Azalan Algoritması

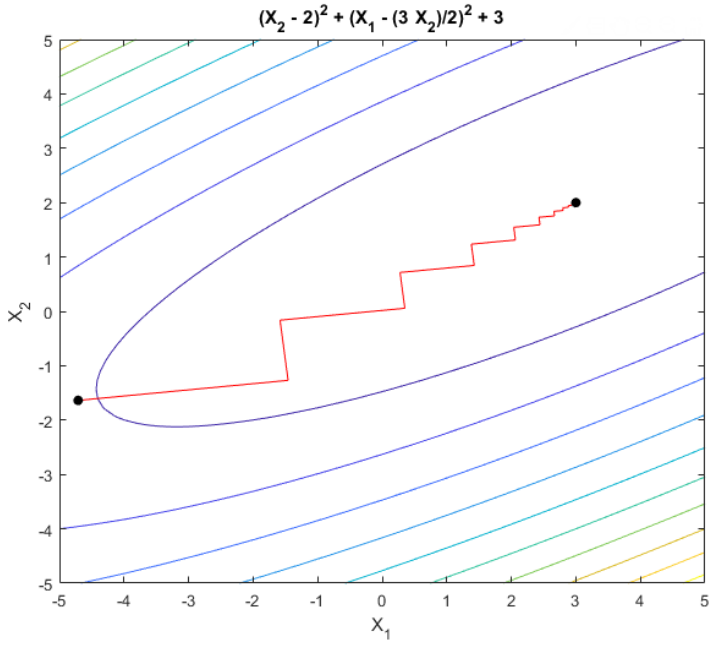
Gradyan azalan algoritması $\boldsymbol{x}^*$ parametre değerini bulmak için amaç fonksiyonunun gradyan vektörünü kullanmaktadır (Nocedal and Wright, 2006).
Gradyan azalan algoritmasının sözde kodu şekil 13'te gösterilmiştir.

ALGORİTMA : Gradyan Azalan Algoritması

1 : $\boldsymbol{x}_0, s$ *Parametrelerin başlangıç noktasını belirle*
2 : $i \leftarrow 0$
3 : $\boldsymbol{x}_i$ *noktasında* $\nabla f(\boldsymbol{x}_i)$ *hesapla*
4 : $\boldsymbol{p}_i = -\nabla f(\boldsymbol{x}_i)$ *İlerleme yönü belirle*
5 : $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + s\boldsymbol{p}_i$ *Parametreleri güncelle*
6 : $i \leftarrow i + 1$
7 : *Eğer* $\| \nabla f(\boldsymbol{x}_i) \| < 1e-6$ *Algoritmayı sonlandır*
8 : *Değilse 3. adıma geri dön*

Şekil 13 Gradyan Azalan Algoritması

Örnek gradyan azalan algoritmasının grafiği şekil 14'te gösterilmiştir.



Şekil 14 Gradyan Azalan Algoritması Grafiği

5. Tek Katmanlı Yapay Sinir Ağları

Tek katmanlı yapay sinir ağları bir gizli katmanı olan ağlardır ve şekil 15'te görülmektedir (Goodfellow et al., 2017).

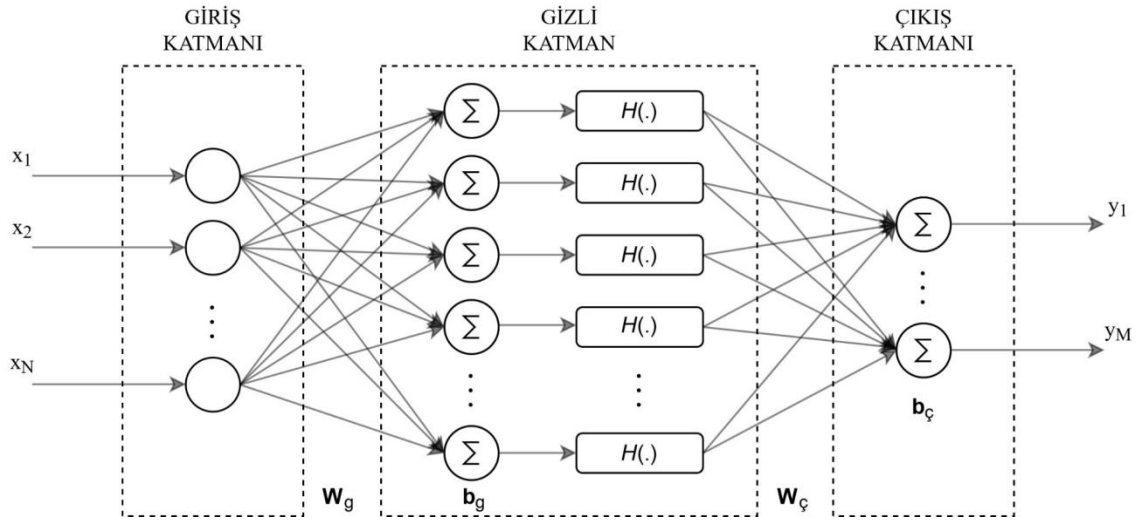
Tek katmanlı sinir ağlarının ileri modeli denklem 16'da gösterilmiştir.

$$\hat{y}_j = W_ç\, H\left(W_g x_i + b_g\right) + b_ç \qquad i = 1,2,\dots,Nj = 1,2,\dots,M \tag{16}$$

Denklem 16'da $H(.)$ aktivasyon fonksiyonu olarak adlandırılır. Örnek bir aktivasyon fonksiyonu denklem 17'da gösterilmiştir.

$$H(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{17}$$

Araştırmacılar birçok aktivasyon fonksiyonu önermişlerdir ancak bu çalışma kapsamında denklem 17'de ki aktivasyon fonksiyonu kullanılacaktır.



Şekil 15 Tek Katmanlı Yapay Sinir Ağı

6. Tek Katmanlı Yapar Sinir Ağı Eğitimi

Denklem 16'de belirtilen $\boldsymbol{W_g}$,$\boldsymbol{b_g}$,$\boldsymbol{W_ç}$,$\boldsymbol{b_ç}$ sinir ağının parametrelerini temsil ederler. Yapay sinir ağlarında modelin parametreleri rastgele bir değerden başlanarak döngü boyunca güncellenir. Bu döngü sinir ağları en iyi modeli elde edene kadar devam eder. Güncelleme işlemi şekil 13'te önerilen gradyan azalan algoritması ile

yapılmaktadır. Örnek olarak elimizde T , Y i = 1,2,3, . . . , N şeklinde Çizelge 2'de gösterilen şekilde bir veri seti olsun.

Çizelge 2 Örnek Veri Seti

MIMO	**Giriş Verisi**				**Çıkış Verisi**				**Model Çıkışı**			
i	$T_i \in R^R$				$Y_i \in R^M$				$y_i \in R^M$			
1	X_{11}	X_{12}	…	X_{1R}	Y_{11}	Y_{12}	…	Y_{1M}	y_{11}	y_{12}	…	y_{1M}
2	X_{21}	X_{22}	…	X_{2R}	Y_{21}	Y_{22}	…	Y_{2M}	y_{21}	y_{22}	…	y_{2M}
3	X_{31}	X_{32}	…	X_{3R}	Y_{31}	Y_{32}	…	Y_{3M}	y_{31}	y_{32}	…	y_{3M}
…		…				…				…		
N	X_{N1}	X_{N2}	…	X_{NR}	Y_{N1}	Y_{N2}	…	Y_{NM}	y_{N1}	y_{N2}	…	y_{NM}

Hali hazırdaki veriler T , Y i = 1,2,3, . . . , N olduğunda, çıkış verileri de $Y \in \{-1,1\}$ ise sınıflandırma problemi olarak isimlendirilir. Ancak çıktı verileri $Y \in R^M$ olduğunda regresyon problemi olacaktır. Mevcut veri setini eğitim ve test olarak iki parçaya bölmemiz gerekmektedir. Örnek olarak %80 eğitim %20 test şeklinde verimizi rassal olarak bölebiliriz. Ayrılan eğitim verileriyle sinir ağı eğitilecektir. Eğitim işlemi daha öncesinde de bahsedildiği üzere $\boldsymbol{W}_g$, $\boldsymbol{b}_g$, $\boldsymbol{W}_Ç$, $\boldsymbol{b}_Ç$ parametrelerinin en hatasız halini bulmaktır. Çizelge 2'de bahsedilen gradyan azalan algoritması ile, parametrelerin değerleri amaç fonksiyonunu minimum yapana dek hesaplanacaktır. Hesaplama tamamlandıktan sonra daha öncesinde ayrılan test verileri ile model test edilecek ve sonuçlar karşılaştırılacaktır. Sinir ağının maliyet fonksiyonu denklem 18'de gösterilmektedir.

$$\frac{1}{n}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{18}$$

Şekil 16'da sinir ağlarının eğitimi için gradyan azalan algoritması gösterilmektedir (Goodfellow et al., 2017).

ALGORİTMA : Yapay Sinir Ağlarının Gradyan Azalan Algoritmasıyla Eğitimi

1 : $\boldsymbol{W_g, b_g, W_ç, b_ç, s}$ *Parametrelerin başlangıç değerlerini belirle*

2 : $T_i, Y_i i = 1,2,3,\ldots,N$ *Veri setini üç parçaya böl*

3 : $i \leftarrow 0$

4 : $\hat{Y}_j = \boldsymbol{W_ç} H(\boldsymbol{W_g} T_i + \boldsymbol{b_g}) + \boldsymbol{b_ç}$ *Yapay sinir ağının eğitim verisiyle çıkışını hesapla*

5 : $\boldsymbol{e_i} = \frac{1}{n}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$ *Yapılan hatayı hesapla*

6 : $\boldsymbol{p_i} = -\nabla\hat{Y}(\boldsymbol{W_g, b_g, W_ç, b_ç})$ *Parametrelere göre gradyan vektörünü hesapla*

7 : $\boldsymbol{W_g, b_g, W_ç, b_ç}(i+1) = \boldsymbol{W_g, b_g, W_ç, b_ç}(i) + \boldsymbol{p_i e_i}$ *Parametreleri güncelle*

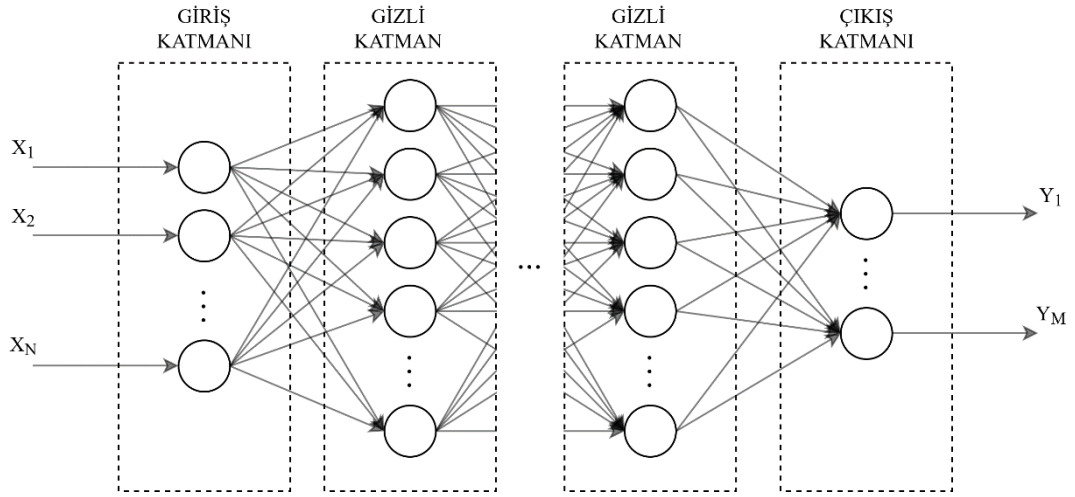
8 : *Eğer* $\| \boldsymbol{e_i} \| < 1e-6$ *Algoritmayı sonlandır*

9 : *Değilse 3. adıma geri dön*

Şekil 16 Yapay Sinir Ağlarının Gradyan Azalan Algoritmasıyla Eğitimi Algoritması

7. Çok Katmanlı Yapay Sinir Ağları

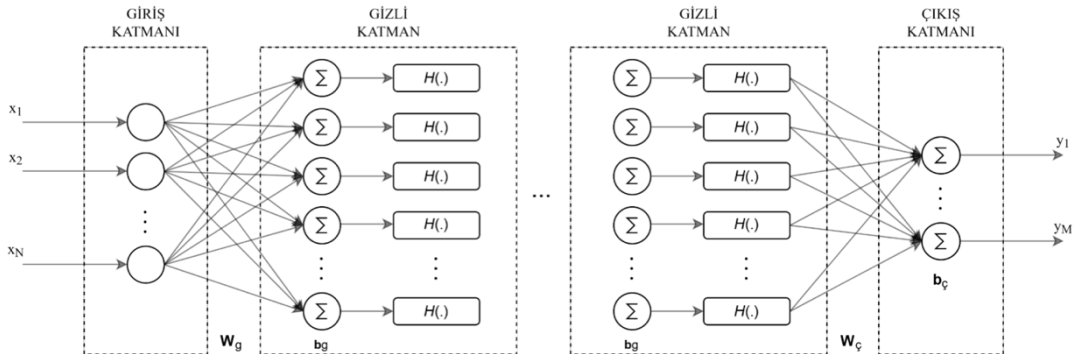
Çok katmanlı yapay sinir ağları, tek katmanlı yapay sinir ağlarının genişletilmiş halidir. Tek katmanlı yapay sinir ağlarından farklı olarak problemin zorluluğuna ve farklılığına göre mimariye istenildiği kadar gizli katman eklenebilmektedir. Şekil 17'de çok katmanlı yapay sinir ağları gösterilmektedir.



Şekil 17 Çok Katmanlı Yapay Sinir Ağı

8. Çok Katmanlı Yapay Sinir Ağının İleri Modeli

Çok katmanlı ve çok girişli çok çıkışlı sinir ağının modeli şekil 18'de gösterilmiştir (Goodfellow et al., 2017).



Şekil 18 Çok Katmanlı Çok Girişli Çok Çıkışlı Yapay Sinir Ağı

Çok katmanlı yapay sinir ağının matematiksel olarak ifadesi denklem 19'da gösterilmiştir.

$$\hat{Y}_j = W_{ç} H\left(W_{gk} \dots H\left(W_{g2} H\left(W_{g1} X_i + b_{g1}\right) + b_{g2}\right) \dots + b_{gk}\right) + b_{ç} \qquad (19)$$
$$i = 1,2,\dots,Nj = 1,2,\dots,Mk = 1,2,\dots,K$$

Denklem 19'da K adet gizli katman ve $X \in R^N$ ve $Y \in R^M$ olmalıdır. Aynı şekilde çok katmanlı yapay sinir ağının optimizasyon problemi şekil 19'da gösterilen algoritma ile çözülebilmektedir.

Adam algoritması stokastik tanımlanan amaç fonksiyonu ve fonksiyonun gradyanı üzerinden geliştirilen bir algoritmadır (Goodfellow et al., 2017). Algoritma adını adaptif moment tahmininden almaktadır. Bu çalışmada pekiştirmeli öğrenme algoritmaları için oluşturulan çok katmanlı yapay sinir ağlarının eğitimi için kullanılmıştır (Kingma and Ba, 2015). Şekil 19'da Adam algoritmasının sözde kodu gösterilmiştir.

ALGORİTMA : ADAM Algoritması

1	:	α	*Adım aralığını belirle (0.001) önerilen*
2	:	$\beta_1, \beta_2 \in [0,1)$	*Moment değerinin üstel çürüme parametresi*
3	:	$f(\boldsymbol{\theta})$	*Stokastik amaç fonksiyonu*
4	:	$\boldsymbol{\theta}_0$	*Parametrelerin başlangıç değeri*
5	:	$\boldsymbol{m}_0$	*Birinci dereceden moment vektörü*
6	:	$\boldsymbol{v}_0$	*İkinci dereceden moment vektörü*
7	:	$t \leftarrow 0$	
8	:	$t \leftarrow t + 1$	
9	:	$\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$	*Stokastik amaç fonksiyonunun gradyan vektörü*
10	:	$\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$	*Birinci moment tahminin güncellenmesi*
11	:	$\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$	*İkinci moment tahmini güncellemesi*
12	:	$\widehat{\boldsymbol{m}}_t \leftarrow \frac{\boldsymbol{m}_t}{(1-\beta_1^t)}$	*Birinci momentin gerçek değeri*
13	:	$\widehat{\boldsymbol{v}}_t \leftarrow \frac{\boldsymbol{v}_t}{(1-\beta_2^t)}$	*İkinci momentin gerçek değeri*
14	:	$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \frac{\widehat{\boldsymbol{m}}_t}{\sqrt{\widehat{\boldsymbol{v}}_t} + \varepsilon}$	*Parametrelerin güncellenmesi* $\varepsilon = 1e-8$
15	:	Eğer $\| \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1} \| < 1e-6$	
16	:	Algoritmayı sonlandır	

Şekil 19 Adam Algoritması

B. Pekiştirmeli Öğrenme

Pekiştirmeli öğrenme algoritmasında yapay zeka ajanları belirlenen bir alana dağılır ve ceza-ödül mekanizmasına göre optimal sonuçları bulmaları beklenir. Ajan öğrenme sırasında doğru kararlar verdiğinde ödüllendirilir, yanlış kararlar verdiğinde ise cezalandırılırlar. Pekiştirmeli öğrenmede ajanlar biyo-ilhamlıdır ve aynı canlılarda olduğu içi hayatta kalma ve büyüme için ödül ve ceza deneyiminden öğrenirler. Pekiştirmeli öğrenmede ajanlardan ödüllerini maksimize etmeleri beklenir (Khan et al., 2012). Bu şekilde yapay zeka ajanları optimal bir politika geliştirir.

Pekiştirmeli öğrenme araştırması kırk yıldır devam etmektedir ve kökeni bilgisayar bilimlerine dayanmaktadır, ancak uyarlamalı dinamik programlama (ADP) ve nörodinamik programlama (NDP) gibi benzer yöntemler Werbos, Bertsekas ve diğer birçok araştırmacı tarafından paralel olarak geliştirilmektedir. Williams'a göre, modern pekiştirmeli öğrenme, yapay zekadan gelen zaman fark yöntemlerinin, optimal ve kontrol ve hayvan çalışmalarından öğrenme teorilerinin bir karışımıdır (2009) Werbos'un son çalışmaları (2004; 2007; 2008; 2009) sınırları daha da aşmış

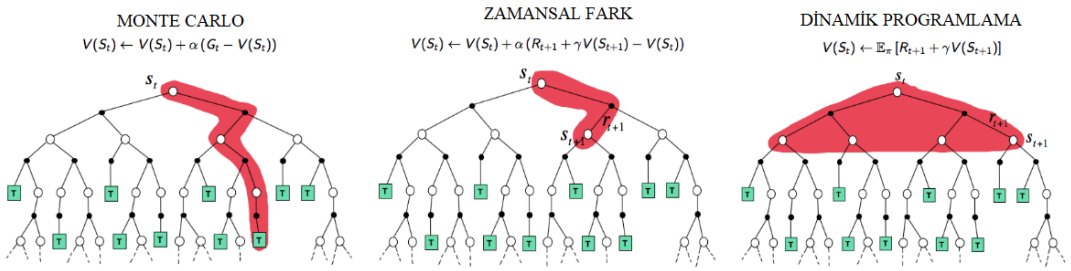
ve beynin nasıl çalıştığın anlamak ve geliştirerek daha iyi hale getirmek için pekiştirmeli öğrenmeyi kullanmıştır.

Şekil 10'a göre pekiştirmeli öğrenme algoritması da kendi içinde 2 alt gruba ayrılır. Bu gruplardan ilki model tabanlı algoritmalardır, diğeri ise model tabanlı olmayan algoritmalardır. Model tabanlı olmayan algoritmalar günümüzde daha yaygın olarak kullanılmaktadır (Richard and Andrew, 2015; Yu, 2020). Model tabanlı olmayan algoritmalar da kendi içlerinde iki gruba ayrılabilir: politika tabanlı algoritmalar ve değer tabanlı algoritmalar.

Politika, bir yapay zeka ajanının belirli bir zamanda davranışı olarak tanımlanır. Politika, pekiştirmeli öğrenme şemalarının çok önemli bir parçasıdır ve bir arama tablosu veya bir fonksiyon ile temsil edilebilir. Politika stokastik veya deterministik olabilir. Bazı pekiştirmeli öğrenme şemalarında, politikanın hesaplamasında karmaşık bir arama süreci kullanılır. Bu süreç içerisinde değer fonksiyonunun en aza indirilmesi üzerinde çalışılır (Sutton and Barto, 1998).

Değer işlevi, gelecekteki ödülün tahmini ve aracın daha yüksek ödüller üretecek bir eylemde bulunmayı beklediği temeldir. Normalde iki tür değer işlevi kullanılır, yani genel olarak V(s) ile temsil edilen durum değeri işlevi ve Q(s,a) ile temsil edilen eylem değer işlevi. Burada s durumu, a ise eylemi temsil eder. Garcia'a göre, ortamın modeli bilindiğinde, dinamik programlama yöntemlerinde olduğu gibi bir durum değeri fonksiyonu kullanılır. Ancak ortamın modeli bilinmiyorsa, o zaman bir eylem değeri işlevi tercih edilir (2005).

Çalışmada yazar pekiştirmeli öğrenme problemlerini çözmek için üç ana yöntem sıralamıştır; dinamik programlama (DP), monte carlo (MC) yöntemleri ve zamansal fark yöntemleri (ZM). Bu yöntemler şekil 20'de gösterilmiştir (Garcia, 2005).



Şekil 20 Çeşitli Pekiştirmeli Öğrenme Algoritmaları Grafiği

1. Rastgele Süreçler ve Markov Süreci

Fiziksel sistemlerin matematiksel modelleri lineer olmayan diferansiyel denklem 20'de açıklanmıştır.

$$\dot{x}(t) = f(t, x, u, w) \qquad (20)$$
$$y(t) = g(t, x, u, v)$$

Denklem 20'de $\boldsymbol{t}$ zamanı, $\boldsymbol{x}$ durum vektörünü, $\boldsymbol{u}$ kontrol işaretini, w proses gürültüsünü, v ölçüm gürültüsünü, f (.) sistem modelini oluşturulan lineer olmayan diferansiyel denklemleri, g(.)çıkış, denklemini temsil etmektedir. Fiziksel sistemlerde, eğer sistemin matematiksel modeli biliniyorsa, belirlenen durum geçiş matrisi üzerinden elde edilebilir. Ancak genel itibariyle fiziksel sistemlerdeki belirsizlik, gürüldü ve herhangi bir şekilde model elde edilememesi dinamikleri yüzünden çözülememektedir. Rastgele süreçlerde var olan sistemin modelini oluşturmak için durumu temsil eden $\boldsymbol{x}$'ler rastgele değişken olarak temsil edilmektedir. $\boldsymbol{x}$'ler rastgele olarak kabul edildiğinde sistem stokastik bir biçimde modellenmiş olur. Pekiştirmeli öğrenmede fiziksel sistemler ve durumlar arasındaki geçişler Markov karar süreci ile modellenmektedir. Rastgele süreç matematiksel olarak $\boldsymbol{X}(t)$, $\boldsymbol{X}(n)$ ile ifade edilir. Olasılık uzayındaki tüm fonksiyonların birleşimi rastgele ya da stokastik süreç olarak adlandırılır. Bu fonksiyonların birleşiminde basit (PDF) veya ortak olasılık yoğunluk fonksiyonları (JPDF), zamanla değişmiyorsa bu süreç durağan olarak adlandırılmaktadır. Durağan süreçlerde, modeller ve parametreler zamandan bağımsız olarak nitelendirilir. Bazı durumlarda tüm bu rastgele süreç aynı istatistiksel özellikleri sergilemektedir. Bu gibi durumlarda tek bir rastgele sürecin örnek fonksiyonlarının bilinmesiyle birlikte tüm sürecin istatistiksel özellikleri modellenebilir. Rastgele süreçler ayrık değerli ayrık zamanlı rastgele süreç, ayrık değerli sürekli zamanlı rastgele süreç, sürekli değerli ayrık zamanlı rastgele süreç ve son olarak sürekli değerli zamanlı süreç olarak dört sınıfta incelenmektedir. Markov süreci denklem 21'de gösterilmiştir.

$$P[X(t_{k+1}) = x_{k+1} | X(t_k) = x_k, \dots, X(t_1) = x_1] \qquad (21)$$
$$= P[X(t_{k+1}) = x_{k+1} | X(t_k) = x_k]$$

Denklem 21'de gösterildiği gibi $X(t)$ rastgele sürecinin, $X(t_{k+1})$ değeri bir geçmiş değer olan $X(t)$ 'ye bağlıdır. Bu özellik Markov özelliği olarak adlandırılır. Rastgele sürece ilişkin ortak olasılık yoğunluk fonksiyonunun (JPDF) bilinmesi durumunda

Markov süreci durum geniş matrisiyle ifade edilmektedir ve Denklem 22'de gösterilmiştir.

$$P[X_{n+1} = j|X_n = i] = P_{ij} \tag{22}$$

Denklem 22'te X_n homojen geçiş olasılıkları olarak adlandırılmaktadır. $X_n, \ldots, X_0$ ortak olasılık yoğunluk fonksiyonunun verilmesiyle denklem 23 ortaya çıkar.

$$P\left[X_n = i_{n,\ldots,} X_0 = i_0\right] = p_{i_{n-1},} i_n \ldots p_{i_0 i_1} p_{i_0}(0) \tag{23}$$

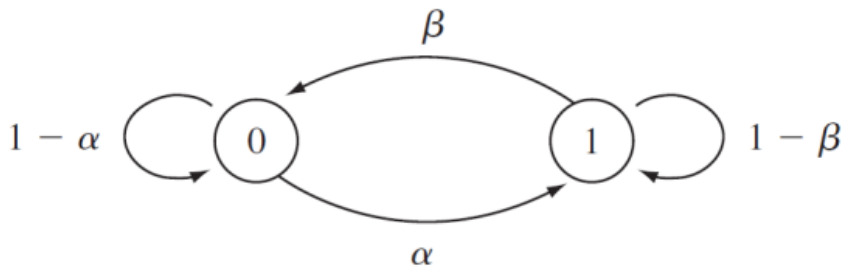
Buradan geçiş olasılıkları matrisini P , elde etmek için başlangıç değerlerine $p_i(0)$ Verildiğinde denklem 24 elde edilir.

$$P = \begin{bmatrix} p_{00} & p_{01} & p_2 & \cdots \\ \vdots & \vdots & \cdots & \cdots \\ \vdots & \vdots & \cdots & \cdots \\ p_{i0} & p_{i1} & \cdots & \cdots \end{bmatrix} \tag{24}$$

Denklem 24'ün sütunları toplandığında denklem 25 ortaya çıkacaktır.

$$\sum_j P[X_{n+1} = j|X_n = i] = \sum_j pij = 1 \tag{25}$$

Şekil 21'de İki durumlu Markov süreci için örnek bir akış gösterilmiştir.



Şekil 21 Markov Süreci İçin Örnek Bir Akış

Şekil 21'de ifade edilen iki durumlu Markov zincirinin akış matrisi denklem 26'de gösterilmiştir.

$$P = \begin{bmatrix} 1-a & a \\ \beta & 1-\beta \end{bmatrix} \tag{26}$$

Denklem 26'nin kuvvetleri alındığında Markov zincirinin zamana bağlı hesaplaması gerçekleştirilir. Denklem 27'de gösterilmiştir.

$$P^n = \begin{bmatrix} 1-a & a \\ \beta & 1-\beta \end{bmatrix} \begin{bmatrix} 1-a & a \\ \beta & 1-\beta \end{bmatrix} \cdots \begin{bmatrix} 1-a & a \\ \beta & 1-\beta \end{bmatrix} \quad (27)$$

Denklem 8 ile n durumlu geçiş olasılıkları, P durum geçiş matrisinin bilinmesiyle hesaplanabilir. Bir X_k rastgele süreci bağımsız ve aynı olasılık dağılımıyla ifade edilmiş ise bu süreç IID olarak ifade edilmiştir ve denklem 28'da gösterilmiştir.

$$F_{x_1,x_2,\dots,x_k}(x_1, x_2, \dots, x_k) = P\left[X_1 \leq x_1, X_2 \leq x_2, \dots, X_k \leq x_k\right] \quad (28)$$
$$F_{x_1,x_2,\dots,x_k}(x_1, x_2, \dots, x_k) = F_x(x_1)F_x(x_2)\dots F_x(x_k)$$

Denklem 28'da X_k rastgele değişkenlerin istatistiksel bağımsızlığını ifade eder. Ancak bu bölümde pekiştirmeli öğrenme algoritmalarındaki ajanların ortamlardan elde ettikleri veriler IID olmamalıdır. Bu şekilde elde edilen verilerden beslenen modeller eğitim sürecinde lokal noktaya takılmaktadır (Garcia, 2005).

2. Markov Karar Süreci

Markov karar süreci, ödül sürecinin $a \in A$ olduğu aksiyon kümesiyle genişletilmiş versiyonudur. Burada S Markov sürecini sağlayan sonlu ve ayrık durum kümesidir. A aksiyon kümesi olarak tanımlanır. $P^a = P[S = s'|S = s, A = a]$ denklemi koşullu geçiş durumunu ve $R^a = E[R\,|S = s, A = a]$ ödül fonksiyonunu tanımlar. $\gamma \in [0,1]$ İse azaltma faktörü olarak adlandırılabilir. Markov karar sürecinde iki önemli faktör vardır. Bunlar politika ve değer fonksiyonu olarak isimlendirilir. Politika π şeklinde gösterilir. Denklem 29'da tanımlanmıştır.

$$\pi(\alpha|s) = P[A_t = \alpha|S_t = s] \quad (29)$$

Denklemde s ajanının durumunu ve a alacağı aksiyonu belirtir. Markov sürecinde politika s_t olarak gösterilen ajanın t zamanındaki durumunu belirtir. Politika ise zamandan bağımsız olarak tanımlanır ve denklem 30'da açıklanmıştır.

$$A_t \sim \pi(.\,|S_t) \forall t > 0 \quad (30)$$

Denklem 18'de ve Markov karar sürecinde $M = (S, A, P, R, \gamma)$ olarak tanımlanır ve politika olarak ödül verildiğinde durumlar Markov özelliği olan (S,P^{π}) fonksiyonunu sağlamalıdır. Denklem 31'de P^{π}, R^{π} formülleri tanımlanmıştır.

$$P_{ss'}^{\pi} = \sum_{\alpha \epsilon A} \pi(\alpha|s) P_{ss'}^{\alpha}, \quad R_s^{\pi} = \sum_{\alpha \epsilon A} \pi(\alpha|s) R_s^{\alpha} \tag{31}$$

Markov sürecindeki değer fonksiyonu denklem 32'de tanımlanmıştır.

$$V_{\pi}(s) = E_{\pi}[G_t|S_t = s] \tag{32}$$

Denklem 33'te durum-değer fonksiyonu görülmektedir. Yapay zeka ajanları Markov süreçleri ile s durumundan başlayıp sonuca kadar π politikasını takip ederek ödüllere ulaşmaya çalışır. Markov süreçlerindeki aksiyon-değer fonksiyonu denklem 33'te tanımlanmıştır.

$$q_{\pi}(s, \alpha) = E_{\pi}[G_t|S_t = s, A_t = \alpha] \tag{33}$$

Denklem 33'teki aksiyon-değer formülü daha öncesinde modellenmiş belirli bir çevrede s durumundan başlayarak ve belirlenen politikayı takip ederek formülde a olarak tanımlanmış aksiyonu alması sonucunda ödüllere ulaşmaya çalışır. Daha öncesinde belirtildiği üzere Markov sürecinde iki farklı çözüm uygulaması bulunmaktadır. Durum-değer ve aksiyon-değer formülleri iteratif hale getirildiğinde sırasıyla denklem 34 ve denklem 35 ortaya çıkar.

$$V_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma V_{\pi}(S_{t+1})|S_t = s] \tag{34}$$

Denklem 34 durum-değer fonksiyonunun Bellman denklemi ile harmanlanmış halidir. Bu durumda durum-değer fonksiyonu belirli bir $S_t = s$ durumunda belirlenmiş bir π politikasını takip eder ve sonuç olarak $V_{\pi}(s)$ değerini hesaplar.

$$q_{\pi}(s, \alpha) = E_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1})|S_t = s, A_t = \alpha] \tag{35}$$

Denklem 34'te aksiyon-değer fonksiyonu tanımlanmıştır. Bu denklemi kullanarak yapay zeka ajanı Markov süreci ile modellenmiş ve belirli bir politikayı takip ederek

$S_t = s$ $A_t = a$ durumunda $q_\pi(s, a)$ fonksiyonunu hesaplar. Denklem 34'ün ve denklem 35'in lineer sistemde çözümlenmiş hali denklem 36'te tanımlanmıştır.

$$V_\pi = R^\pi + \gamma P^\pi V_\pi \tag{36}$$

$$V_\pi = (1 - yP^\pi)^{-1} R^\pi$$

Markov ödül sürecinde, denklem 34'te tanımlanmış formülizasyon çözümlendiğinde aynı denklem 35'te tanımlanan Markov karar sürecinin de bir çözümü bulunmaktadır. Denklem 36 uygulanması sonlu sayıda durum için, sonlu sayıda geçiş olasılıkları matrisinin bilindiği durumda mümkündür. Eğer geçiş olasılığı bilinmezse farklı Markov karar süreçleri uygulanmaktadır. Bunlar: yaklaşık dinamik programlama, zamansal fark ve monte carlo metodu olarak adlandırılır. Markov sürecinin optimal çözümü denklem 37'de tanımlanmıştır.

$$V_*(s) = \max_\pi V_\pi(s) \tag{37}$$

Denklem 37'de $V*(s)$ optimal değer fonksiyonu olarak adlandırılır ve tüm politikaların değerlerinin maksimum olarak seçilmesiyle hesaplanır. Denklem 38'de ise optimal aksiyon değer fonksiyonu tanımlanmıştır.

$$q_*(s, \alpha) = \max_\pi q_\pi(s, \alpha) \tag{38}$$

Optimal durum-değer fonksiyonu aynı zamanda Markov karar sürecinin çözümünü oluşturur. Aynı şekilde Markov aksiyon-değer fonksiyonunun belirlenmesiyle de çözülebilir. Kısaca açıklamak gerekirse Markov sürecinin iki farklı çözüm metodu mevcuttur. Bunlar durum-değer ve aksiyon-değer olarak daha öncesinde tanımlanmıştır. Optimal durum-değer fonksiyonu belirlenmiş politikalar üzerinde hareket ederken denklem 39'u sağlamalıdır.

$$\pi \geq \pi' \qquad V_\pi(S) \geq V_{\pi'}(s) \qquad \forall s \tag{39}$$

Denklem 39'u genel itibariyle açıklamak gerekirse politika değeri, bir sonraki adımdaki politika değerinden büyük ve eşit ise bundan sonraki tüm durum-değer fonksiyonları da aynı şekilde politika değerinden büyük ve eşit olmalıdır.

Ancak Markov karar sürecinde, optimal bir politika π_* vardır ve bu π_* değeri $\pi_* \geq \pi \forall \pi$ koşulunu sağlamak zorundadır. Aynı zamanda optimal olarak hesaplanan politika değeri, optimal olarak hesaplanan durum-değer fonksiyonunu sağlamak zorundadır. Bu durum denklem 40'ta gösterilmiştir.

$$V_{\pi}, (s) = V_*(s) \tag{40}$$

Denklem 40'ta gösterildiği üzere hesaplanan tüm optimal politika değerleri de aksiyon-değer fonksiyonunu sağlamak zorundadır.

$$q_{\pi_*}(s, a) = q_*(s, a) \tag{41}$$

İki durumun da sağlandığı formülizasyon denklem 42'de tanımlanmıştır.

$$\pi_*(\alpha|s) = \begin{cases} 1 & \text{a} = \arg\max_{\alpha \epsilon \forall} q_*(s, \alpha) \\ 0 & diğer\ durumlar \end{cases} \tag{42}$$

Daha öncesinde de bahsedildiği üzere politika, yapay zekâ ajanının belirlenen durumda $a \in A$ aksiyonunu nasıl seçeceğini belirtmektedir. Denklem 42'de sürecin optimal politika seçimi tanımlanmıştır. Bellman'ın öz yinelemeli olarak hesaplama formülü denklem 43'de tanımlanmıştır.

$$V_*(s) = \max_{\alpha} q_*(s, \alpha) \tag{43}$$

Denklem 43'deki durum-değer formülüne, Bellman denklemi eklendiğinde Denklem 44'e ulaşılır.

$$V_*(s) = \max_{\alpha} R_s^{\alpha} + y \sum_{s' \epsilon S} P_{ss'}^{\alpha} V_*(s') \tag{44}$$

Denklem 43 ve denklem 44'de tanımlanan formülasyona benzer şekilde aksiyon-değer fonksiyonu optimal olarak hesaplanmış durum-değer fonksiyonu ile birleştirildiğinde denklem 45 ortaya çıkmaktadır.

$$q_*(s, \alpha) = R_s^{\alpha} + y \sum_{s' \epsilon S} P_{ss'}^{\alpha} V_*(s') \tag{45}$$

Eğer denklem 45 ve denklem 44 birleştirilirse optimal aksiyon-değer fonksiyonu denklem 46'daki gibi olacaktır.

$$q_*(s,\alpha) = R_s^{\alpha} + y\sum_{s\prime\in S} P_{ss\prime}^{a} \max_{\alpha\prime} q_*(s',\alpha') \tag{46}$$

Denklem 45 ve denklem 46'da Markov karar sürecinin iteratif versiyonları gösterilmiştir. Ancak Bellman'ın optimal denkleminin çözümü lineer değildir ve analitik çözümleri ne yazık ki bulunmamaktadır. Standart Markov karar süreci denkleminde P_{ss}^{a} bilinmediğinden dolayı denklem 44 ve denklem 43 kullanılmak zorundadır. Bir sonraki bölümde ise Markov süreçlerinin iteratif çözümlü versiyonları olan değer tabanlı ve politika tabanlı yaklaşımlar incelenecektir (Silver, 2015).

3. Markov Karar Süreci Çözüm

Markov karar süreci $M = (S, A, P, R, \gamma)$ fonksiyonu ile ifade edilir. Daha öncesinde de bahsedildiği üzere fonksiyon içerisinde durumlar, aksiyonlar, geçiş olasılığı matrisi, ödül fonksiyonu ve azaltma faktörünü barındırmaktadır. Açıkça söylemek gerekir ki Markov sürecinde yapay zeka ajanının alacağı karar ve uygulayacağı politika matematiksel olarak tanımlıdır. Stokastik olarak tanımlanan politika [0,1] arasında tanımlanmakta ve yine aynı aralık içerisindeki bir olasılık değerine karşılık gelmektedir. Markov karar sürecindeki değer fonksiyonu Bellman denklemi ile genişletilirse denklem 47 ortaya çıkar.

$$\hat{V}_*(s) = R(s) + \gamma \max_{\alpha\epsilon A} \sum_{s\prime\in S} P(s'|s,\alpha)\, \hat{V}_*(s') \tag{47}$$

Optimal durum değer fonksiyonu ise denklem 48'de tanımlanmıştır.

$$\hat{V}_*(s) = R(s) + \gamma \max_{\alpha\epsilon A} \sum_{s\prime\in S} P(s'|s,\alpha)\, \hat{V}_*(s') \tag{48}$$

Markov karar sürecinde denklem 47 ve denklem 48 iteratif olarak hesaplandığında optimal çözüme ulaşılır. Denklem 47'de de fark edileceği üzere değer tabanlı yaklaşımda aç gözlü bir politika benimsenmektedir. Eğer Markov karar sürecinde R ve P bilinmiyor ise bu durumda yaklaşık çözümler oluşturulur.

Markov sürecinin robotik sistemlerde pek sık kullanılmamasının nedeni durum geçiş matrisinin tam olarak bilinmemesinden kaynaklanmaktadır. Bundan dolayı robotik

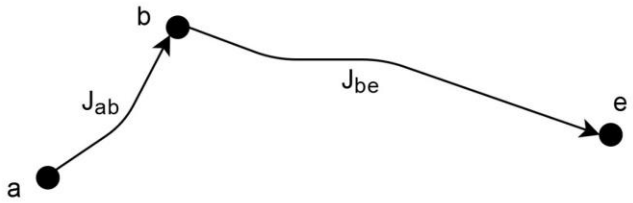
sistemlerde Markov süreçleri yerine zamansal fark ve monte carlo metodu kullanılmaktadır (Silver, 2015).

4. Dinamik Programlama

Optimizasyon problemleri ve karar verme problemlerinde dinamik programlama sistemi çokça kullanılmaktadır. Alt dallara ayrılabilen büyük ve yeterince karmaşık problemler dinamik programlama yöntemiyle ayrıştırılır. Sonrasında bu dallar tek tek çözülerek belleğe alınır ve sonrasında bellekte toplanan verilerle optimal çözüm bulunur. Dinamik programlama sisteminde her problem için ayrı bir çözüm tekniği oluşturulmalı ve tüm çözümler optimal değere ulaşmak zorundadır.

Hesaplamadaki verimlilik, dinamik programlamanın odaklandığı nokta olmuştur. Söz konusu model, kompleks problemleri basit parçalar halinde değerlendirdiğinden dolayı hızlı çalışır. Uyarlanabilir dinamik programlama söz konusu olduğunda, pekiştirmeli öğrenme de çalışma alanlarından biri haline gelir (Kulkarni, 2012).

Şekil 22'de örnek bir optimal yol probleminin grafiği mevcuttur.



Şekil 22 Optimal Yol Problemi Örnek Grafik

Şekilde a noktasından b noktasına gitmenin maliyeti J_{ab} olarak belirlenmiş, b noktasından e noktasına gitmenin maliyeti ise J_{be} olarak tanımlanmıştır. Optimal maliyet denklemi aşağıdaki gibi olacaktır.

$$J*_{ae} = J_{ab} + J_{be} \tag{49}$$

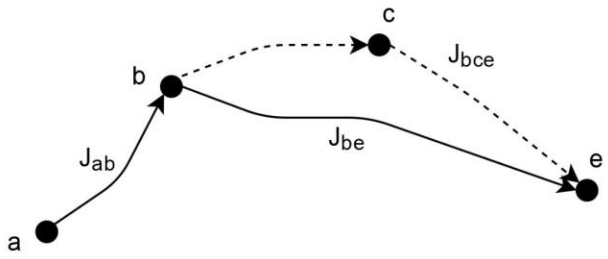
Denklem 49'da J^*_{ae} olarak tanımlanan parametre a noktasından e noktasına olan maliyeti temsil etmektedir. Eğer J^*_{ae} optimal ise denklemdeki tüm noktalar optimal olmak zorundadır aksi halde bir ikilem oluşur. Bu durum denklem 50'de gösterilmiştir.

$$J_{bce} < J_{be} \tag{50}$$

Eğer denklem 49 ve denklem 50 birleştirilirse denklem 51 ortaya çıkar.

$$J_{\alpha b} + J_{bce} < J_{\alpha b} + J_{be} = J *_{\alpha e} \tag{51}$$

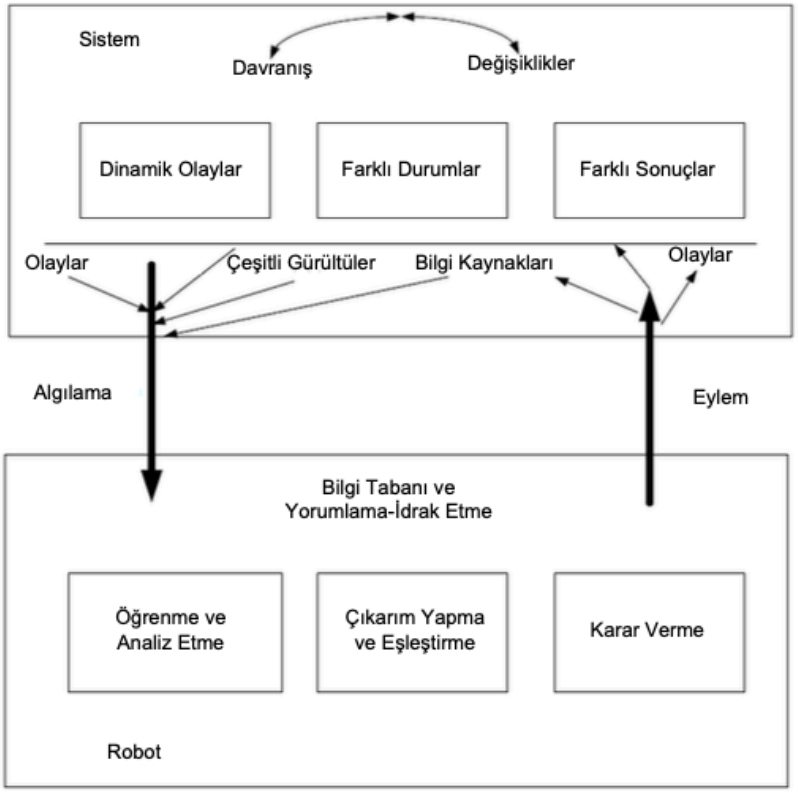
Optimal yol problemi çelişkisi şekil 23'te gösterilmiştir.



Şekil 23 Optimal Yol Problemi Çelişkisi

Doğrusal programlamada optimallik prensibi dağınık sistemler için $\pi^* = \{\pi_0^*, \pi_1^*, \ldots, \pi_{N-1}^*\}$ fonksiyonuna tanımlanabilir. Tüm S_t durumları erişilebilir ise zamana bağlı olarak alt dallardaki problemlerin aktarım maliyetleri minimum olarak belirlenmelidir. Böylelikle alt dallardaki optimallik prensibi farklı farklı hesaplandıktan sonra birleştirilerek en optimal çözüm elde edilir. Dinamik programlamada iki farklı çözüm tekniği bulunmaktadır. Bunlar: ileri yönlü ve geri yönlü olarak tanımlanabilir. Dinamik programlamada, zamanlar ve durumlar arası geçişler rastgele olarak belirlenmişse buna stokastik dinamik programlama denir. Eğer zamanlar ve durumlar arasındaki geçişler deterministik ise buna da deterministik dinamik programlama denmektedir. Dinamik programlama model tabanlı pekiştirmeli öğrenme sınıfına girmektedir ve aynı zamanda Markov karar süreçlerinin bu yöntemle çözülebilmesi için daha öncesinde de söylenildiği gibi durum geçiş matrislerinin bilinmesi ve modellerinin oluşturulması gerekmektedir (Sutton and Barto, 2018).

Uyarlanabilir dinamik programlama modeli her hareketin ve/veya her ödülle cezanın sonucunda hesaplamaları tekrarlar ve en doğru yöntemi arar. Genel çalışma prensibi şekil 24'te gösterilmiştir.



Şekil 24 Dinamik Programlama Modeli Genel Çalışma Prensibi

5. Bellman Denklemi

Yapay zeka ajanının modelden bağımsız olan pekiştirmeli öğrenme sisteminde optimal değeri hesaplaması gerekmektedir. Bu formül denklem 52'da belirtilmiştir.

$$V(s) = E[G_t|S_t = s] \tag{52}$$

Denklem 52'de *G* parametresi modelde belirlenen zaman içerisinde ajanın alabileceği ödül değerini ifade etmektedir. *G* parametresi denklem 52'deki ($G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$) yerine eklenecek olursa Denklem 53'e ulaşılır.

$$\begin{aligned} V(s) &= E[G_t|S_t = s] \\ &= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s] \\ &= E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s] \\ &= E[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= E[R_{t+1} + \gamma V(S_{t+1})|S_t = s] \end{aligned} \tag{53}$$

Denklem 53'te durum değer fonksiyonu tanımlanmıştır ve E parametresi beklenen değer anlamına gelmektedir ve bir sonraki durumu temsil eder ($E[R_{t+1}] \rightarrow R_{t+1}$). Denklem 53'teki tanımlama eğer iteratif hale gelirse Denklem 54'deki Bellman denklemi elde edilir.

$$V(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} V(s) \tag{54}$$

Eğer Bellman denklemi aksiyon-değer fonksiyonu şeklinde yazılacak olursa denklem 55 ortaya çıkmaktadır.

$$q(s, \alpha) = R_s^{\alpha} + \gamma \sum_{s' \in S} P_{ss'}^{a} q(s, \alpha) \tag{55}$$

Denklem 54 ve denklem 55 kullanılarak Markov süreciyle modellenen değer tabanlı pekiştirme öğrenme yaklaşımı hesaplanabilir (Silver, 2015).

6. Pekiştirmeli Öğrenmede Politika Tabanlı Çözümler

Önceki bölümlerde Markov süreçlerinin optimal çözümü için durum-değer ve aksiyon-değer fonksiyonları tanımlanmıştır. Ancak yapay zeka ajanının arama yaparken oluşturulacağı politikaya değinilmemiştir. Eğer ajanın davranışı deterministik ise denklem 55, ajanın davranışı stokastik ise denklem 56 kullanılarak bir politika oluşturulabilir.

$$\pi_{\theta}(s) = \max_{\alpha \epsilon A} \theta\,(s, \alpha) \tag{56}$$

$$\pi_{\theta}(\alpha | s) = \frac{exp\theta(s, \alpha)}{\sum_{\alpha' \epsilon A} exp\theta(s, \alpha')}$$

Denklem 55 ve denklem 56 da tanımlanan politika, Markov karar sürecinde bahsedilen denklem 45'den farklıdır. Burada ajanın kullanılacağı politika aşama aşama hesaplanmalıdır. Bu sebepten dolayı denklem 57 oluşturulmuştur.

$$V_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma V_{\pi}(S_{t+1}) | S_t = s] \tag{57}$$

Basit bir politika arama algoritması şekil 25'te açıklanmıştır.

ALGORİTMA : Basit Politika Arama Algoritması

1 : $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}$ *M deneme için parametrelere rastgele gürültü ekle ve ampirik olarak ödül toplamını* $V^{(i)} = \sum_{t=1}^{T} \gamma^t r_t$ *hesapla.*

2 : $V^{(i)} \approx f(\theta^{(i)})$ *herhangi bir parametrik makine öğrenmesi modelini danışmanlı öğrenme modeline göre eğit.*

3 : $\theta \leftarrow \theta + \alpha \nabla_\theta f(\theta)$ *parametreleri güncelle.*

Şekil 25 Basit Politika Arama Algoritması

Şekil 25'te açıklanan basit politika arama algoritması haricinde başka politika arama algoritmaları da bulunmaktadır. Bunlardan ilki takviye-pekiştirme (reinforce) metodu olarak adlandırılır ve pekiştirmeli öğrenmenin ilk adımıdır. Değer t zamanında ajanın izlediği politika belirlenmek istenirse denklem 58 kullanılabilir.

$$V_\theta(s) = E[R(\tau); \theta] = \int p(\tau; \theta)\, R(\tau) dr \tag{58}$$

Pekiştirmeli öğrenme algoritmasının tamamen oluşturulması için denkleme gradyan hesabı da eklenerek denklem 59'a ulaşılır.

$$\begin{aligned} \nabla_\theta V_\theta(s) &= \nabla_\theta \int p(\tau; \theta)\, R(\tau) d\tau \\ &= \int \nabla_\theta p(\tau; \theta)\; R(\tau) d\tau \\ &= \int \frac{p(\tau; \theta)}{p(\tau; \theta)} \nabla_\theta p(\tau; \theta) R(\tau) d\tau \\ &= \int p(\tau; \theta) \nabla_\theta \log p(\tau; \theta)\, R(\tau) d\tau \\ &= E[\nabla_\theta \log p(\tau; \theta) R(\tau)] \end{aligned} \tag{59}$$

Denklemde belirlenen $V\theta(s)$ parametresinin gradyan hesabı $\nabla\theta V\theta(s)$ şeklinde tanımlanabilir. Eğer denklem 59 tekrar düzenlenirse denklem 60'a ulaşılır.

$$\begin{aligned} \nabla_\theta \log p(\tau; \theta) &= \nabla_\theta \log\Big(\prod_{t=1}^{T} p(S_{t+1}|S_t, \alpha_t) \pi_\theta(\alpha_t|s_t)\Big) \\ &= \nabla_\theta \sum_{t=1}^{T} (\log p(s_{t+1}|s_t, \alpha_t)) + \nabla_\theta \log \pi_\theta(\alpha_t|s_t) \\ &= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\alpha_t|s_t) \end{aligned} \tag{60}$$

Şekil 26'da denklem 60 kullanılarak politika gradyanı ve pekiştirme algoritması oluşturulmuştur.

ALGORİTMA : Politika Gradyanı ve Takviye Algoritması

1 : *Stokastik bir politika* π_θ *için M adımda* τ *yörüngeyi oluşturun.*

2 : *Gradyan değerini yaklaşık olarak hesaplayın.*

$$g_\theta \leftarrow \frac{1}{M}\left(\sum_{i=1}^{M}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta\left(a_t^{(i)}|s_t^{(i)}\right)\right) R(\tau^{(i)})\right)$$

3 : *Parametreleri güncelle.*

$$\theta \leftarrow \theta + \alpha g_\theta$$

Şekil 26 Politika Gradyanı ve Takviye Algoritması

Şekil 26'da açıklanan algoritma, gradyanın belirli bir örneklem ile hesaplanmış halini açıklamaktadır, gerçek gradyan değeri hesaplanamamıştır. Modelin belirlenmediği pekiştirmeli öğrenme yaklaşımlarında ortam üzerinde daha öncesinde herhangi bir bilgi olmadığından dolayı ajanın ilk olarak ortamı keşfetmesi ve keşiflerden topladığı bilgileri kullanması gerekir. Ajanın izleyeceği yol denklem 61'de tanımlanmıştır (Kolter, 2016).

$$\pi(s) = \begin{cases} \max_{\alpha \epsilon A} \hat{Q}\ (s, \alpha) & 1-\varepsilon \\ rastgele\ aksiyon & diğer\ durumlar \end{cases} \tag{61}$$

7. Pekiştirmeli Öğrenmede Değer Tabanlı Çözümler

Markov sürecinden daha önceki bölümlerde detaylıca bahsedilmiştir. Markov karar sürecinin çözülmesi optimal olarak hesaplanan durum-değer ve aksiyon-değer fonksiyonlarının birlikte çözülmesiyle sağlanır. İki denklem de denklem 62'de gösterilmiştir.

$$V_*(s) = \max_\alpha R_s^\alpha + \gamma \sum_{s\prime \epsilon S} P_{ss\prime}^\alpha V_*(s) \tag{62}$$

$$q_*(s,\alpha) = R_s^\alpha + \gamma \sum\nolimits_{s\prime \epsilon S} P_{ss\prime}^\alpha \max_\alpha q_*(s,\alpha)$$

Denklem 62'de $V_*(s)$ durum-değeri, $q_*(s,\ a)$ aksiyon-değeri ifade eder. Bu fonksiyonlar birlikte hesaplandığında Markov sürecinin optimal çözümü elde edilir. Daha önceki bölümlerde de bahsedildiği gibi durum geçiş matrisi ve ödül fonksiyonu bilinmediğinden dolayı Markov sürecinin kapalı formda herhangi bir çözümü

bulunmamaktadır. Ancak yapay zeka ajanı bulunduğu ortamdan bilgiler elde ederek t+1 durumunu $P(s'|s_t,a_t)$ dağılım fonksiyonunda örneklemiş olur. Yapay zekâ ajanının ortamdan topladığı bilgiler ile belirlenen ödül fonksiyonu üzerine yaklaşık olarak hesaplama yapılabilmektedir. Denklem 63'te açıklanmıştır

$$\hat{V}^{\pi}(s_t) = r_t \ + \gamma \hat{V}^{\pi}(s_{t+1}) \tag{63}$$

Denklem 63'te durum, belirlenen zaman anındaki ödül ve bir sonraki durumu $r_t + \gamma \hat{V}^{\pi}(s_{t+1})$ ile günceller. Eğer denklem 63'e zamansal fark da eklenirse denklem 64 elde edilebilir.

$$\hat{V}^{\pi}(s_t) = (1-\alpha)\hat{V}^{\pi}(s_t) + \alpha(r_t + \gamma \hat{V}^{\pi}(s_{t+1})) \tag{64}$$
$$\hat{V}^{\pi}(s_t) = \hat{V}^{\pi}(s_t) + \alpha(r_t + \underbrace{\gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t)}_{Zamansal\ Fark})$$
$$\alpha < 1$$

Dikkat edilmelidir ki belirlenen zamandaki $V\ (s_t)$ değerinin güncellenmesi, bu parametreye zamansal fark eklendiğinde sağlanacaktır. Yine aynı şekilde ödül değerindeki güncelleme $V\ (s_t)$ parametresinin zamansal fark eklenmiş haliyle sağlanabilir. Zamansal fark denklem 64'te zamanlar arasında bağlantı kurmaktadır. Zamansal fark fonksiyonu düzgün bir şekilde anlaşıldığında tahmin edilen değer ve gelecekte tahmin edilecek değerler arasında ilişki kurulabilir. Denklem 64'te ki hesaplama ile ödül değeri denklem 63'ten daha optimal bir şekilde hesaplanabilir. Bunun nedeni bahsedildiği gibi Zamansal fark fonksiyonunun formülizasyona eklenmesiyle sağlanmaktadır. Zamansal fark algoritması şekil 27'de gösterilmiştir.

ALGORİTMA : Zamansal Fark Algoritması

1	:	$\hat{V}^{\pi}(s) = 0 \forall s \in S$	*Algoritma başlangıç parametresi.*
2	:	$t \leftarrow 0$	
3	:	s_t, r_t	*Durumları ve ödül değerlerini gözlemle.*
4	:	$a = \pi(s)$	*Belirli bir politika ile bir aksiyon uygula.*
5	:	s_{t+1}	*Bir sonraki durumu gözlemle.*
6	:	$\hat{V}^{\pi}(s_t) = \hat{V}^{\pi}(s_t) + \alpha(r_t + \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t))$	*Zamansal Fark ile güncelleme yap.*
7	:	$t \leftarrow t + 1$	

Şekil 27 Zamansal Fark Algoritması

Zamansal fark algoritması ile herhangi bir Markov süreci oluşturmadan ödül ve durum değerini yaklaşık olarak hesaplamamızı sağlar. Bundan dolayı bu çalışmada da kullanılan Q-öğrenme algoritması ortaya çıkmıştır (Kolter, 2016).

8. Q-öğrenme Algoritması

Q-öğrenme fikri ilk olarak Watkins tarafından ortaya atılmıştır (1989). Sonrasında matematiksel modeli ve kanıtı Watkins ve Dayan tarafından sunulmuştur (1992). Q-öğrenme politika dışı bir yöntemdir ve Werbos un 1992'de yayınladığı çalışmasındaki (1992) ADHDP'sine (Eyleme bağlı buluşsal dinamik programlama) benzer. Öncesinde de söylendiği gibi Q-öğrenme politika dışı bir yöntemdir ve Q(s,a) fonksiyonu mevcut değerlendirilen politikadan bağımsız olarak tahmin edilir. Markov sürecinde optimal hesaplamanın bulunması için adımların sonuna kadar beklenmesi gerekirken, Q-öğrenmede değer işlevi her zaman iterasyonunda güncellenir. Geçiş olasılıklarının ve ödüllerin başlangıçta bilinmediği durumlarda Q-değeri yineleme algoritmasının uyarlanmasını sağlar (Geron, 2017). Bir Q-öğrenme işlevi uygulamalarının ana yararı mevcut politikanın dışındaki eylemlerden de öğrenmesine izin vermesidir ve kesin bir tahmine ya da kesin bir çevresel modelin kullanımına gerek duymamaktadır. Bu nedenle modelden bağımsız pekiştirmeli öğrenme sınıfına girer (Glascher et al., 2010; Luo et al., 2016). Ancak Q-öğrenme, aksiyon-değer fonksiyonu için istenen deneyimi elde etmek için yüksek miktarda gürültülü araştırma gerektirir bundan dolayı eğitim süresi Markov sürecinden daha uzun sürmektedir (Todorov et al., 2012).

Genel itibariyle Q öğrenmesi, durumların ve eylemlerin arasındaki ilişkiyi gerçekçi değerlerle tahmin etmeye dayanır. Q(s,a), s durumunda a hareketi yapılırsa ne kadar ödül kazanılacağını belirtir. Verimli olarak değerlendirilirse robotun s durumuna geçişte a hareketini hayata geçirmesi beklenir. Bu öğrenme modelinin özellikleri şöyle sıralanabilir:

- Bu öğrenme türüne göre, her hareketin sonucunda Q değerleri farklılaşacaktır.
- Belirli sayıda durum ve hareket ikilisi varsa, denklem çalıştırıldıkça değerler birbirlerine yakınlaşacaktır (Sutton and Barto, 1998)
- Öğrenme sağlanması için herhangi bir sırayla durum ve hareket denklemleri uygulanabilir, bir sıraya ihtiyaç duyulmaz.

Bu öğrenmenin özelliklerine dayanarak, ödüllerin kaybedilmeyeceği söylenebilir. Aksine elde edilen değerler her zaman saklanır.

Daha önceki başlıklarda bahsedildiği gibi değer tabanlı yaklaşıma alternatif olarak ortaya koyulan Q-öğrenme yönteminde aksiyon ve durumlar denklemi aşağıdaki gibidir.

$$Q^{\pi}(s,\alpha) = R(s) + \gamma \sum_{s' \epsilon S} P(s'|s,\alpha) Q^{\pi}(s', \pi(s')) \tag{65}$$

Denklem 65'te yapay zekâ ajanının politikası $\pi(s')$ şeklinde gösterilir ve denklem 65'in optimal olarak çözümü denklem 66'da gösterilmiştir.

$$Q^{*}(s,\alpha) = R(s) + \gamma \sum_{s' \epsilon S} P(s'|s,\alpha) \max_{\alpha' \epsilon A} Q^{*}(s', \alpha') \tag{66}$$

Denklem 66 Q-öğrenme yönteminin temeli olarak kabul edilmektedir ancak $P(s'|s,a)$,$R(s)$ 'nin bilinmediği durumlarda $Q^{*}(s,a)$ fonksiyonunun çözümünün bulunması için denklem 66'ya zamansal fark fonksiyonunun eklenmesi gerekmektedir.

$$\begin{gathered} \hat{Q}^{*}(s,\alpha) = (1-\alpha)\hat{Q}^{*}(s,\alpha) + \alpha(r + \gamma \max_{\alpha' \epsilon A} \hat{Q}^{*}(s', \alpha')) \\ \hat{Q}^{*}(s,\alpha) = \hat{Q}^{*}(s,\alpha) + \alpha(r + \underbrace{\gamma \max_{\alpha' \epsilon A} \hat{Q}^{*}(s', \alpha') - \hat{Q}^{*}(s,\alpha)}_{Zamansal\ Fark}) \end{gathered} \tag{67}$$

Denklemde γ gelecekteki ödülleri, R anlık ödülleri, α öğrenme katsayısını belirtmektedir. Yani Q öğrenmesinin tahminleri, Q değerlerine göre belirlenir. Bu sayede kompleks hesaplamalardan kaçınarak, uzun vadeli durum ve hareketlerin Q değerlerine göre öğrenilmesini mümkün kılar. Daha verimli olduğu için bu öğrenme şekli tercih edilir.

Denklem 67'nin algoritma olarak tasarlanmış hali Şekil 28'de gösterilmiştir.

ALGORİTMA : Q Öğrenme Algoritması

1 : $\alpha \in (0,1], \gamma$ *Öğrenme oranı ve azaltma parametresini belirle.*
2 : $\varepsilon > 0$ *Rastgele aksiyon seçimi için epsilon çok küçük bir sayı.*
3 : $Q(s,a) = 0 s \in S, a \in A$ *Q değerinin başlangıcı aksiyon ve durumlar için sıfır.*
4 : $Eps = 1000$ *Algoritmanın kaç bölüm çalışacağı.*
5 : $T = 100$ *Algoritmanın her bölüm içerisinde kaç adım atacağı.*
6 : $Eps \leftarrow 0$
7 : s *Ajanın başlangıç durumu.*
8 : $T \leftarrow 0$
9 : ε *olasılıkla rastgele a* t
10 : $1 - \varepsilon$ *olasılıkla* $a = \max_{a \in A} Q(s,a)$ *aksiyon seç.*
11 : *a aksiyon uygula,* s', r *bir sonraki durum ve ödülü kaydet.*
12 : $Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a))$ *Q tablosunu güncelle*
13 : $s \leftarrow s'$

Şekil 28 Q-Öğrenme Algoritması

Şekil 28 aynı zamanda Q-öğrenmenin temel yöntemlerini de açıklamaktadır (Kolter, 2016).

9. Sarsa Algoritması

Sarsa algoritması durum-eylem-durum-eylem yöntemini tekrar eder. Genel itibariyle Q-öğrenmeye benzer ancak politikaya dayalı bir zaman fark yöntemidir. Eylem değer fonksiyonu $Q(s',a')$ ile mevcut politika ve durum-eylem çifti için tahmin edilir. Q-öğrenmesinden farklı olarak keşfi hesaba katmaktadır. Sarsa algoritmasının matematiksel modeli denklem 68'da gösterilmiştir

$$\begin{aligned} \hat{Q}^{\pi}(s,\alpha) &= (1-\alpha)\hat{Q}^{\pi}(s,\alpha) + \alpha(r + \gamma \hat{Q}^{\pi}(s', \pi(s'))) \\ \hat{Q}^{\pi}(s,\alpha) &= \hat{Q}^{\pi}(s,\alpha) + \alpha(r + \underbrace{\gamma \hat{Q}^{\pi}(s', \pi(s')) - \hat{Q}^{\pi}(s,\alpha)}_{Zamansal\ Fark}) \end{aligned} \tag{68}$$

Denklem. 68'a bakıldığı taktirde belirli Q değerleri mevcut bir politika üzerinden güncellendiği fark edilecektir. Ancak Q-öğrenmesinde sabit olan politika, Sarsa yönteminde parametrik olarak değişmektedir. Bundan dolayı Sarsa algoritması açık bir politikaya bağlı bir yöntem olarak karşımıza çıkmaktadır. Sarsa yönteminde her iterasyonda Q değerinden bir politika üretilip o politika üzerinden tekrar hesaplama

yapılır ve buna göre güncelleme sağlanır. Sarsa algoritması genel itibariyle aç gözlü bir şekilde hesaplanır. Sarsa yöntemi şekil 29'da algoritma olarak açıklanmıştır.

ALGORİTMA : Sarsa Algoritması

1 : $\alpha \in (0,1], \gamma$ *Öğrenme oranı ve azaltma parametresini belirle.*
2 : $\varepsilon > 0$ *Rastgele aksiyon seçimi için epsilon çok küçük bir sayı.*
3 : $Q(s,a) = 0 s \in S, a \in A$ *Q değerinin başlangıcı aksiyon ve durumlar için sıfır.*
4 : $Eps = 1000$ *Algoritmanın kaç bölüm çalışacağı.*
5 : $T = 100$ *Algoritmanın her bölüm içerisinde kaç adım atacağı.*
6 : $Eps \leftarrow 0$
7 : s *Ajanın başlangıç durumu.*
8 : a *Ajanın s durumu için alması gereken aksiyonu belirliyoruz*
9 : $T \leftarrow 0$
10 : *a Aksiyonunu uygula* ve r, s' *kaydet.*
11 : a', *Ajanın* s' *durumu için alması gereken aksiyonu belirliyoruz*
12 : $Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$ *Q tablosunu güncelle*
13 : $s \leftarrow s', \quad a \leftarrow a'$

Şekil 29 Sarsa Algoritması

Algoritmadan da anlaşılacağı üzere Q-öğrenme yaklaşımı sadece optimal yolu hesaplarken, Sarsa yöntemi bu optimal yolu en güvenli şekilde hesaplamaya çalışır. Bundan dolayı Q-öğrenme yöntemi sadece maksimum ödüller üzerinde hareket ederken, Sarsa yöntemi bu maksimum ödülleri gelecekteki durumları ve aksiyonları da hesaba katarak hareket eder. Q-öğrenme yaklaşımı ile Sarsa yönteminin bir diğer farklı da Sarsa yöntemi iterasyonda güncel politikaya göre ilerlerken, Q-öğrenme kararını sadece maksimum ödül ve değer üzerinden hesaplar (Sutton ve Barto, 2018).

10. Derin Pekiştirmeli Q-öğrenme Algoritması

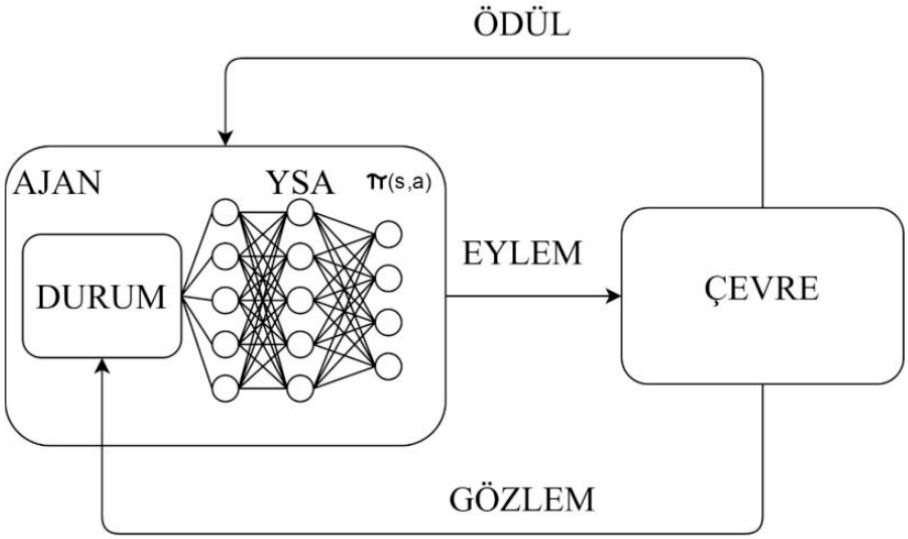
Daha önceki bölümlerde bahsedilen Q-öğrenme ve Sarsa yöntemleri tablo metotları olarak adlandırılmaktadır. Gerçek hayatta bu algoritmaların kullanımı kontrol problemleri ve durum uzayının karmaşıklığı nedeniyle yetersiz kalmaktadır. Bundan dolayı pekiştirmeli öğrenme ve derin öğrenme yaklaşımlarının birleşimiyle derin pekiştirmeli öğrenme yaklaşımı ortaya çıkmıştır.

Pekiştirmeli öğrenmede standart olarak her adımda durum ve aksiyonlar ayrıştırılarak her adıma Q-öğrenme yöntemi uygulanabilir ancak gerçek dünya problemlerinde her adımın ayrıştırılması karmaşık ve maliyetli olacaktır. Yapay sinir ağlarının fonksiyon yaklaştırıcı özelliği (Watkins and Dayan, 1992; Sutton, 1999; Riedmiller, 2005) ile

Q-öğrenme yaklaşımı birleştirildiğinde sürekli durumların bulunduğu kontrol problemleri daha kolay bir şekilde çözülebilmektedir (Levine et al., 2016; Williams, 1992; Peters, 2008; Deisenroth, 2013; Mnih, 2015; Kiumarsi et al., 2018; Xu et al., 2017; Tsurumine et al., 2019).

Ancak derin pekiştirmeli öğrenme algoritmasının dezavantajlı yönleri de mevcuttur. Bunlardan ilki kontrol problemlerinde karşımıza çıkan $u(t)$, zamanın sürekli bir fonksiyonudur. Ancak bunun pekiştirmeli öğrenmede karşılığı $a(t)$ parametresidir. Pekiştirmeli öğrenmede aksiyonlar birbirlerinden ayrık değerde olmasından dolayı, pekiştirmeli öğrenme ile kontrol problemleri çözülmeye çalışıldığında derin Q-öğrenme algoritması yeterli bir çözüm üretemeyebilir (Kohl and Stone, 2004; Pastor et al., 2009).

Derin Q-öğrenme algoritmasının akış diyagramı şekil 30'da görüldüğü gibidir.



Şekil 30 Derin Q-Öğrenme Algoritması Akış Diyagramı

Derin pekiştirmeli öğrenme algoritması ilk olarak 2015 yılında önerilmiştir (Levine et al., 2016). Denklem 69'da Q-öğrenme formülüne bakıldığında

$$Q(s,\alpha) = Q(s,\alpha) + \alpha(r_s + \gamma \max_{\alpha \epsilon A} Q(s',\alpha) - Q(s',\alpha)) \qquad (69)$$

İteratif olarak güncellendiği görülmektedir. Ancak sürekli durum değerlerin herhangi bir iteratif çözümü bulunmamaktadır. Derin Q-öğrenme algoritmasının temelinde

fonksiyon yakınlaştırma kullanıldığından denkleme fonksiyon yaklaştırma tekniği uygulanmalıdır. Bu fonksiyon denklem 70'de gösterilmektedir.

$$Q(s,\alpha:\theta) \approx Q^{*}(s,\alpha) \tag{70}$$

Denklem 70'de θ parametresi derin sinir ağlarının parametrelerini temsil etmektedir. Denklem 69 Denklem 70'deki gibi tekrar oluşturulduğunda aşağıdaki denklem ortaya çıkar.

$$Q(s,\alpha) = Q(s,\alpha) + \alpha(\underbrace{\underbrace{r_s + \gamma \max_{\alpha \epsilon A} Q(s'\alpha')}_{Hedef} - \underbrace{Q(s,\alpha))}_{Tahmin}}_{Maliyet} \tag{71}$$

Eğer denklem 71 iki farklı parçaya ayrılırsa denklem 72'teki denklem grubu ortaya çıkar.

$$\tilde{\gamma} = r_s + \gamma \max_{\alpha \epsilon A} Q(s'\alpha':\theta) \tag{72}$$
$$\tilde{\gamma} = Q(s,\alpha:\theta)$$

Denklem 72'de ilk parça hedefi, ikinci parça tahmin edilen değeri açıklar. Denklem 72 baz alınarak maliyet fonksiyonu oluşturulmak istendiğinde denklem 73 elde edilir.

$$L_i(\theta_i) = E_{s,\alpha,r}[(\tilde{\gamma} - Q(s,\alpha,\theta_i))^2] \tag{73}$$

Denklem 64'e maliyet fonksiyonunun gradyanı eklendiğinde denklem 74 elde edilir.

$$\nabla_{\theta_i} L(\theta_i) = E_{s,\alpha,r,s'}\left[(r + \gamma \max_{\alpha \epsilon A} Q(s',\alpha':\theta_{i-1}) - Q(s,\alpha:\theta_i))\nabla_{\theta_i} Q(s,\alpha:\theta_i)\right] \tag{74}$$

Sırasıyla denklem 73 ve denklem 74 kullanılarak derin Q-öğrenmenin algoritması oluşturulabilir (Mnih et al, 2015).

Derin pekiştirmeli öğrenmenin algoritması şekil 31'de verilmiştir.

ALGORİTMA : Derin Q Öğrenme Algoritması

1 : D, C *Hafıza alanı ve C güncelleme adım parametrelerini oluştur.*

2 : $\tilde{Q}(s, a: \theta)$ *Tahmin değerini rastgele üretilen* θ *parametreleriyle hesapla.*

3 : $\hat{Q}(s, a: \theta)$ *Hedef değerini rastgele üretilen* θ *parametreleriyle hesapla.*

4 : $Eps \leftarrow 0$

5 : s *Ajanın başlangıç durumu.*

6 : $t \leftarrow 0$

7 : ε *olasılıkla rastgele* a_t *aksiyon seç.*

8 : $1-\varepsilon$ *olasılıkla* $a_t = \underset{a \in A}{argmax} Q(s_t, a_t: \theta)$ *aksiyon seç.*

9 : a_t *aksiyonunu sisteme uygula ve* r_t, s_{t+1} *ödül ve bir sonraki durumu hesapla.*

10 : $D = (s_t, a_t, r_t, s_{t+1})$ *Belleğe verileri kaydet.*

11 : $D = (s_j, a_j, r_j, s_{j+1})$ *Bellekten rastgele veriler oluştur.*

12 : $$\hat{y} = \begin{cases} r_j & t = j+1 \\ r_j + \gamma \max_{a \in A} \hat{Q}(s_{j+1}, a, \theta_{t-1}) & dd \end{cases}$$

13 : $(\hat{y} - \tilde{Q}(s_j, a_j: \theta_t))^2$ *Maaliyet fonksiyonunu gradyan azalan ile optimum yap.*

14 : *Her C adımda bir* $\hat{Q} = \tilde{Q}$

Şekil 31 Derin Q-Öğrenme Algoritması

11. Derin Deterministik Politika Gradyan Algoritması

Derin deterministik politika gradyan algoritması, derin politika gradyan ve derin Q-öğrenme yönteminin birleştirilmesiyle oluşturulmuş bir algoritmadır. Derin Q-öğrenme algoritması sürekli durum ve birbirinden ayrık aksiyon kümeleri üzerinde çalıştığından dolayı sürekli aksiyon uzayına uygulanamaz. Örneğin 7 eklem bölgeli bir robot kol ele alındığında, robot kolun eklemlerinin uygulayacağı her bir aksiyon 3 boyutlu bir çalışma uzayınca 3^7=2187 olacaktır. Aksiyon alanındaki yapay zeka ajanlarının atacağı her adımda aksiyon üstel olarak büyüyecektir. Bundan dolayı da aksiyon uzayı çok fazla büyüyecek ve yapay zeka ajanlarının ortamı keşfetmesi imkansız hale gelecektir. Ayrıca yapay zeka ajanlarının öğrenme gerçekleştireceği uzayı ayrıklaştırdığımızda bu durum bilgi kaybına sebep olacaktır. Bu sebepten dolayı derin deterministik politika gradyan algoritması ortaya çıkmıştır. Derin deterministik politika gradyan algoritması politika tabanlı bir algoritmadır ve yapay zeka ajanlarının aksiyon alacakları ortamda derin öğrenmeden gelen fonksiyon yaklaştırıcı yöntemi sayesinde ayrık uzay problemini çözmektedir.

Robot kol uygulamalarında, robot kolun eklemleri ve kolun hareket edeceği alan değişkeni parametreleri Derin deterministik politika algoritmasına giriş verisi olarak eklenecek ve sonuç olarak uygun politika değerleri elde edilecektir (Lillicrap et al., 2015).

Standart pekiştirmeli öğrenme fonksiyonu denklem 75'te gösterilmiştir.

$$Q^{\pi}(s_t, \alpha_t) = E[r(s_t, \alpha_t) + \gamma Q^{\pi}(s_{t+1}, \alpha_{t+1})] \tag{75}$$

Eğer denklem 75'e deterministik politika yöntemi de eklenirse denklem 76 elde edilecektir.

$$\begin{gathered} L(\theta^Q) = E[(Q(s_t, \alpha_t : \theta^Q) - \gamma_t)^2] \\ \gamma_t = r(s_t, \alpha_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}) : \theta^Q) \end{gathered} \tag{76}$$

Denklem 76'a gradyan tabanlı bir güncelleme eklenirse denklem 77'e ulaşılır

$$\begin{gathered} \nabla_{\theta^{\pi}} J = E[\nabla_{\theta^{\pi}} Q(s, \alpha | \theta^Q) | s = s_t, \alpha = \pi(s_t | \theta^{\pi})] \\ \nabla_{\theta^{\pi}} J = E\left[\nabla_{\theta^{\pi}} Q(s, \alpha \middle| \theta^Q) | s = s_t, \alpha = \pi(s_t) \nabla_{\theta_{\pi}} \pi(s_t | \theta^{\pi}) | s = s_t\right] \end{gathered} \tag{77}$$

Denklem 77' aynı zamanda deterministik politika gradyanı algoritmasının fonksiyonudur.

Pekiştirmeli öğrenmeye derin yapay sinir ağları eklendiğinde model oluşturmak ve algoritmanın performansını arttırmak için veriler arasındaki korelasyonu en aza indirmek gereklidir. Ancak pekiştirmeli öğrenmede yapay zeka ajanlarının topladığı veriler arasında yüksek bir korelasyon bulunmaktadır. Bundan dolayı derin Q-öğrenme yönteminde her iterasyonda oluşturulan bellek alanı aynı şekilde derin deterministik politika gradyanı yönteminde de oluşturulmalı ve bu belleklere toplanan veriler birbirlerinden bağımsız şekilde seçilip buna göre eğitim gerçekleştirilmelidir.

Derin deterministik politika gradyan algoritması şekil 32'de gösterilmiştir.

ALGORİTMA : Derin Deterministik Politika Gradyanı Algoritması

1 : $Q(s,a|\theta^Q), \pi(s|\theta^\pi)$ *Kritik ve aktör yapay sinir ağlarını rastgele parametreler* θ^Q, θ^π *ile başlangıcını oluştur.*

2 : $\hat{Q}, \hat{\pi}$ *Hedef kritik ve aktör yapay sinir ağlarını* $\theta^{\hat{Q}} \leftarrow \theta^Q, \theta^{\hat{\pi}} \leftarrow \theta^\pi$ *parametreleriyle başlangıç değerlerini oluştur.*

3 : D *Bellek alanını oluştur.*

4 : M *Algoritmanın kaç bölüm çalışacağını oluştur.*

5 : T *Algoritmanın her bir bölüm içerisinde kaç adım atacağını oluştur.*

6 : $Bölüm = 1:M$

7 : *Ajanın kefiş oluşturması için,* T *adet N rastgele değer üret.*

8 : *Başlangıç durumunu* s_1 *oluştur.*

9 : $t = 1:T$

10 : *Aksiyon seçimi* $a_t = \pi(s_t|\theta^\pi) + N_t$ *oluştur.*

11 : *Aksiyonu* a_t *ortama uygula* ve r_t, s_{t+1} *değerlerini gözlemle.*

12 : $D = (s_i, a_i, r_i, s_{i+1})$ *Değerlerini hafızaya kaydet.*

13 : *Hafızadan P adet uniform örnek değerler oluştur.*

14 : $y_i = r_i + \gamma \hat{Q}(s_{t+1}, \hat{\pi}(s_{t+1}|\theta^{\hat{\pi}})|\theta^{\hat{Q}})$ *Değerini oluştur*

15 : $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\theta^Q))^2$ *Kritik* yapay sinir ağını güncelle.

16 : $\nabla_{\theta^\pi} J = \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q(s,a|\theta^Q)\,|s = s_i, a = \pi(s_i) \nabla_{\theta^\pi}\pi(s|\theta^\pi)|s_i$ *Aktör yapay sinir ağını politika gradyanı ile güncelle.*

17 : $\theta^{\hat{Q}} \leftarrow \tau\theta^Q + (1-\tau)\theta^{\hat{Q}}$

$\theta^{\hat{\pi}} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\hat{\pi}}$ *Hedef yapay sinir ağlarının parametresini güncelle.*

Şekil 32 Derin Deterministik Politika Gradyanı Algoritması

C. Federe Öğrenme

Derin pekiştirmeli öğrenmede, durum özellik alanı küçük ve eğitim verileri sınırlı olduğunda yüksek kaliteli politikalar oluşturmak zordur. Derin pekiştirmeli öğrenmede önceki transfer öğrenme yaklaşımlarının başarısına rağmen, veri ve/veya modellerin gizliliği nedeniyle veri ya da modellerin bir ajandan diğerine doğrudan transferine genellikle mahremiyet bilincine sahip birçok uygulamada izin verilmez. Bundan dolayı Federe Öğrenme yöntemi ortaya çıkmıştır.

Bu yöntemde aracılar arasında sınırlı bilgi (yani Q-ağı çıktısı) paylaşarak her aracı için özel bir Q-ağ politikası öğrenmeyi amaçlar. Bilgi, başkalarına gönderildiğinde "kodlanır" ve başkaları tarafından alındığında "kod çözülür". Bazı ajanların durumlara ve eylemlere karşılık gelen ödülleri olduğunu, diğer ajanların ise ödülleri olmadan durumları gözlemlendiği bu yöntemde kabul edilebilir. Ceza olmadan, bu

temsilciler kendi bilgilerine dayalı olarak karar yönergeleri oluşturamazlar. Politika kararları alırken tüm temsilcilerin bir koalisyona takılmalarından fayda sağlanır.

Birleşik pekiştirmeli öğrenmenin birçok uygulaması mevcuttur. Örneğin imalatta, bir ürünün imalatı, ürünün farklı bileşenlerini üreten farklı fabrikaları içerebilir. Fabrikanın karar verme politikası özeldir ve paylaşılmaz. Öte yandan, sınırlı işleri ve (bazı fabrikalar için) ücret eksikliği nedeniyle, bireyler için yüksek kaliteli karar verme yönergelerini bağımsız olarak geliştirmek genellikle zordur. Bu nedenle kişisel verilerin açıklanmaması durumunda karar alma politikalarını federe bir şekilde gözden geçirmelerinde fayda vardır. Başka bir örnek, hastaneler için hasta bakım politikaları geliştirmektir. Hastalar bazı hastanelerde tedavi edilebiliyor ve tedavi hakkında hiçbir zaman geri bildirimde bulunmuyor olabilir, bu da bu hastanelerin hastaları tedavi etmek ve hastalar için tedavi kararları vermek için ödüllendirilmediğini düşündürür. Ayrıca hasta kayıtları özeldir ve hastane ile paylaşılmaz. Bu nedenle, federe düzeyde hastane tedavi politikalarının gözden geçirilmesine ihtiyaç vardır.

Federe derin öğrenme algoritması, küresel bir durumu (veya doğrudan "küresel" bir durum oluşturmak için kullanılan alt durumları) gözlemleyen, bireysel eylemleri seçen ve ekip ödülleri (veya her ajan, diğer ajanlarla paylaşılan bireysel ödül alır) (Mnih et al., 2013). FedRL, acentelere bazı gördüklerini paylaşmamalarını ve bazı acentelere ödül almamalarını tavsiye eder. FedRL çerçevesi genel itibariyle, gözlemlerin birbirleriyle paylaşılması şartıyla (Cao, 2019), ilgili ancak farklı görevlerin veya aracıların öğrenme performansını iyileştirmek için öğrenme deneyimini görevlere aktarmayı amaçlayan pekiştirmeli öğrenmede transfer öğrenmeden de farklıdır (Liu et al., 2019). FedRL algoritması politikanın aracılar arasında paylaşılamayacağını öne sürmektedir.

Dağıtılmış öğrenme modellerinin amacı, özellikle bilgi işlem gücünün paralelleştirilmesi iken, federe öğrenme modellerinin amacı, öncelikle heterojen veri kümeleri üzerinde eğitim yapmaktır. Hem dağıtılmış öğrenme modelleri hem de federe öğrenme modelleri, modelleri birden çok sunucuda eğitir. Bununla birlikte, öğrenme modellerinde genel bir varsayım, yerel veri kümesinin düzgün dağıldığı ve sınıfların kabaca aynı büyüklükte olduğudur (Mnih et al., 2015; Duan et al., 2016). Buna karşılık, birleşik bir öğrenme modelinde, veri kümesi heterojen olabilir ve sınıflar dengesiz olabilir. Bu nedenle, dağıtılmış bir öğrenme modeli durumunda, merkezi sunucu tipik olarak yerel istemcilerden toplanan ortalama gradyan

güncellemelerinin ortalamasını alır ve daha sonra bunun yerine güncellenmiş modeli merkezi sunucudan indirebilir.

Aktörlerin kendi ağ parametreleri vardır θ. Çalışanlar, çevresel gözlemler göz önüne alındığında yapılacak öğrenme görevlerini ve eylemleri gerçekleştirmek için katılımcı ağları kullanır. İşçiler, katılımcı ağı tarafından tanımlanan eylemleri gönderir ve ortamın bir sonraki durumunu gözlemler. Eylemler sonucunda çalışanlar olumlu veya olumsuz ödüller alırlar. Eleştirmenler, eylem ödüllerini ağ parametreleri olarak görüyor. Eleştirmen, işçilerle birlikte, aktör tanımlı eylemlerin ortamı daha olumlu hale getirip getirmediğini değerlendirmeyi öğrenir ve eleştirmenin geri bildirimi, oyuncuyu optimize etmek için kullanılır.

FedRL çerçevesi üç aşamada çalışmaktadır. İlk olarak, her ajan Gauss farklılaşması kullanılarak "karşılaştırılan" diğer ajanlardan Q ağlarının çıktı değerlerini toplar. Ayrıca, Q ağının global çıktısını hesaplamak için yerel Q ağı çıkışını ve şifrelenmiş değerleri girdi olarak alan MLP (multilayer perceptron) gibi paylaşılan bir değer ağı oluşturur. Son olarak, global Q-ağının çıktısına dayalı olarak paylaşılan değer ağını ve kendi Q-ağını günceller. MLP'nin aracı tarafından paylaşıldığını ve aracının kendi Q-ağının başkaları tarafından bilinmediğini ve eğitim sırasında paylaşılan şifreli Q-ağının çıktısından türetilmemesi gerekmektedir.

Markov karar süreci daha önceki maddelerde de bahsedildiği gibi *(S,A,T,r)* şeklinde tanımlanabilir. Burada S durum uzayını, A aksiyon uzayını, T ise geçiş fonksiyonunu temsil etmektedir. $S \times A \rightarrow S$, *i.e.*, $T(s,a,s') = P(s' \mid s, a)$, durum uzayı için geçerli olan, sonraki durum olasılığını belirtir, $s' \in S$ mevcut durumunu belirtir. *r*, ödül fonksiyonu ve $S \rightarrow R$, burada *R* gerçek sayıların alanıdır. π eldeki politikayı temsil eder. Ve $V^{\pi}(s)$ fonksiyonu ve $t + 1$ adımındaki Q-fonksiyonu aşağıdaki denklemler ile güncellenebilir:

$$V_{t+1}^{\pi}(s) = r(s) + \sum_{s' \in S} T(s, \pi(s), s') V_t^{\pi}(s'), \tag{78}$$

Ve

$$Q_{t+1}^{\pi}(s,a) = r(s) + \sum_{s' \in S} T(s, a, s') V_t^{\pi}(s'), \tag{79}$$

Burada $t \in \{0,\ldots,K-1\}$ koşulu sağlanmalıdır. Markov sürecinin çözümü π^* en iyi politika ile $V^{\pi^*}(s) = max_{\pi} V^{\pi}(s)$ ya da $Q^{\pi^*}(s, \pi^*(s)) = max_{\pi} Q^{\pi}(s, \pi(s))$ şeklinde olacaktır. Ancak derin öğrenmede geçiş türevi *T* bilinmediğinden dolayı Q-ağı

Q(s, a; θ) fonksiyonu ile güncellenir ve burada θ parametresi ağın parametresini temsil etmektedir. Aşağıdaki denklem ile güncellenir:

$$Q_{t+1}(s,a;\theta) = E_{s'}\left\{\gamma \max_{a' \in A} Q_t(s',a';\theta)|s,a\right\} \quad (80)$$

Çalışmada araştırmacılar tarafından yapıldığı gibi, θ parametrelerini öğrenmenin bir yolu *(s,a, s', r)* geçişlerini yeniden yürütme belleklerinde Ω depolamak ve *θ'*yı tekrar tekrar güncellemek için bir toplu örneklemden yararlanmaktır (Smart ve Kaelbling, 2002). Θ bir kez öğrenildiğinde π^* ilkesi *Q(s, a; θ)* ' den çıkarılabilir.

$$\pi^*(s) = arg \max_{a \in A} Q\,(s,a;\theta) \quad (81)$$

Federe derin pekiştirmeli öğrenme problemi şu şekilde tanımlanır:

$D\alpha = \{(s\alpha , a\alpha , s'_\alpha , r\alpha)\}$ ajan a tarafından toplanan durum ve eylem çiftleri $D\beta = \{(s\beta,a\beta)\}$ ajan β tarafından toplanan durum ve eylem çiftleri. π_α^* ve π_β^* politikalarının federatif olarak oluşturulmasını amaçlıyoruz.

Birleşik derin pekiştirmeli öğrenme probleminde aşağıdaki maddeler varsayılmaktadır.

A1: $s\alpha$ ve $s\beta$ durumlarının özellik uzayları, α ve β ajanları arasında farklıdır. Örneğin, bir $s\alpha$ durumu, bir hastanın α hastanesindeki kardiyogramını belirtirken, başka bir $s\beta$ durumu, aynı hastanın β hastanesindeki elektroensefalogramını belirtir ve $s\alpha$ ve $s\beta$ 'nin özellik uzaylarının farklı olduğunu gösterir.

A2: $D\alpha$ ve $D\beta$ geçişleri, kendi modellerini öğrenirken α ve β arasında doğrudan paylaşılamaz. Bununla birlikte, $D\alpha$ ve $D\beta$ geçişler arasındaki yazışmalar birbirleri tarafından bilinmektedir. Başka bir deyişle ajan α, ajan β'a bir geçişin ID'sini gönderebilir ve ajan β, Dβ'de karşılık gelen geçişi bulmak için bu ID'i kullanabilir. Örneğin, hastanede ID belirli bir hastaya karşılık gelebilir.

A3: $Q\alpha$ ve $Q\beta$ işlevlerinin çıktıları, bazı gizlilik koruma mekanizmaları tarafından korunmaları koşuluyla birbirleriyle paylaşılabilir

A1,2,3 maddelerine dayanarak, veri ve modellerin gizliliğini koruyarak tüm ajanlar için yüksek kaliteli politikaları öğretme amaçlanmaktadır.

MLP ile birleştirilmiş federe Q-ağının fonksiyonu aşağıdaki gibidir.

$$Q_f^a(.,C_\beta;\theta_a,\theta_g) = MLP([\hat{Q}_a(.;\theta_a)|C_\beta];\theta_g) \quad (82)$$
$$Q_f^\beta(.,C_a;\theta_\beta,\theta_g) = MLP([C_a|\hat{Q}_\beta(.;\theta_\beta)];\theta_g)$$

Burada θg MLP'nin parametreleridir ve [.|.] birleştirme işlemini gösterir. Bir MLP'nin parametreleri aracılar arasında paylaştırılabilir. MLP güncellendiğinde güncellenen parametreler diğer aracılarla paylaştırılır.

Ajanlar ile ilgili olarak, aşağıda gösterildiği gibi, kendi temel Q-ağını güncellerken diğer ajanın temel Q-ağını (Gauss gürültüsü ile) sabit olarak görüntüleyerek her ajanın birleşik Q ağları tanımlanır.

$$Q_f^a(.,C_\beta;\theta_a,\theta_g) = MLP([\hat{Q}_a(.;\theta_a)|C_\beta];\theta_g) \quad (83)$$
$$Q_f^\beta(.,C_a;\theta_\beta,\theta_g) = MLP([C_a|\hat{Q}_\beta(.;\theta_\beta)];\theta_g)$$

Burada $C\alpha = \hat{Q}\alpha(s\alpha,a\alpha;\theta\alpha)$ ve $C\beta = \hat{Q}\beta(s\beta,a\beta;\theta\beta)$ fonksiyonları sırasıyla ajan β' nin Q-ağı ve ajan α'nın Q-ağı güncellerken sabittir.

Ajan α ve ajan β nin Q-ağlarının kare hata kaybı $\dot{L^j}\alpha(\theta\alpha,\theta g)$ ve $\dot{L^j}\beta(\theta\beta,\theta g)$ fonksiyonları minimize edilerek eğitilir. Ve son olarak aşağıdaki denklemlere ulaşılır

$$L_a^j(\theta_a,\theta_g) = E[(Y^j - Q_j^a(s_a^j,a_a^j,C_\beta;\theta_a,\theta_g))^2] \quad (84)$$
$$L_\beta^j(\theta_\beta,\theta_g) = E\left[(Y^j - Q_f^\beta(s_\beta^j,a_\beta^j,C_a;\theta_\beta,\theta_g))^2\right]$$

Federe öğrenmenin basit bir algoritması şekil 33 ve 34'te gösterilmiştir.

ALGORİTMA: Federe Derin Öğrenme Algoritması [Sunucu]

1 : *Başla* w_0
2 : *for her t = 0,1,.. do*
3 : $M \leftarrow max\ (\ [C\ .\ K],\ 1)$
4 : S_t = *rastgele m katılımcı ayarla*
5 : *for katılımcı için* $k \in S_t$ *in parallel do*
6 : w_{t+1}^k = *Yerel Güncelle (k,* w_t*)*
7 : $w_{t+1} = \sum_{k\in St} \frac{nk}{n\sigma} w_{t+1}^k, \quad n_\sigma = \sum_{k\in St} nk$

Şekil 34 Kullanılacak Federe Derin Öğrenme Algoritması Sözde Kodu (Sunucu)

ALGORİTMA: Federe Derin Öğrenme Algoritması [Yerel]

1 : B = *yerel minibatch boyutu*
2 : m = *yerel katılımcı sayısı*
3 : E = *eğitim turu*
4 : n = *öğrenme oranı*
5 : *Başla*
6 : $w_0 \leftarrow$ *rastgele başlatma*
7 : *{iletişim boyunca}*
8 : *for* $t = 1,..,T,\ldots$ *do*
9 : $S_t \leftarrow$ *(rastgele alt küme – max (C x K,1) katılımcı)*
10 : *{her katılımcı için yerel optimizasyon}*
11 : *for katılımcı* $k \in S_t$ *do*
12 : *yerel ağırlıkları başlat:* $w_{t,k} \leftarrow w_{t-1}$
13 : *for epoch* $e \in [1,E]$ *do*
14 : *Yerel verileri böl,* $B(\frac{B}{nk}$ *B yığınları)*
for yığın $b \in B$ *do*
15 :
16 : $w_{t,k} \leftarrow w_{t-k} - n_{local}\, \Delta l\, (w_{t-k}\,;\, b)$
17 : *end for*
18 : *end for*
19 : *end for*
20 : *{Merkezi Ortalama}*
21 : $w_t \sum_{k \in St} \frac{nk}{n} w_{t,k}$
22 : *end for*

Şekil 33 Kullanılacak Federe Derin Öğrenme Algoritmasının Sözde Kodu (Yerel)

III. ROBOTLARDA PEKİŞTİRMELİ ÖĞRENME

Pekiştirmeli öğrenmeye göre, gelişen ve öğrenen robot sistemi, harekete geçerken ve karar alırken her zaman ortam ile etkileşim içerisindedir. Öğrenme yeteneği olan robot etkenleri anlayıp analiz ederek, en iyi eylemi seçmek için çalışır. Bu robotun analiz yeteneğine örnek olarak düşünülebilir. Bu işlemleri gerçekleştirirken her seferinde bulunduğu ortamdan deneme yanılma adı verilen iyi ve kötü dönüşler alır. Bu şekilde robot, en iyi sonuca ulaşmış olur. Bununla birlikte gerçek dünyadaki pekiştirmeli öğrenmenin uygulamaları, genellikle öğrenme yönteminin kendisinin ötesinde önemli teknik gelişmeler gerektirmektedir: fiziksel donanım için pratik olan öğrenme sürelerini de elde etmek için politika ya da değer işlevi için uygun bir temsil seçilmelidir (Bicchi, 1995; Platt, 2007). Ancak gerçek dünyadaki robotik çalışmalarına derin pekiştirmeli öğrenme yöntemlerinin uygulanmasındaki temel zorluklardan birisi de yüksek örnek karmaşıklığı olmuştur (Bicchi, 1995; Stulp et al., 2011)

Bu öğrenme yönteminde ortamın özellikleri robota tanımlanmalıdır. Deneme yanılmadan alınan sonuçlar performansı ölçümlemek için kullanılır. Bu ölçümleri etkileyen tüm değişkenler ortam olarak düşünülebilir (Russel and Norvig, 1995).

Bu robotlar her seferinde yeni şeyler öğrenmeyebilir. Bunun yerine geçmiş bilgilerini de kullanabilir. Bunlar kompleks olabileceği gibi basit şeyler de olabilir. Öğrenebilen robotun hedefe giderken geçmiş tecrübelerinden faydalanıp faydalanmayacağını ya da yeni olasılıklar bulup bulamayacağını yapılan plan belirler. Ortam ile öğrenebilen robot arasında yaygın kabul edilen ilişkiler şekilde gösterilmiştir (Kulkarni, 2012).

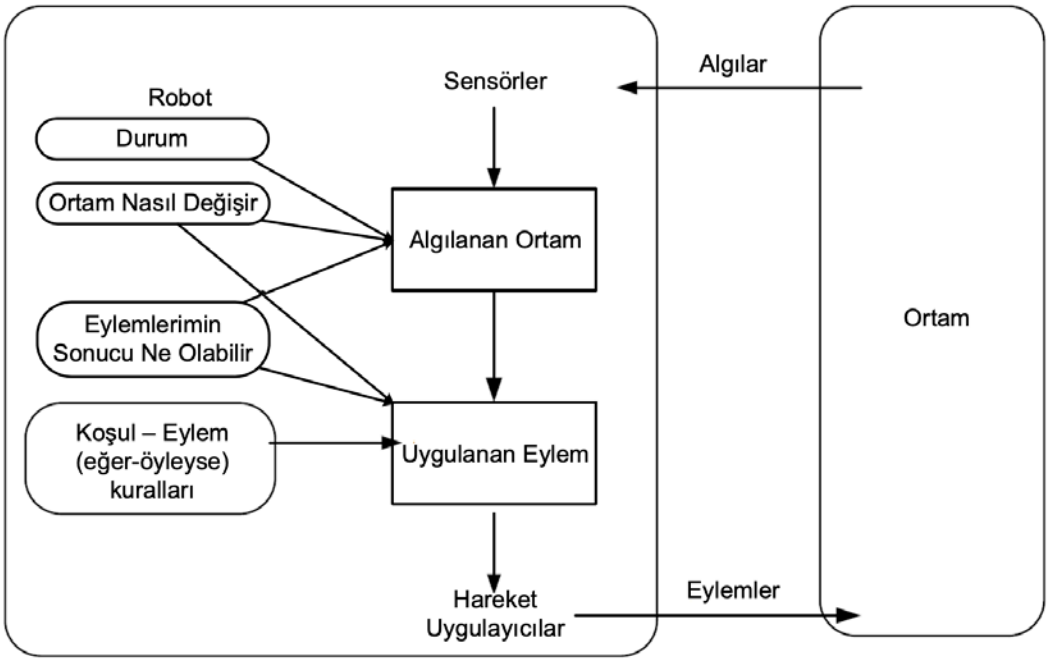
Bir robotun öğrenebilen rasyonel robot sayılabilmesi için, ortamdaki karmaşık problemleri tespit edip onları çözebilmesi gerekmektedir. Bu robotların sahip olması gereken özellikler de şöyle sıralanabilir:

1. Sürekli veya belirli aralıklarla ortamlardan veri toplamalı
2. Tek başına karar almalı ve bu kararları uygulayabilmeli
3. Farklı bilgileri bir araya getirerek, tekil bilgilerle daha geniş bir bilgi havuzu oluşturabilmeli.

4. Sürekli öğrenme yeteneği olmalı
5. Daha önce edindiği tecrübelerle yeni bilgiler edinebilmeli.

Eğer ortamda sürekli bir değişim varsa, sensörlerden alınan veriler yeterli değilse ya da yeterli olsa bile temiz ve anlaşılabilir değilse, robot için durum karmaşıklaşabilir. Bu durumlarda robot inisiyatif almalı ve geçmiş tecrübelerinden de yola çıkarak gerekirse bulunduğu eylemleri ödül ve ceza deneyimlerinin arasına katabilmelidir.
Bu şekilde esnek hareketlerde bulunmak için robotta şu özellikler olmalıdır:

1. Responsive olmalıdır. Elde ettiği datalara göre olaylara zamanında cevap verebilmelidir.
2. Proactive, yani önsezili olmalıdır. Ansızın gerçekleşen çevresel konulardaki fırsatları görebilmeli ve uygun eylemi seçmelidir.
3. Bulunduğu ortamdaki diğer robotlarla ve kişilerle etki tepki ilişkilerine girebilmeli, yani sosyal olmalıdır.

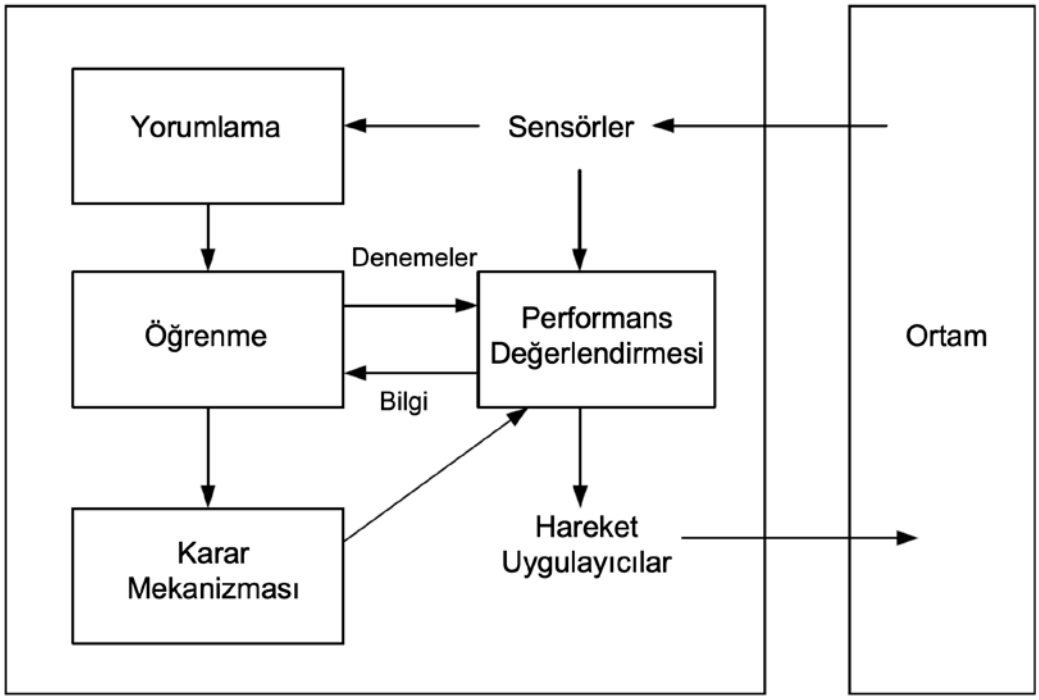


Şekil 35 Akıllı Robot Ortam İlişkisi

A. Öğrenen Robotlar

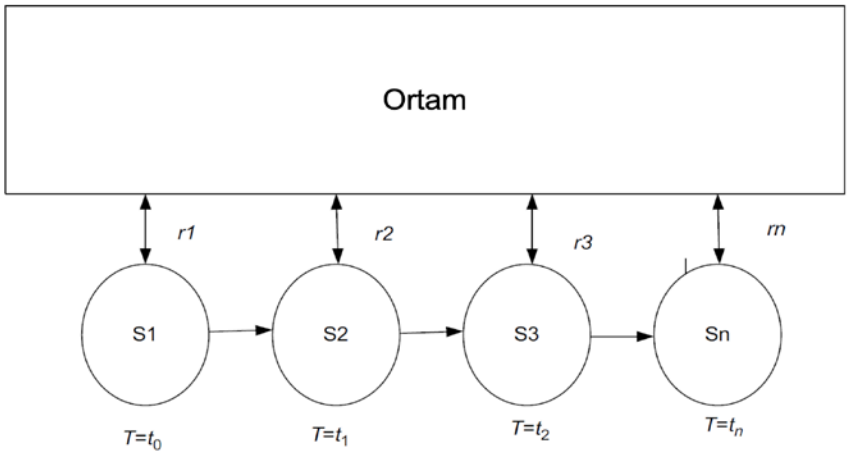
Daha önce de anlatıldığı gibi bir robotun akıllı olarak sınıflandırabilmesi için, tek başına değişken ortamlarda eyleme geçilmesi ve karar verebilmesi gerekliliği bulunur. Robotun geçmiş tecrübelerinden edindiği veriler, çıkarım yapmaya ve doğru hedefe ulaşmasına yeterli olmayabilir. Yani robotlar her karardan sonra kendini geliştirmelidir ve bu değişken ortamlarda mantıklı karar alması sağlanmalıdır.

Bizim dünyamızdaki öğretmen eşliğindeki öğrenme, robotlar için gözetimli öğrenmeye karşılık gelmektedir. Bu sebeple robotlar tahmini modeller geliştirebilmelidir. Bu tahminler çıktıların eylemlerle olan ilişkilerine, rakiplerin verebileceği tepkilerin tahminlerine dayanır. Öğrenebilen bu robotlar bazen de rastgele hareketler yapmak durumunda kalır. Ancak bu eylemler rastgele olsa da sonucuna göre ödül ve ceza datalarının incelenebilir olması gerekir.



Şekil 36 Akıllı Robot Öğrenme Süreci

Robotların dünyasında da tıpkı bizim dünyamızda olduğu gibi ödül ve ceza eylem tamamlandıktan sonra gelmektedir. Ve yine tıpkı bizde olduğu gibi kararın haklılığını ölçmek için çok önemlidirler. Çünkü bu sayede robot verilen kararlarda değişim gerekip gerekmediğini değerlendirir ve eylemlerine bu şekilde devam eder. Robotun toplam ödülünün en yüksek seviyeye ulaşabilmesi için periyodik ödüller önemlidir. Süreç içerisinde ödül gelmezse, robot tahminlere dayalı bir şekilde maksimum ödüle ulaşmaya çalışacaktır.



Şekil 37 Ödülün Ortamdan Durum-Eylem İkilisi Sonrası Nasıl Geldiğinin İlişkisi

Öğrenebilen robotlar, her zaman maksimum ödülü hedefler. Ödülleri bazen belirli periyodlarla bazen de asıl hedefe ulaştığında elde eder. Ödülü belirten denklem 85'deki gibidir.

$$R_T = r_{t-1} + r_{t-2} + r_{t-3} + \ldots + r_T \tag{85}$$

Ortam ile robot arasında bulunan ilişkiyi incelersek, ödül ve eylemlerden sonra elde edilen sonuçlar farklı olacaktır. Sürekli görevlerde kısımlara ayırmak zor olabilir ama periyodik görevlerde bu çok daha kolay olacaktır. Toplam ödül de bu süreçte alınmış ödüllerin hepsinin bir araya gelmesinden oluşacaktır.

B. Robotlarda Eylem Seçim Metodları

Pekiştirmeli öğrenme türünün diğer türlerden en büyük ayrıştığı nokta, ödül ve cezaların talimat olması değil de yorumlanabilmesi ve geçmişteki, gelecekteki pozisyonlara göre şekillenmesidir. Bu şekilde tutarlı bir araştırma mümkün olmaktadır (Sutton ve Barto, 2018). Sadece iyi ve kötü davranış olarak tanımlanmaktan ziyade bu değerlendirmenin sonucunda optimizasyon temelli veriler elde edilir. Başka bir yandan da gözetimli öğrenme türü çok önemli bir yöntemdir. Bu yöntemde ödüller değil de hareketler ve onların sonuçları belirlenir ve aralarında ilişki kurulur.

Durum ve eylem arasındaki ilişkiyi anlamak için, araştırma ve faydalanma iyi dengelenmelidir. Bunu dengelemek için de seçim metodu ve değişkenler iyi belirlenmelidir. Eğer robot sürekli faydalanmaya çalışırsa, yani sürekli aynı yolu seçmeye başlarsa daha hedef için iyi bir yol olsa da robot bunu görmezden gelecektir ya da ödüller düşükse robot bu yola girmez ve hiçbir zaman istenilen sonuç alınamayabilir. Ancak robot sürekli gelişir ve tek bir yola sadık kalmazsa, ilerleme ve öğrenme mümkün olmaz. İşte bu sebeple araştırma ve faydalanma iyi dengelenmelidir.

IV. İLGİLİ ÇALIŞMALAR

Bu bölümde pekiştirmeli öğrenme, federe öğrenme, robot kollarının cisimlerin tutulabilirliğini öğrenmesi konusunda sunulan farklı çalışmalar incelenecektir.

Çalışmada araştırmacılar, bir robot kolunun hareket planlama ve hareket kontrolü, iki robot kol arasındaki iş birliği ve bir robot kolunun uzaktan kontrolü gibi hizmetler için Mobil Uç Bulutu (MEC) algoritmasını önermişlerdir (Tsokalo et al., 2019). İki robot kolunun iş birliği kullanım durumu, üretim hattındaki karşılıklı çalışma konusunda birden fazla robot kolu arasındaki iş birliğine ilişkin bir tahminde bulunmuşlardır. Çalışmada araştırmacılar otomobil endüstrisinde algılama, kontrol ve üretim için yeni mi al ve yerleştir robotu önermişlerdir (Smys and Ranganathan, 2019). Alma ve yerleştirme robotları, çeşitli endüstrilerde montaj, paketleme, kutu paketleme ve denetimde kullanılmaktadır.

Robotik öğrenmede, dinamik sistemlere dayalı motor ilkeleri (Schaal et al., 2007; Ijspeert et al., 2002) hem taklit hem de pekiştirmeli öğrenme yoluyla yeni davranışların hızlı ve güvenilir bir şekilde kazanmasına izin vermektedir. Örnekler etkileyici olsa da görevi yeniden öğrenmeden bir motor ilkelinin deneme yanılma yoluyla farklı bir davranışa nasıl genelleştirilebileceğini ele almamışlardır. Örneğin bir fincanda top (Kober and Peters, 2011) hareketinde ip uzunluğu değiştirilmişse, hareket parametreleri de değiştirilecek ve davranış yeniden öğrenilmelidir. Birkaç santimetrelik bir dizi uzunluğu varyasyonları nedeniyle davranışın büyük miktarda değişmeyeceği göz önüne alındığında, öğrenilen davranışı değiştirilmiş göreve genellemek daha iyi olacaktır. Davranışların bu tür genelleştirilmesi, hareket temsilinin meta-parametrelerinin uyarlanmasıyla başarılabilir.

Takviyeli öğrenmede, görevler arasında genelleme yapmak için meta-parametreleri kullanmaya yönelik birçok girişimde bulunulmuştur (Caruana, 1997). Özellikle, şebeke dünyası etki alanlarında, politikaların meta-parametrelerini değiştirerek ve ayarlayarak önemli bir hızlanma elde edilebilir (McGovern and Barto, 2001). Robotikte, bu tür meta-parametre öğrenimi, yüksek boyutlu durumlar ve eylenmeler karmaşık motor beceriler için pekiştirmeli öğrenmenin karmaşıklığı nedeniyle yararlı olabilir.

Araştırmacılar bu (Gu et al, 2017) çalışmada, politika dışı derin Q-öğrenme ölçeklerine dayanan yeni bir derin pekiştirmeli öğrenme algoritmasının karmaşık 3B manipülasyon problemlerine ölçeklendiğini ve derin sinir ağı stratejilerinin öğrenme için çok etkili bir şekilde kullanıldığını ve gerçek fiziksel dünya üzerinde geliştirilebileceğini göstermektedirler. İlke güncellemelerini eş zamansız olarak toplayan birden çok robotta algılamayı paralel hale getirmişler ve bu sayede eğitim süresini de azaltılabileceğini vadetmişlerdir. Çalışmada araştırmacılar yöntemlerinin simülasyonda çeşitli 3B manipülasyon yeteneklerinin yanı sıra gerçek robotlarda ön izleme ve elle oluşturma olmadan kapı açma yeteneği kazandırılabileceğini gösterdiğini iddia etmektedirler. Bu makalenin ana katkısı, bir robotik küme üzerinde paralel NAF algoritmalarını kullanarak asenkron derin pekiştirmeli öğrenmeyi göstermektir. Araştırmacıların gerçekleştirdiği önceki çalışmada (Levine et al., 2016) model tabanlı pekiştirmeli öğrenme ve model tabanlı olmayan pekiştirmeli öğrenme yaklaşımlarını incelemişlerdir. Model tabanlı algoritmalar, Gauss süreçleri (Deisenroth and Rasmussen, 2011), karışım modelleri (Moldovan et al., 2015) ve yerel doğrusal öğrenme algoritmalarını (Lioutikov et al., 2014) incelemişlerdir. Derin sinir ağı stratejileri, model tabanlı yöntemler tarafından derin ağ stratejilerini öğrenmek için kullanılan yönetimli politika arama algoritması (Levine et al., 2016) bağlamında model tabanlı öğrenme ile birleştirilir. Bu tür yöntemlerin bir dizi gerçek dünya probleminde başarılı olduğu ve yüksek öğrenme verimliliği sağladığı gösterilmiştir. Derin pekiştirmeli öğrenmede simülasyon deneylerini hızlandırmak için paralel öğrenme yöntemi de önerilmiştir (Mnih et al., 2016). Birden fazla robotun deneyimlerini birleştirerek robotik öğrenmeyi hızlandırmak için de çeşitli araştırmacılar çalışmalarını sunmuşlardır (Inaba et al., 2000; Kuffner, 2010; Kehoe et al., 2013; Kehoe et al., 2015). Ters kinematik problemleri çözmek için uyarlamalı öğrenme algoritmalarının kullanıldığı bir çalışma yapılmıştır (Hasan et al., 2006). Bu çalışmada uçakta kullanılabilecek ve konum kontrolü yapabilen bir manipülatör geliştirilmiştir. Başka bir çalışmada 6 eklem bölgeli çift kollu mobil robot H20'nin konum kontrolü ve farklı özelliklere sahip kavrayıcı uçlarının tasarımını gerçekleştirmişlerdir (Mohammed et al., 2016). Bulanık mantık tabanlı kontrol algoritmalarını destekleyen bir çalışmada, bir robotik kola bulanık mantık tahminlerine dayalı genelleştirilmiş bir kontrol yapısı uygulamışlardır (Cronin et al, 2014). Benzer bir çalışma da 2 eklem bölgeli robot kol için bulanık mantık tabanlı bir

kontrol yapısı geliştirilmiş olsa, bu yapının kayma ve PID kontrol yapısını araştırmak amacıyla karşılaştırmalı bir makale de sunulmuştur (Lochan and Roy, 2014).

Son on yılda, robotikte pekiştirmeli öğrenme uygulaması istikrarlı bir şekilde artmıştır. Schaal (1997) bir robotu ters kinematik polarite dengesi görevinde (göstererek öğrenme) eğitmek için pekiştirmeli öğrenmeyi kullanmıştır. Daha yakın zamanlarda Bhatnagar, Sutton, Gavamzade ve Lee (2009) bu çalışmanın devamını sunmuşlardır. Theodorou, Peters ve Schaal (2007), el hareketlerinin kontrolünü optimize etmek için pekiştirmeli öğrenmeyi kullanmışlar. Peters ve Schaal (2008), robotikte Doğal Aktör Kritik (NAC) algoritmasını sunmuşlardır. Buchli, Thedorou, Stulp ve Schaal (2010), değişken empedans kontrol döngüleri için bir yol entegrasyon yöntemi kullanarak politika tabanlı bir pekiştirme öğrenme yaklaşımını önermişlerdir. Ancak önerilen şemanın etkinliği sadece simülasyonlarda gösterilmektedir. Aktör ve eleştirmen tabanlı robotlar için pekiştirmeli öğrenme hakkında daha fazla bilgi grubun diğer çalışmalarında da bulunabilir (Atkeson and Schaal, 1997; Hoffmann et al., 2012). Daha yakın zamanda, Theodorou, Buchli ve Schaal (2010) yol integrallerini kullanarak kılavuz tabanlı pekiştirmeli öğrenmeyi robot bir köpek üzerinde test etmişlerdir (Kappen, 2005). Digney ve June (1996), robot-çevre etkileşimi için iç içe bir Q-öğrenme algoritması önermişlerdir. Kuan ve diğerleri (1998), sağlam kayan mod empedans kontrolü ile uyumluluk sorunlarını ele almak için bir pekiştirmeli öğrenme mekanizması önermişlerdir ve bu yaklaşımı simülasyonda test etmişlerdir. Çalışmaları, farklı adaptasyon görevlerindeki farklılıkları hesaba katmak için pekiştirmeli öğrenme algoritmalarını kullanmaktadır. Bucak ve Zohdy (1999; 2001), tek bağlantılı ve çift bağlantılı robotlar için bir pekiştirmeli öğrenme yaklaşımı kontrol şeması önermişlerdir. Gaskett (2002) robot kontrolü için Q-öğrenme üzerine çalışmıştır. Smart ve Kaelbling (2002), mobil robotlarda gezinme görevleri için pekiştirmeli öğrenmeyi kullanmıştır. Izawa, Kondo ve Ito (2004), robotik bir kolun iki eklemli ve altı paketli kas-iskelet kolunu en iyi şekilde kontrol etmek için bir pekiştirmeli öğrenme yaklaşımı olan aktör kritik metodu uygulamışlardır. Shah ve Gopal (2009), güvenli olmayan koşullar altında robotik kolların pekiştirmeli öğrenme kontrolünü önermişlerdir. Kim, Park ve Kang (2008; 2010) çevre ile etkileşime girerek farklı durumlara uygun çözümler bulmak için pekiştirmeli öğrenme yöntemlerini kullanmışlardır. Riedmiller, Gabel, Hafner ve Lange (2009), robotik futbol oynamak için toplu pekiştirmeli öğrenmeyi kullanmışlardır. Bu çalışmada değer fonksiyonuna yaklaşmak için bir patlama modu

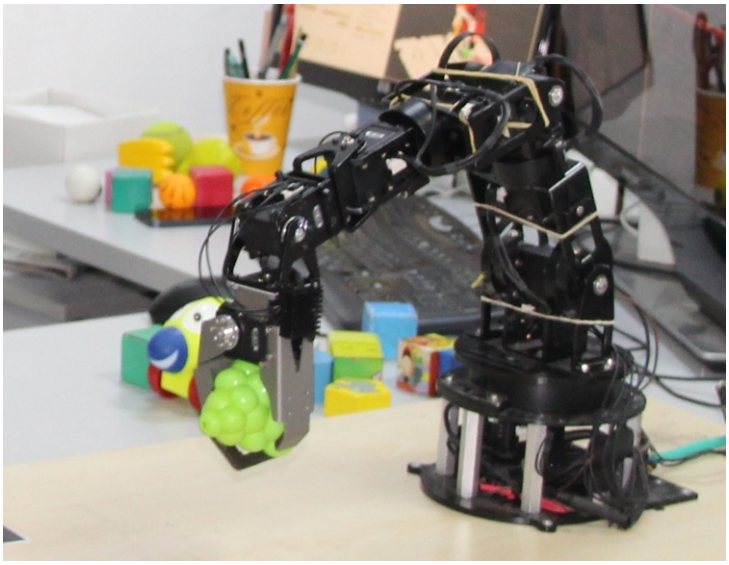
yapısı kullanarak çok katmanlı bir algılayıcı kullanmışlardır. Adam, Busoniou ve Babushka (2012) tarafından yapılan çalışmada Q-öğrenme yöntemlerinin deneysel uygulamasını ve Sarsa örnek tekrarını robotik kaleci ve ters sarkaç örneklerine uygulamışlardır. Stingu ve Lewis (2011), bir quadcopter kullanarak insansız hava aracı (İHA) kullanarak rotayı simüle etmişlerdir. Q-fonksiyonu ve kontrol politikasını modellemek için Sarangapani'nin (2006) çalışmasına dayanan radyal tabanlı fonksiyon sinir ağlarını kullanmışlardır. Araştırmacılar, yararlı öğrenme sonuçları elde etmek için konvolüsyonel sinir ağları (CNN'ler) kullanarak ham piksel girişlerinden kontrol politikalarını öğrenmek için model tabanlı olmayan bir algoritma olan Q-öğrenme yöntemini kullanan bir algoritma göstermişlerdir (Mnih et al., 2013). Çalışmada yeni farklı atari 2600 oyunuyla ortaya koydukları algoritmayı test etmişler ve çalışmaları altı oyunda insan oyuncuyu geçmiştir. Daha yakın zamanlarda araştırmacılar kavram kanıtlamada yakın politika optimizasyonundan daha iyi performans gösteren hiyerarşik eleştiri adlı bir algoritma önermişlerdir (Cao, 2019). Son zamanlarda, hassas kontrol ve doğru sonuçlar gerektiren oyunlara, robotlara ve otonom sürüşe pekiştirmeli öğrenme uygulanmıştır (Sutton and Barto, 1998; Silver et al., 2016; Mnih, 2015; Duan ve diğerleri, 2016; Li ve diğerleri, 2018). Araştırmacılar etkileşimi araştırmak ve ajanlar arasındaki öğrenmeyi paylaşmak için pek çok etmenli ortamda pekiştirmeli öğrenme uygulamışlardır (Tampuu et al, 2017). Çalışmada araştırmacılar, verileri veya bilgileri doğrudan aktarmadan, diğer ajanların yardımıyla her ajan için bir Q-öğrenme ağı oluşturularak veri ve modellerin gizlilik gereksinimleri göz önünde bulundurularak federe bir pekiştirmeli öğrenme yaklaşımı önerilmiştir (Zhuo et al., 2019). Bir ajandan başka bir ajana geçiş çalışmada (Liu et al., 2019) bulut robotlarının navigasyon yardımı için bulutta konuşlandırılmış paylaşılan bir modeli geliştirmek için evrimsel transfer öğrenmeli bir bilgi füzyon algoritması olarak yaşam boyu birleşik pekiştirmeli öğrenme algoritması önermişlerdir. Bonawitz ve ark. (2019) federasyon sürecini tanımladı ve 2019 yılında federasyon politikası için bir federasyon sistemi protokolü tasarladı. Federasyon politikası, her dağıtılmış cihazdaki yerel verileri kullanarak eğitimi gerçekleştirir ve hesaplanan gradyan veya eğitim modelini merkezi sunucuya gönderir. Zhuo ve ark. (2019), bir federasyon politikası kullanarak her ajan için bir Q-öğrenme ağını kullanarak yeni bir pekiştirmeli öğrenme algoritması önermişlerdir. Her aracı, yerel verileri kullanarak kendi Q-öğrenme ağını kullanarak optimal bir Q değeri hesaplar. Çalışmada araştırmacılar federe öğrenmeyi denetimli makine öğrenimi teknikleri

üzerinden gerçekleştirmişlerdir (McMahan et al., 2016). Çalışmada birleşik öğrenme, temel kullanıcı verilerine doğrudan erişim olmaksızın sanal klavye arama öneri kalitesini iyileştirmek için bir modeli eğitmek, değerlendirmek ve dağıtmak için ticari, küresel ölçekli bir ortamda kullanmışlardır (Yang et al., 2018). Denetlenen sınıflandırma görevleri için birleştirilmiş öğrenmenin daha fazla uygulaması (Khan et al., 2012; Chen et al., 2018; Zhou et al., 2018; Smith et al., 2017)'de bulunabilir. Stokastik gradyan iniş (SGD) algoritmasının çeşitli dağıtılmış versiyonu çalışmada önerilmiştir (De and Goldstein, 2016; Chen et al., 2016). Çalışmada yazarlar, istemci modellerinin yanı sıra paylaşılan modeli tek bir büyük Q-öğrenme ağı olarak ele almışlar ve Bellman denklemini kullanarak optimize etmişlerdir (Zhuo et al., 2019). Ancak bu çalışmada, istemcilerin her birine ayrı Q-öğrenmesi vardır ve paylaşılan model parametrelerine bir federasyon politikası karar verir. Çalışmada yazarlar, mevcut çalışmada bir federasyon politikası model birleştirme yöntemi belirlerken, üretici ağlara dayalı bilgi birleştirme algoritmasını kullanmışlardır (Liu et al., 2019). McMahan ve ark. (2016), geleneksel gradyan iniş güncellemelerini uygulamak yerine istemci cihazlardan model ağırlıklarının ortalamasını almak için gereken iletişim turlarının sayısını azaltmayı amaçlayan birleşik ortalama (FEDAVG) algoritmasını sunmuşlardır. FEDAVG, etkinliğini test etmek için büyük ölçekli bir sistemde (Bonawitz et al., 2019) çalışmasında kullanılmıştır.

V. YÖNTEM

Çalışmanın bu bölümünde ortaya konulan algoritmalar ve bu algoritmanın uygulanabilirliği ve uygulama sonuçlarının analizi ve değerlendirilmesi hakkında bilgiler verilecektir.

Söz konusu çalışma daha öncesinde çok fazla odaklanılmamış bir alan ve yeni ortaya çıkmış bir alanı birleştirecektir. Hali hazırdaki bu çalışmada robot kollarının (Şekil 38) federe derin öğrenme ve federe derin pekiştirmeli öğrenme algoritmalarıyla belirli cisimleri tutmaya çalışması üzerinedir. Federe derin öğrenme ile robot kollarının belirli cisimleri tutabilirliğinin ölçümü sadece federe derin pekiştirmeli öğrenme yönteminin sonuçlarının tutarlı olup olmadığı konusunda test merci olarak bulunmaktadır. Buradaki amaç elde tutma verileri olmadığı durumlarda yapılacak olan çalışmalara öncü olmaktır. Federe derin öğrenme ile yapılan çalışmada veriler Kaggle.com (https://www.kaggle.com/datasets/ugocupcic/grasping-dataset) adlı siteden alınmıştır. Eldeki veride 3 kol eklem bölgeli ve 3 parmaklı tutma ucu olan ve bu tutma ucundaki parmakların da 3 eklem bölgesi bulunmaktadır. Algoritma Python dilinde yazılmıştır ve Intel(R) Xeon (R) Gold 5218R CPU ile Windows Server üzerinden gerçekleştirilmiştir.



Şekil 38 Örnek Bir Robot Kol

Çalışmanın kaynak kodları çalışmanın geçerli olması durumunda açık kaynak olarak Github.com adlı sitede çalışma ismi ile yayınlanacaktır.

Bilindiği üzere federe öğrenme birden fazla sistemin aynı anda öğrenmesini amaçlamaktadır. Bu sebeple çalışma 3 robot kolun birlikte eğitilmesi üzerine odaklanmıştır. Robotların tutabilirliğini federe derin öğrenme ile öğrenmesi algoritması yukarıda daha önce açıklanan sistemde en iyi öğrenme yönteminin ve en iyi ağın bulunması konusunda çalıştırılmıştır. Yapılan değerlendirmeler sonucunda en iyi ağın 18 girdi nöronlu (18 hareket noktası olduğundan dolayı), ve 3 tane 18x18 nöronlu gizli katman ve 1 tane 18 nöronlu gizli katman ve çıktı katmanı olarak da 1 nöronlu bir katman olduğu görülmüştür. Çıktı katmanının aktivasyon fonksiyonu 'sigmoid', kayıp fonksiyonu 'binary crossentropy' (tek bir çıktı olduğundan), optimizer olarak 'adam' fonksiyonu seçilmiştir. Oluşturulan model toplam 10 döngü yapacaktır. Aynı şekilde robotların kendi içlerinde yapacağı öğrenme döngüsü de 10 olarak belirlenmiştir.

Federe derin pekiştirmeli öğrenme yönteminde ise modelden bağımsız bir Q-öğrenme algoritması kullanılmıştır ve federe derin öğrenme algoritması yaklaşık %90 öğrenme gerçekleştirmiştir. Bundan dolayı federe derin pekiştirmeli öğrenme yöntemi de yaklaşık %90 öğrenme düzeyine erişene kadar kaç döngü yapacağı hesaplanacak ve gereken döngü öğrenildikten sonra iki algoritma aynı anda çalıştırılacak ve sonuçlar test edilecektir.

Test konusu olarak kullanılacak federe derin öğrenme algoritmasının sözde kodu şekil 39 ve şekil 40'da gösterilmiştir.

ALGORİTMA: Federe Derin Öğrenme Algoritması [Sunucu]

1 : *Başla* w_0
2 : *for her t = 0,1,.. do*
3 : $M \leftarrow max\ (\ [C\ .\ K],\ 1)$
4 : S_t = *rastgele m katılımcı ayarla*
5 : *for katılımcı için* $k \in S_t$ *in parallel do*
6 : w_{t+1}^k = *Yerel Güncelle (k,* w_t*)*
7 : $w_{t+1} = \sum_{k \in St} \frac{nk}{n\sigma}\, w_{t+1}^k, \quad n_\sigma = \sum_{k \in St} nk$

Şekil 39 Kullanılacak Federe Derin Öğrenme Algoritması Sözde Kodu (Sunucu)

ALGORİTMA: Federe Derin Öğrenme Algoritması [Yerel]

B = *yerel minibatch boyutu*
m = *yerel katılımcı sayısı*
E = *eğitim turu*
n = *öğrenme oranı*
Başla
$w_0 \leftarrow$ *rastgele başlatma*
{iletişim boyunca}
for $t = 1,..,T,\ldots$ *do*
    $S_t \leftarrow$ *(rastgele alt küme – max (C x K,1) katılımcı)*
    {her katılımcı için yerel optimizasyon}
    for katılımcı $k \in S_t$ *do*
        yerel ağırlıkları başlat: $w_{t,k} \leftarrow w_{t-1}$
        for epoch $e \in [1,E]$ *do*
            Yerel verileri böl, $B(\frac{B}{nk}$ *B yığınları)*
            for yığın $b \in B$ *do*
                $w_{t,k} \leftarrow w_{t-k} - n_{local}\, \Delta l\,(w_{t-k}\,;\, b)$
            end for
        end for
    end for
    {Merkezi Ortalama}
    $w_t \sum_{k \in St} \frac{nk}{n}\, w_{t,k}$
end for

Şekil 40 Kullanılacak Federe Derin Öğrenme Algoritmasının Sözde Kodu (Yerel)

Algoritmada asıl odaklanılan ve sunulan federe derin pekiştirmeli öğrenme algoritmasının sözde kodu aşağıdadır (Şekil 41, Şekil 42).

ALGORİTMA: Federe Derin Pekiştirmeli Öğrenme (İşçi)

for i = 1,2,3,..., M do
  for her bölümün t admında do
    Her s_t durumu için π_θ aktör modelini çalıştır ve do aksiyon a_t
    Ortamdan r_{t+1}, s_{t+1} alın
    <s_t, a_t, r_{t+1}, s_{t+1}> yörünge belleğine kaydedin
  End
  if öğrenme süreci biterse then
    Θ_w aktör modelin parametresini dahil ederek m_w mesajını sunucuya gönderin
    Break
  Else
    Güncelle $\pi_{\theta_{old}} \leftarrow \pi_\theta$
    for j = 1,2,3,…, K do
      Yörünge belleğinden bir mini-batch B alın (B'nin boyutu U'dur)
      for t = 1,2,… U do
        V_μ ve denklem kritik modelini kullanarak $\hat{A}_t$ 'yi hesaplayın
        $V_\mu(st)$ ve V_μ^{target} değerlerini alın
        $L^V{}_t(\mu)$ değerini hesaplayın
        π_θ Aktör modeli kullanarak $L_t^{CLIP}(\theta)$ değerini hesaplayın
      End
      Gradyanı $g_\mu = \nabla L^V(\mu)$ ortalama($L_1{}^V(\mu)$,...,$L_U{}^V(\mu)$) 'a göre hesapla
      V_μ parametresini SGD aracılığıyla g_μ ile güncelleyin
      $g = \nabla L^{CLIP}(\theta)$ gradyanı, ortalama($L^{CLIP}(\theta)$, ..., $L^{CLIP}(\theta)$) ve aktör model parametresi π a göre güncelleyin
      π_θ parametresini g_θ 'e göre SGD ile güncelleyin
  End
  Sunucuya g_θ içeren m_W mesajını gönderin
End
Müsait değilse sunucudan m_C mesajını bekleyin
if m_C aktör model parametresine $\bar{\theta}$ sahipse then
  Mevcut aktör modeli parametresini alınan $\bar{\theta}$ ile değiştirin
else if m_C ortalama $\bar{g}$ gradyanına sahipse
  SGD aracılığıyla alınan $\bar{g}$ ile π_θ güncelleyin
End

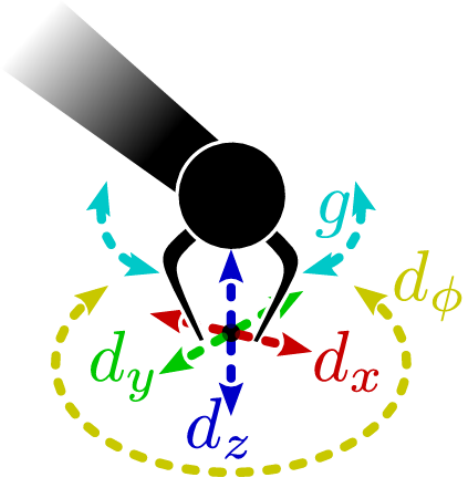
Şekil 41 Kullanılacak Federe Derin Pekiştirmeli Öğrenme Algoritması (İşçi)

ALGORİTMA: Federe Derin Pekiştirmeli Öğrenme (Sunucu)

```
for i = 1,2,3,..., M do
    P = []
    for w ∈ W do
        w işçisinden m_w mesajını al
        m_w 'yi P'ye böl
    End
    if bir mesaj varsa  m_w ∈ P ve m_w w işçisi aktör model parametresi θ_w then
        θ¯ = θ_w
        W − {w} içerisindeki işçilere θ¯ model parametresini de ekleyerek m_C mesajını gönder
        W = W − {w}
    Else
        Tüm m_W ∈ P  g_θ^1, g_θ^2, ..., g_θ^|W| gradyanlarını topla
        g¯ = ortalama(g_θ^1, g_θ^2, ..., g_θ^|W|)
        Ortalama gradyan g¯ dahil olmak üzere W 'daki tüm işçilere m_C mesajını gönder
    End
    if W boş ise then
        Break
    End
End
```

Şekil 42 Kullanılacak Federe Derin Pekiştirmeli Öğrenme Algoritması (Sunucu)

Çalışmada, uçtan uca robotik kavrama için eylem alanı, kartezyen uzayda sürekli eylemlerden oluşmaktadır. Eklem alanı yerine kartezyen uzaydaki eylemleri kullanarak, eylem alanı bir robotun belirli kinematik konfigürasyonuna göre değişmez. Ayrıca, kartezyen eylemler, kendi kendine çarpışmalardan kaçınırken düşük seviyeli eylem kontrolörlerine güvenilir bir şekilde komutlar da sağlamak için geleneksel yöntemler ve hareket planlama yaklaşımlarının kullanılabildiği yerlerde daha iyi güvenilirlik sağlamaktadır. Kolların tutucularının kullanabileceği kartezyen uzay şekil 43'te gösterilmiştir. Tutucu pozisyonu için eylemler, her ikisi de robot taban koordinat çerçevesine göre ifade edilen öteleme yer değiştirmesi için (*dx*,*dy*,*dx*) ve z ekseni *dφ* etrafında nispi rotasyonlardan oluşmaktadır.



Şekil 43 Robot Kollarının Tutucularının Kullanabileceği Katezyen Uzay

Şekil 43'te (*dx,dy,dx*)'nin bir öteleme yer değiştirmesini *dφ'*nin bir nispi sapma dönüşünü ve kıskaç kapanması ve açılmasının *g* ile gösterilmektedir. Bu eylemler [-1,1] aralığında normalleştirilir ve ardından uygulanmadan önce metrik ve açısal birimlere yeniden ölçeklendirilir. Kıskaç eylemi *g* ayrıca sürekli bir aralıktadır. Burada pozitif değerler tutucuyu açar ve negatif değerler tutucunun kapanmasını sağlar. Bu nedenle, RL aracısının bir tanımlama grubu (*dx,dy,dx, dφ, g*) için karşılık gelen değerleri sürekli eylemlerin herhangi bir kombinasyonunu yapmasına izin verir.

A. V-REP (Coppelia Robotics Virtual Robotic Experiment Platform)

Bu projede V-REP kullanılmıştır. V-REP Coppelia Robotics firmasının sanal ortamda robotik çalışmalar yapılması amacıyla geliştirilen bir simülasyon ortamıdır. V-REP'in eğitim amaçlı kullanımı ücretsizdir. Python dahil olmak üzere birden çok programlama dilini destekler ve bir uzak API aracılığıyla erişilebilir. Simülatör, Mico kolu da dahil olmak üzere birden fazla robotik platform modeline sahiptir. Simülasyon modeli çok doğru ve gerçek kolu çok iyi temsil ederken, parmakların kontrolü farklıdır. Az çalıştırılan parmakların simülasyonda uygulanması daha zor olduğundan, bir parmağın her biri bir segmenti kontrol eden iki aktüatörüdür. Gerçek kol bir lineer aktüatör kullanırken, simülasyondaki iki aktüatör döner aktüatördür. V-REP sadece kavramak için değil, daha çok genel simülasyonlar için üretildiğinden, nesneleri fiziğe dayalı olarak kavramaya çalışmak oldukça zordur. Çünkü V-REP

sürtünmeyi o kadar doğru bir şekilde modellenemez. Ancak simülasyonda çarpışmayı oldukça kolay bir şekilde tespit edebiliriz, bu nedenle parmaklar bir nesneyi kavramaya çalışırken çarpıştığında fiziğe güvenmek yerine nesneyi bir kavrama simüle ederek uç efektöre bağlanabilir. V-REP farklı modlarda çalışabilir; asenkron ve senkron. Asenkron modun yaygın olarak kullanılır, simülasyonu duraklatmadan olabildiğince hızlı çalıştırır. Ancak senkron mod için simülasyonu bir adım ilerletmek için bir tetikleme sinyaline ihtiyaç vardır. Kolun mevcut durumuna ihtiyaç olduğundan dolayı senkron modu kullanmak gerekmektedir. Her adımda bir işlem gerçekleştirilir ve delta süresini (Δt) aynı tutarken kolun yeni durumu alınır. Bu aynı zamanda ödül değerini hesaplamaya, veriyi tren arabelleğinde göndermeye ve yeni eylemi, kol önceki eylemini gerçekleştirmeden farklı bir duruma getirerek hesaplamaya izin verir.

Senkron modunu kullanmanın en büyük dezavantajı, simülatöre bir kamera eklendiğinde, güncelleme hızının normalde yaklaşık 40Hz-60Hz'de çalışmaya kıyasla yaklaşık 4Hz-6Hz'e düşmesidir. Bu, eğitim için görsel girdiyi kullanmayı imkânsız hale getirmektedir, çünkü eğitimin uzun sürmesi gerekir. Bu proje için görsel girdi olmadığından dolayı üzerinde doku olmayan bir küre cisim kavranmaya çalışılacaktır. Küre cismin yarıçapı 3cm olarak belirlenmiştir. V-REP, noktalar arasındaki minimum mesafeyi hesaplayabilir ve görselleştirebilir, bizim durumumuzda, ödül işlevi için kullanılacak olan uç efektör ile kürenin ortası arasındaki minimum mesafeyi almaktadır. V-REP çarpışma testini yapar, ancak yapabileceği tüm çarpışmaları kaydetmek istenmez, çoğunlukla son efektörle ilgilenilmektedir. Bu nedenle, yalnızca uç efektörün kolla değil, zeminle veya nesneyle çarpışıp çarpışmadığı kontrol edilir. Simülasyonda parmakların nesneyi kavrayıp kavramadığı kontrol edilmek için her bir parçanın nesneyle çarpışıp çarpışmadığı kontrol edilmektedir.

B. Eğitim Öncesi

Eğitim veya daha iyisi eğitim örnekleri oluşturmak, V-REP gibi bir fizik simülatörü kullanılarak nispeten yavaş olabilir. Ancak simülatörü sadece ileri kinematik kullanarak basitleştirebiliriz. Her bir eklemin mevcut konumunu takip ederek, ileri kinematiği hesaplayarak son efektör konumunu hesaplayabiliriz. Actor ağındaki eylemi ve keşif süresinin 50ms'lik bir delta süresiyle ekleyerek, kolun hareketini

simüle edebiliriz. Bir denemenin çalışması yaklaşık 12,5 saniye sürdüğünden dolayı birçok deneme yapılabilecektir. Ancak bu yöntemin dezavantajı, nesneleri kavrayamamak ve herhangi bir çarpışma kontrolü olmamasıdır. Bu nedenle, fizik simülatörü kullanırken eğitim süresini azaltıp azaltamayacağımızı görmek için bu yöntemi yalnızca eğitim öncesi bir aşama olarak kullanılacaktır.

C. Mimari

Kolu kontrol etmek için bir sinir ağının kullanmanın yanı sıra, kolu kontrol etmek için mimariyi optimize etmenin mümkün olup olmadığı araştırılmak istenmektedir. Bir mimari, bir dizi eklemi kontrol eden ağların miktarıdır. Bir sinir ağının, hepsi aynı ödül işlevine bağlı olarak 6 eklemi ve parmağı kontrol edeceği yerde, her birinin mevcut aktüatörlerin bir alt kümesini kontrol ettiği ve kendi ödül işlevine sahip olduğu iki sinir ağına sahip olmak mümkün olabilir. Örneğin, alt eklemler ve uç efektörün nerede biteceği üzerinde büyük bir etkiye sahipken, yüksek eklemler, uç efektörün son konumunun ince ayarını yapmak için kullanılır. Bu nedenle, bu ağlara farklı ödül işlevleri vermek, eğitim performansını iyileştirebilir. Birden fazla ağ kullanırken ince ayarlanabilen farklı hiper parametrelerin miktarını azaltmak için mimarideki her ağ için gizli nöronların miktarını aynı tutuyoruz. Ancak ağlar için farklı ödül işlevleri denenmektedir.

D. Ödül İşlevi

Bir nesneyi başarılı bir şekilde kavradıktan ve kaldırdıktan sonra, temsilciye yalnızca çok seyrek bir ödül verilmesi arzu edilirse de, bu tür bir yaklaşım, rastgele keşif yoluyla bir başarıya ulaşmanın seyrekliği nedeniyle eğitimi uzatacaktır. Bu nedenle, bu çalışma, üç farklı aşamadan oluşmaktadır. Bunlar; ulaşma, dokunma ve kavrama gibi seyrek ödülleri bir araya getiren bileşik bir ödül işlevinden yararlanır. Bu aşamalar, ajanın önce bir nesneye yaklaşması, ardından dokunması, kavraması gereken hiyerarşik bir akışı takip eder. Her bölüm boyunca, temsilcinin, son aşamada istenen bir hedefe yol açmayacak herhangi bir ödüllendirici davranıştan vazgeçirmek için bu aşamaların her birinden yalnızca bir kez ödül almasına izin verilir, örneğin sürekli olarak biriktirmek için bir nesneyi tekrar tekrar itmek gibi. Dokunmanın ödülü, ödül işlevindeki her bileşenin oranı ve ölçeği, aracının optimize etmeyi amaçladığı ilkeyi doğrudan etkilediği için ayarlanabilir bir ortam hiper parametresi

olarak ele alınabilir. Genel olarak, son aşamadı ödül, ilk aşamada verilen ödülden çok daha yüksek olmalıdır; bu, yalnızca temsilcinin eğitimine rehberlik etmesi anlamına gelir bu nedenle, her *i* aşaması için bireysel ödülü belirlemek için üstel bir işlev r^{i-1}_{exp} kullanılır. Temel $r_{exp} \in [1, \infty)$ ayarlanabilir, burada $r_{exp} = 7$, uygulanan kavrama görevi için ampirik olarak tatmin edici sonuçlar vermektedir. Ve teorik olarak elde edilebilir maksimum ödül $r_{max} = 400$ ile bulunur. Görevi başarmak için pozitif ödüle ek olarak, aracıya ayrıca istenmeyen çarpışmaların sayısından caydırmak için robotun yer düzlemi ile çarpışmada olduğu her zaman adımı için -1 negatif ödül verilir. Ayrıca, aracıyı görevi mümkün olduğunca hızlı gerçekleştirmeye teşvik etmek için sonlandırılana kadar her adımında -0.005'lik küçük bir ödül çıkarılmaktadır. Tüm ödüller çizelge 3'te özetlenmiştir.

Çizelge 3 rexp = 7 ayarlandığı ve her bölümün en fazla 100 zaman adımına sahip olduğu kavrama görevi için bu çalışmada kullanılan ödül fonksiyonuna genel bakış.

Hareketler	Ödüller	
Ulaşmak	$r^0_{exp} = 1$	
Dokunmak	$r^1_{exp} = 7$	*(bölüm başına bir kez)*
Kavramak	$r^2_{exp} = 49$	
Çarpışma	*-1*	*(her adımda)*
Çabuk Hareket Etme	*-0.005*	

E. Keşif

Keşif pekiştirmeli öğrenmenin bir parçasıdır, onsuz ajan herhangi bir öğrenme gerçekleştiremez. Bu çalışmada iki Gauss keşif yöntemi kullanılmaktadır. Bu yöntemin yanı sıra, bir aktüatörün keşfedeceği bir olasılık belirleyerek keşif miktarı da sınırlanmaktadır. 7 aktüatör, 6 eklem ve bir kavrayıcı bulunduğundan dolayı 3 boyutlu uzayda hareket ederek bir hedefe ulaşmak ve kavrama görevi yapmak, hedefe ulaşmak için tek başına keşif yapmak çok zor olacaktır. Bir aktüatörün bazen öğrenilmiş hareketini gerçekleştirmesine izin vererek, kol hedefine daha yakın hareket edecek ve hedefin etrafını keşfedebilmesi sağlanabilir.
Çalışmada kullanılacak olan robot kolunun D-H (Denavit-Hartenberg) değişkenleri çizelgede gösterilmiştir.

Çizelge 4 Robotik Kol İçin Verilen D-H Parametreleri

i	*a(i - 1)*	a*(i - 1)*	d_i	*theta*
1	0	0	0.2755	*q1*

2	*-π / 2*	0	0	*q2*
3	0	0.2900	0.0070	*q3*
4	*-π / 2*	0	0.1661	*q4*
5	1.0472	0	0.0856	*q5*
6	1.0472	0	0.2028	*q6*

Çizelge 4'teki eklem sayısı i olarak temsil edilir. *a* x ekseni etrafında ölçülen *z* eksenleri arasındaki açıdır, a x ekseni etrafında ölçülen z eksenleri arasındaki mesafedir, d_i z ekseni etrafında ölçülen x eksenleri arasındaki mesafedir, theta ise eklemin açısıdır.

Denavit-Hartenberg yöntemini biraz açıklamak gerekirse; robotların kinematik modelini çıkarırken en çok kullanılan yöntemlerden bir tanesidir. D-H yöntemi bir nevi homojen dönüşüm sağlamaktadır. Bu yöntemde dört ana değişken kullanılır ve robot kinematiği çıkarılır. Yukarıda açıklanan değişkenler hesaplanırken öncelikle robot kolunun eklem bölgeleri belirlenir ve dönme eksenleri bağlardan bir fazla olacak şekilde numaralandırılır. Bu aşamadan sonra bu eksenlerin her birine koordinat çerçevesi yerleştirilir ve bağ dönme ekseni koordinat çerçevesinin Z ekseni kabul edilir.

F. Ajan

Bir ajan, simülasyon ve sinir ağları ile olan bağlantıyı yönetmektedir. Aracı bir çalıştırma ile başlamadan önce en son aktör ağını alır ve simülasyon dünyasını sıfırlar. Daha sonra maksimum 250 adım gerçekleştirilir, burada bir adım simülasyonda Δt'lik 50ms ile gerçekleştirilen tek bir eylemdir ve maksimum çalışma süresi her adımda 12,5 saniyedir. Aracı her adım için durumu alır, durumu aktör ağı için girdi olarak kullanılır. Eylem, bir eklemin gerçekleştirilmesi gereken radyan cinsinden hızıdır. Eğitim sırasında aracı, simülasyonda gerçekleştirmeden önce eyleme keşif de ekleyecektir. Simülasyon bir zaman adımı atıldıktan sonra yeni durum toplanır ve ödül belirlenir. Başlangıç durumu, ödül gerçekleştirilen eylem ve yeni durum tekrar arabelleğe kaydedilir. Durum, kullanılan tüm eklemlerin radyan cinsinden konumu, uç efektör bağlantısının metre cinsinden x,y,z konumu ve kullanıldığında, radyan cinsinden parmakların açısı ve son olarak metre cinsinden hedef konumu (x,y,z). Eklemlerin açı girdileri, açının 2π (bir eklemin tam dönüşü) ile bölünmesiyle 0 ile 1 arasında bir değere normalize edilir. Parmak eklemleri, açabilecekleri maksimum aralık olduğu için 1.0472 radyana (60 derece) bölünerek normalleştirilir. Pozisyon değerleri için, kolun erişimi 1 metrenin altında olduğu için

girişleri normalleştirmemiz gerekmez. Bu bilgiler simülasyondan elde edilebilir. Aracının yanı sıra, arabelleğinden mini bir toplu iş almaktadır. Eğitim dizisi yaklaşık 100Hz'de çalışmaktadır.

G. Konum

İlk önce kolun uç efektörü belirli bir konuma hareket ettirmeyi öğrenmesine izin vererek başlanacaktır, bu algoritmanın doğru çalıştığını gösterecek ve ayrıca parametre ayarına daha iyi fikir verecektir. Deneyler sadece bir eklem kullanılarak başlayacak, bu uç efektörün yalnızca bir düzlemde hareket edeceği, daha fazla eklem eklendiğinde uç efektörün 3B uzayda hareket edeceği ve hedefini bulmasını zorlaştıracağı anlamına gelmektedir. İlk eklem kullanılarak başlanmıştır ve karmaşıklığı artırmak için daha fazla eklem bölgesi eklenecektir. İlk aşamalarda farklı öğrenme oranlarını ve gizli katman boyutları test edilecektir. Test edilecek parametre miktarını azaltmak için en iyi öğrenme oranını ve gizli katman boyutlarını belirlemek için erken sonuçlar kullanılacaktır. Algoritmanın ne kadar sağlam olduğu da test edilecek ve birden fazla hedefi öğrenmenin üstesinden gelip gelemeyeceği görülecektir. Ajan, uç efektörü hedefin 4cm yakınına yerleştirilebildiğinde, en az bir zaman boyunca orada tuttuğunda ve eylem vektörünün uzunluğu 0.001'den küçük olduğunda başarılı olmuştur. Bu hedefe ulaştığında kolun biraz hareket etmesine izin verir, ancak kolun tam konumunda durmayı öğrenmesi çok daha fazla eğitim süresi gerektirmektedir. Bu şekilde, kaleye yakın olduğunda kol hala yavaşlamak zorunda kalır, ancak tam bir durma noktasına gelmek için kalenin çevresinde çok fazla zaman harcamak zorunda kalmaz. Ödül fonksiyonu için, negatif uzaklık çarpı a sabit, c1 ve eylemlerin negatif nokta çarpımı çarpı a sabit, c2 alınır. İlk deneyler, c1 için 1 ve c2 için 0.2 değerinin iyi sonuçlar verdiğini, yani mesafe faktörünün eylem çıktısından daha önemli olduğunu gösterdi. Bu ödül işleviyle, hedef konumuna hareket edilerek en yüksek ödül elde edilebilir ve her eylem için sıfır çıktı değerine sahip olunduğunda, hedef konumundan uzaklaşmak daha düşük bir ödül verir. Kol zemine çarptığında, koşu başarısız olur ve sona erer, son eylemini yaparken hesaplanan mevcut ödüle ekstra bir negatif ödül -1 eklenir. Kol kendine çarpıyorsa koşu devam eder. Kendi kendine çarpışmada başarısız olmamasının nedeni, simülasyonu yavaşlatan ekstra hesaplamalar gerektirmesi ve aynı zamanda erken deneyler

sırasında hiçbir zaman bir soruna neden olmadığının gösterilmesidir, çünkü nadiren gerçekleşir ve olmadığı için yalnızca daha büyük negatif ödül alır.

H. Kavrama

Kavrama için sadece tek bir nesne, bir küre cisim kavranmaya çalışılacaktır. Kavrama ile, parmakların nesneyi kavradığı kastedilmektedir, ancak nesneyi kaldırması gerekmemektedir. Simülasyonun fiziğini kullanarak bir nesneyi kavramak ve kaldırmak doğru bir şekilde yapılması çok zor olabilir ve simülatörde çok sayıda ince ayar yapılmasını gerektirir. Ayrıca son efektörden yani tutucudan çok net bir biçimde geri bildirim almayı gerektirir. Bu çalışmada tüm eklemler kullanılır ve deneylerde her zaman parmaklar da eklenir. Parmaklar tek bir hamle ile kontrol edilmez, her biri için ayrı ayrı öğrenme gerçekleştirilir. Bu aynı zamanda daha iyi bir kavrama hareketi yaratacaktır. Nesnenin nasıl kavranacağını belirlemek için oldukça basit bir yöntem kullanılmaktadır. Nesnenin orta nokta konumuna yaklaşması için ona çoklu açılar verilecektir. Planlayıcı da bu koleksiyondan en iyisini belirleyecektir. Aracı, tüm parmak segmentleri nesneye dokunduğunda başarılı olmuş sayılacaktır. Bu, parmakların nesneyi kavradığı anlamına gelmektedir, ancak kavramanın çok doğru olmadığı durumlar da olabilir. Son efektör, nesneyi, parmakların nesneyi tam olarak değil, yalnızca küçük bir yüzeyle kavramasına neden olacak bir açıyla kavranabilir. Gerçek hayatta nesne daha sonra hareket eder ve büyük olasılıkla kavrayan parmaklara daha iyi oturur, ancak simülasyonda bu, nesnenin garip çarpışma davranışı oluşturmasına neden olabilir. Bu nedenle cismin ağır olması ve kolla hareket ettirilemeyecek olması seçilmiştir. Bu şekilde cismin garip hareket etmesi konusunda endişelenmemize gerek kalmaz, ancak kavramanın her zaman optimal bir kavrama olmadığı da göz önünde bulundurulur. Ödül işlevi için, nesneden 5 cm'den fazla uzaktayken kavrama olmadığın göz önünde bulundurulur. Ödül işlevi için, nesneden 5 cm'den fazla uzaktayken parmakların negatif mesafe çarpı sabit, c2, nesnenin 5 cm yakınında parmakların pozitif açısı çarpı c2 olduğunda, negatif mesafe çarpı sabit, c1 alınır. Ve eylemlerin negatif nokta ürünü çarpı a sabit, c3. İlk deneyler, c1, c2 için 0,5 ve c3 için 0,2 için 1 değerinin iyi sonuçlar verdiğini göstermektedir. Uç efektör yere değdiğinde çalışma durur, ödüle -

1 eklenir, ancak kendi kendine çarpışma olmaz. Mesafe ölçümü, uç efektör noktası ile nesnenin merkezi arasındaki mesafedir.

I. Tensorflow

Bu çalışmada sinir ağlarını programlamak için Tensorflow [Kaynak] kullanılmıştır. Tensorflow, simülasyonu sinir ağlarına bağlamayı epey kolaylaştıran Python'da programlanabilir. Tensorflow ayrıca, sinir ağlarını eğitmek için bir GPU kullanılabilir, ağ boyutları GPU'ların sağladığı paralellikten tam olarak yararlanacak kadar büyük olmadığından dolayı CPU üzerinde çalıştırılmıştır. Tensorflow, karmaşık ağlar oluşturmayı kolaylaştırır, ancak bu çalışmada tamamen bağlı sinir ağları kullanılmaktadır. Tam bağlantılı bir sinir ağı, matrix çarpımları ile kolayca oluşturulabilir. Tensorflow ayrıca birçok farklı etkinleştirme işlevine sahiptir. Tensorflow, ağırlıkları güncellemek için gereken türevleri otomatik olarak hesaplayacaktır. Bu, türevi hesaplama konusunda endişelenmeye gerek kalmadan derin ağlar oluşturulmasına olanak tanır. Hata fonksiyonu ve eğitim algoritması da belirtilebilir. Ayrıca eğitim için toplu işlerden yararlanmak kolaydır ve girdisine göre bir ağın çıktısı da kolaylıkla alınabilir. Farklı parametreleri kullanmak için, belirtilen giriş ve çıkış sayısına, gizli nöron ve katman sayısına ve öğrenme oranlarına dayalı olarak doğru bir Tensorflow modeli oluşturmaya izin veren bir yapılandırma dosyası oluşturulmuş ve çalışmalar bunun üzerinden yürütülmüştür.

İ. Hiper parametre Optimizasyonu

Hiper parametrelerin seçimi, öğrenilen bir ilkenin nihai performansının yanı sıra öğrenme eğrisi de önemli ölçüde etkilenebilir. DRL'nin hiper parametrelere karşı bu kırılganlığı, bu nedenle optimizasyonların büyük önem taşıdığı ve her ortam için gerçekleştirilmesi gerektiği anlamına gelir. Bu çalışmada, oluşturulan ortam, gözlemler ve kullanılan RL algoritmaları için politikanın sağlam bir şekilde öğrenilmesini sağlayacak bir dizi hiper parametre elde etmek amacıyla hem otomatik optimizasyon hem de manuel ince ayar gerçekleştirilir. İlk olarak, Optuna adında bir hiper parametre optimizasyon çerçevesi kullanılarak otomatik bir hiper parametre

optimizasyonu uygulanır. Optuna ve diğer benzer çerçeveler, hiper parametre uzayını yinelemeli olarak aramak ve bazı metriklere göre en iyi sonuçları sağlayan bir kombinasyon bulmak için kullanılan bir dizi farklı deneme gerçekleştirerek DL için uygun bir hiper parametre kombinasyonu seçme sorununu ele alır. RL açısından bu metrik, bir aracının belirli bir değerlendirme dönemi boyunca biriktirebildiği bir ödüldür. Optuna, örnekleyici ve budayıcı olmak üzere iki bölümden oluşur. Örnekleyici, bir sonraki deneme için hiper parametre arama alanından bir dizi hiper parametre seçer. Bu tür bir seçim tamamen rastgele olabilir. Örneğin bir deneyin başında veya önceki tüm denemelerden istatistiksel analiz yapan algoritmalar uygulayarak. Bu bağlamda Pruner, boşa harcanan kaynakların miktarını sınırlamak amacıyla umut verici olmayan denemelerin erken durdurulmasına izin veren bir stratejidir. Budama, her denemenin değerlendirme bölümlerinin düzenli aralıklarla çalıştırılmasını gerektirir; burada her yeni deneme, önceki tüm denemelerin performansıyla karşılaştırılır ve ödül karşılaştırılabilir şekilde çok düşükse budanır. Kavrama ortamı için, güvenilir bir performans sağlayan bir taban çizgisi elde etmek için ilk olarak hiper parametreleri optimize etmek için Optuna uygulanır. Bu optimizasyon, arama alanının, özellik çıkarıcının boyutu ve aktör-eleştirmen ağları dahil olmak üzere çoğu hiper parametreden oluşturduğu SAC kullanılarak gerçekleştirilmiştir. Yeniden oynatma arabelleğinin boyutu, toplu iş boyutu ve ilk entropi otomatik olarak optimize edilmedi. Yeniden yürütme arabelleği ve toplu iş boyutu, sırasıyla maksimum RAM ve VRAM kullanımı açısından kullanılan sistem için yeterince büyük olacak şekilde seçilmiştir. İlk etropi tutarlı tutulur çünkü büyük ilk entropinin istenmeyen budama ile sonuçlanabileceği her denemenin ilk aşamalarında performansı doğrudan etkiler. Maksimum deneme süresi 1500 zaman adımı olan toplam 70 deneme kullanılmıştır. Budamayı tetikleyebilecek her 125 zaman adımında bir 20 değerlendirme bölümü seti gerçekleştirilmiştir. Sonunda, sonraki manuel ayarlama için en iyi performans gösteren hiper parametre seti kullanılmıştır. Optuna ile otomatik optimizasyon, bu çalışmada oluşturulan gibi karmaşık ortamlar için çok fazla hesaplama süresi gerektirdiğinden manuel ayar uygulanır. Bu aynı zamanda, maksimum deneme süresi 1500 zaman adımı olan 70 denemenin kullanılmasının da nedenidir. Ve bu halihazırda 1 haftalık bir deneme süresi almıştır. Bu nedenle, hedeflenen hiper parametrelerin manuel olarak ayarlanması, manuel olarak başlatılan ve durdurulan birkaç denemeyle daha yapıldı.

Bu sürecin odak noktası çoğunlukla uygulanan ortamın hiper parametreleridir. Tüm aktör-eleştirmen algoritmaları için ortaya çıkan parametreler çizelge 5'te görülebilir.

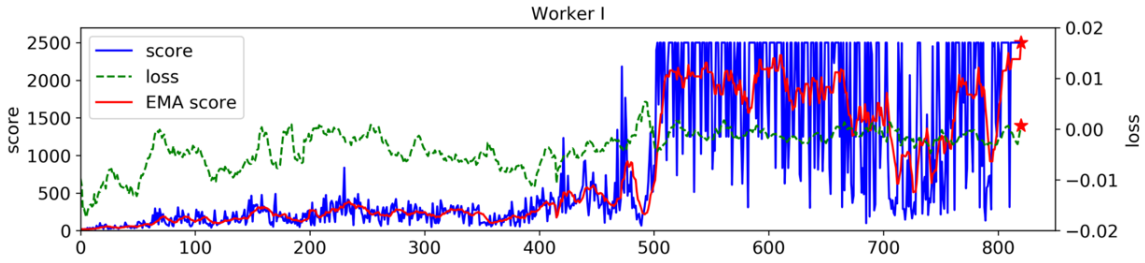
Çizelge 5 Oluşturulan Sinir Ağı İçin Hiper Parametreleri

Değişken	Değer
Hiperparametre	DFQN
Optimizasyon Algoritması	Adam
Öğrenme Oranı	Lineer, $1.5 \cdot 10^{-4} \rightarrow 0$
Mini-Batch Boyutu	32
Güncelleme Frekansı	Her bölümden sonra
Her Güncellemedeki Gradyan Adımı	100
Tekrar Arabellek Boyutu	40000
Discount Faktörü	0.999
Hedef Güncelleme Oranı	$5 \cdot 10^{-5}$
Eleştirmen Sayısı	1
Aktivasyon Fonksiyonu	Binary Crossentropy
Keşif Eylemi Gürültüsü	$N(0, 0.25)$
Hedef Politika Gürültüsü	$N(0, 0.25)$
İlk Entropi Katsayısı	0.1
Entropi Katsayısı	-dim(A) = -5
Atom Numarası	25
Kesilmiş Atom Sayısı	3

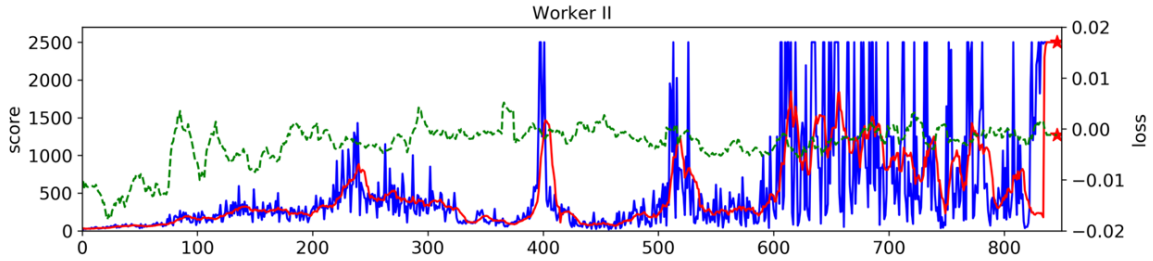
VI. SİMÜLASYON SONUÇLARI VE DEĞERLENDİRMESİ

A. Simülasyon Sonuçları ve Kavrama Deneyleri

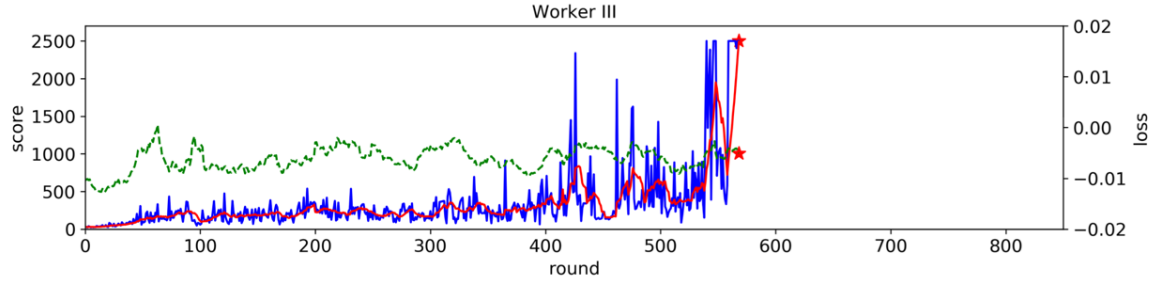
Çalışmada 3 farklı robot kolunun federe derin öğrenme ve federe derin pekiştirmeli öğrenme teknikleri ile cisimlerin tutulabilirliğinin öğrenilmesi üzerinde çalışılmıştır. Sekil 6.1,6.2 ve 6.3'te Robot kolların tek başlarına öğrenme sonuçları gösterilmiştir. Şekillerde sonuçlar 2500 score üzerinden değerlendirilmiş ve 2500 score federe derin öğrenme modelindeki %90'lik eğitim doğruluğuna karşılık gelmektedir. Şekillerdeki 1200 döngü federe derin öğrenme yaklaşımında 300 döngüye karşılık gelmektedir. Aşağıdaki şekillerden anlaşıldığı gibi robot kollar tek başlarına yaklaşık olarak 900 döngüde 2500 puana ulaşmışlardır. Federe derin öğrenme yöntemi ile federe derin pekiştirmeli öğrenme yöntemlerinin karşılaştırılması şekil 47 ve şekil 48'de gösterilmiştir. Şekillerden de anlaşılacağı üzere federe derin pekiştirmeli öğrenme yönteminde robot kollar herhangi bir yerel optimaya takılmadan 1000 döngüde %90 düzeylerinde öğrenme gerçekleştirmişlerdir. Ancak şekil 44'te görüleceği üzere federe derin öğrenme yönteminde robot kol 2'nin çalışma performansı ancak 1200 döngüde %90 düzeylerine erişmiştir. Federe derin öğrenme yöntemi yaklaşık olarak 6 saat sürmekte iken, federe derin pekiştirmeli öğrenme yöntemi 5 saat sürmektedir. Yani federe derin pekiştirme yöntemi ile öğrenme hedeflenen başarı düzeyine federe derin öğrenmeden daha hızlı ulaşmıştır.



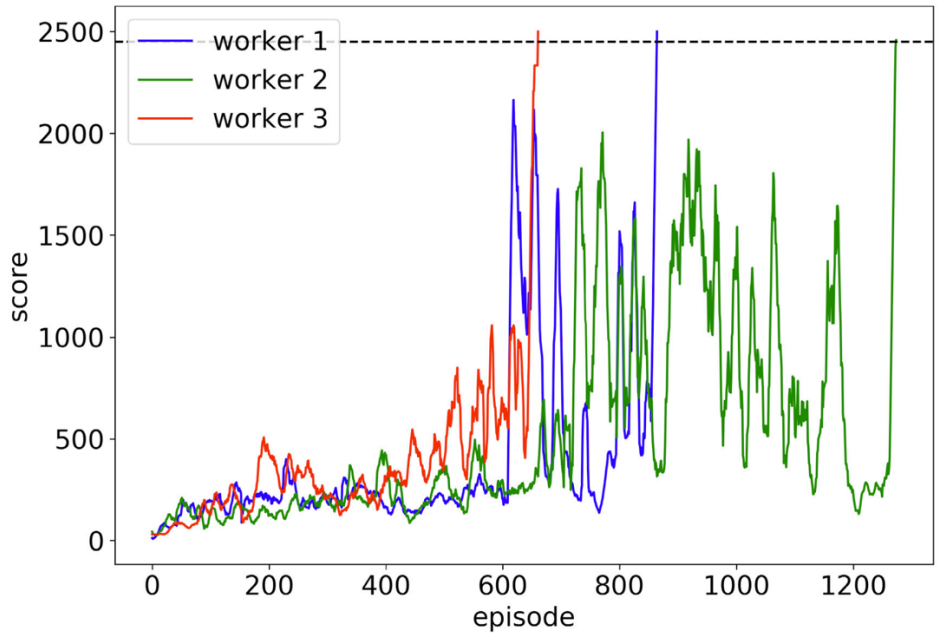
Şekil 44 Robot Kol 1'in derin pekiştirmeli öğrenme sonuçları



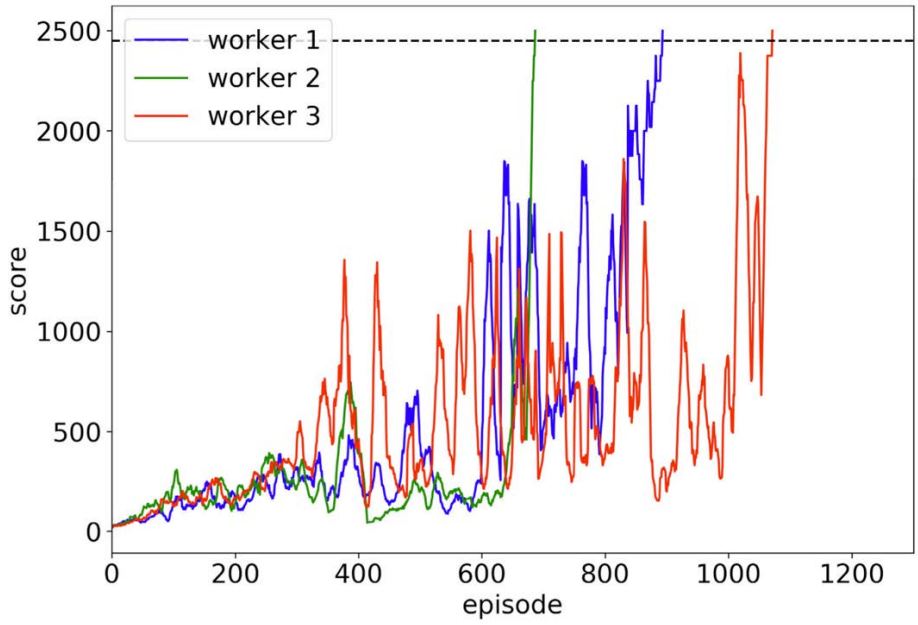
Şekil 45 Robot Kol 2'nin derin pekiştirmeli öğrenme sonuçları



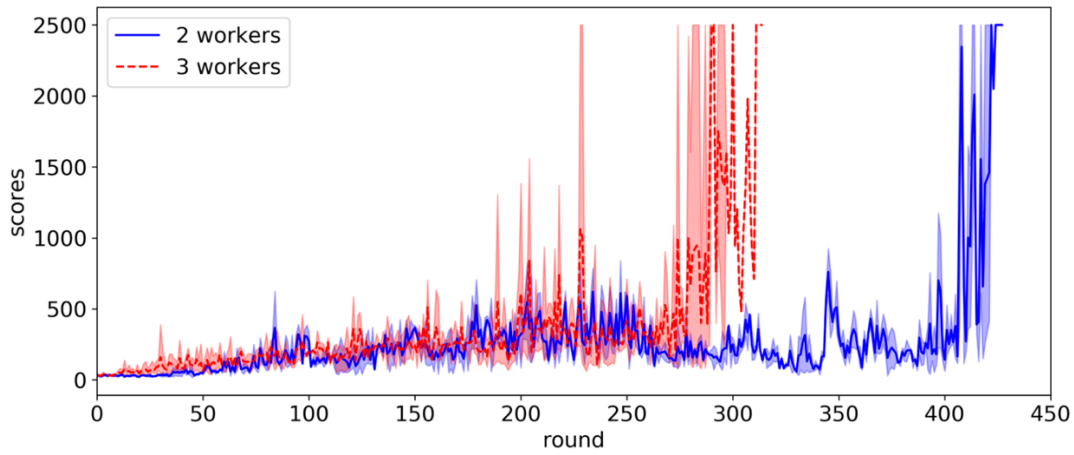
Şekil 46 Robot Kol 3'ün derin pekiştirmeli öğrenme sonuçları



Şekil 47 Federe derin öğrenme yöntemi ile öğrenme



Şekil 48 Federe derin pekiştirmeli öğrenme yöntemi ile öğrenme



Şekil 49 Federe derin pekiştirmeli öğrenme 2 ve 3 robot kolu ile öğrenme sonucunun karşılaştırması

Federe derin pekiştirmeli öğrenme algoritmasında sunucuya bağlı alt sistemlerin sayısı ne kadar arttırılırsa öğrenme o kadar hızlı ve düzenli gerçekleşir. Şekil 49'da robot sayısı 2'den 3'e çıkarıldığında öğrenme düzeyinin ne kadar hızlandığı görülmektedir.

Her 10 denemeden sonra bir test çalıştırması gerçekleştirilen, her deney sırasında, keşif eklemeden ve her adımda ağları eğitmeden bir test çalıştırması yapıldı. Actor ve Critic ağları, her 324 nörondan oluşan 2 gizli katmana sahiptir ve arabellek yöntemi kullanılarak eğitilmiştir. Kavramak için hep aynı nesneleri kullanılmıştır ve nesne

hep aynı pozisyondadır. Çizelge 6 fizik simülatörünce 700 ön eğitim denemesi ve 300 öğrenme denemesi kullanmanın sonuçlarını göstermektedir. Sonuçların da gösterdiği gibi, ağ nesneyi kavramayı yeterince öğrenmemiştir. Ortalama mesafe, nesneye yaklaştığını gösterse de, cismi gerçekten kavrayamamıştır. Test çalışması sırasında bazı denemelerde kavramaya çok yaklaşırken, nesneyi yatay yerine dikey olarak kavramaya çalışırken çoğunlukla yanlış bir yönelime sahiptir. Bu durum büyük olasılıkla eğitim öncesi aşamada aşırı eğitilmesinden kaynaklandığı düşünülmektedir. Bu aşamanın fiziği olmadığı ve dolayısıyla kavramaya bile kalkışamadığından dolayı, yolunda tutması gereken bir nesne olduğunu bilmeden sadece bir konuma gitmeyi öğrenmiştir. Kavrama, bir noktaya gitmekten çok daha zor olduğundan dolayı koşu sayısı 1500'e çıkarılmıştır ve ön eğitim aşaması kullanılmamıştır. Çizelge 7 sonuçları göstermektedir. %90 başarı oranı ile algoritmanın bir nesneyi kavramayı öğrenebildiği gösterilmiştir. Dezavantajı ise, ön eğitim yöntemini kullanmadan genel eğitim 5 saatten fazla sürmektedir. 3,94 cm'lik ortalama mesafe, ağların uç efektörünü son konumuna yaklaştırabildiğini, ancak bazen oryantasyonun doğru olmadığını gösteriyor. Bazen son efektör ve parmaklar nesnenin etrafındadır ancak başarılı olamamıştır. Bu büyük olasılıkla, parmakların tüm bölümlerinin nesneyle doğru şekilde temas etmediği anlamına gelmektedir.

Çizelge 6 Hedef pozisyonuna göre nihai pozisyonun standart sapması ile ortalama başarı oranı ve ortalama mesafe. 700 denemelik ön eğitim kullanılarak ortalama 10 deneme ve toplu öğrenme için bir arabellek ve iki aracı kullanılarak 300 deneme için fizik simülatörü kullanılarak eğitilmiştir. 6 eklem ve 3 parmak kullanılmıştır.

	Gaussian
Doğruluk Oranı	%60
Ortalama Mesafe	6.11cm
Ortalama Hata	4.12cm

Çizelge 7 Hedef Pozisyonuna göre nihai pozisyonun standart sapması ile ortalama başarı oranı ve ortalama mesafe. Ortalama 10 deneme, 1500 deneme için fizik simülatörü kullanılarak eğitilmiş, toplu öğrenme için bir arabellek ve iki aracı; 6 eklem ve 3 parmak kullanılmıştır.

	OUP
Doğruluk Oranı	%90
Ortalama Mesafe	3.94cm
Ortalama Hata	1.93cm

Bir dizi deneyde robot kolunun cisimlerin tutulabilirliğinin öğrenilmesi gösterilmiştir. 1 ile 3 eklem bölgesi kullanarak çevrimiçi öğrenme mümkün olsa da çalışmada kullanılacak 7 eklem bölgeli robot kol kullanıldığında performans düşmektedir. Çözümlerden birisi 1200'den fazla deneme yapılmasına izin vermek olsa da bu nispeten basit görevi öğrenmek de çok fazla zaman alacaktır. Bir eğitim dizisinin ağları mini gruplarda eğitebileceği bir arabellek kullanarak, aynı miktarda deneme kullanıldığında sonuçların çok daha iyi olduğu, ancak henüz mükemmel olmadığı gösterilmiştir. Deneme sayısı iki katına çıkarıldığında başarı oranı her bir keşif yöntemi için %80 ve %90 başarı oranları ile kabul edilebilir hale gelir. Ancak eğitim süresi de iki katına çıkmaktadır. Bir aracın eğitim örnekleri oluşturmak için yapabileceği yinelemelerin miktarını artırmak için FDRL algoritması sunulmuştur. Nispeten büyük miktarda ön eğitim denenesi ve fizik simülatörü kullanılarak daha az miktarda eğitim kullanıldığında, ağları eğitmek için gereken toplam süreyi azaltırken performans artırılmış oldu. 1200 çalıştırma üzerinden %85 ve %90 başarı oranları ile kolun uç efektörü belirli bir konuma hareket ettirmeyi sağlamak için algoritma yeterince sağlam görünmektedir. Bazı durumlarda Gauss keşif yöntemi daha iyi performans gösterirken, diğer durumlarda OUP keşif yöntemi daha iyi performans göstermektedir. Ancak başarı oranı, ortalama mesafe ve standart sapmadaki fark nispeten küçüktür.

Pearson korelasyonu yaygın olarak iki rastgele değişken arasındaki ilişkiyi bulmak için kullanılır. Pearson korelasyon katsayısı, X ve Y değişkenleri tamamen aynıysa +1, tamamen farklıysa 0 ve zıt yönde tamamen aynıysa -1 değerine sahiptir. Çizelge 8 aynı tipteki üç robot kolun dinamikleri için homojenlik testinin sonuçlarını göstermektedir. İki tablodan da bilindiği gibi, farklı yönlere uygulanan kuvvetler 100 defa sabit olmasına rağmen motor ve sarkaç açıları farklı şekilde değişmektedir. Her bir robot kolun özellikle motor açısındaki değişiklik, sarkaç açısındaki değişiklikten daha çeşitlidir. Aynı tipte birden fazla robot kolun sonuç olarak, aynı üretim hattında üretilmelerine rağmen dinamikleri birbirinden biraz farklıdır. Bu, robot kol 2'nin olgun modelini aldıktan sonra bile İşçi 1 ve 3'te ki ek öğrenime hala ihtiyaç duyulduğu anlamına gelir.

Çizelge 8 Aynı türden birden çok robot kol cihazının dinamikleri için homojenlik testinin sonuçları.

(a) Motor açı değişikliklerinin bir pearson korelasyon matrisi				(b) Sarkaç açısı değişikliklerinin pearson korelasyon matrisi			
Motor Açısı	Kol 1	Kol 2	Kol 3	Sarkaç Açısı	Kol 1	Kol 2	Kol 3
Kol 1	1	0.77	0.86	Kol 1	1	0.98	0.96
Kol 2	-	1	0.75	Kol 2	-	1	0.98
Kol 3	-	-	1	Kol 3	-	-	1

Nicel olarak, cismin yer değiştirilmemesi ajana her bölüm sırasında bir nesneyi daha iyi kavrama şansı verir bu nedenle biriken ödülü en üst düzeye çıkarır. Daha uzun epizot süreleri göz önüne alındığında, başarı oranı yapay olarak arttırılabilir. Bununla birlikte, bu tür bir politikanın niteliksel analizinin aşırı derecede kaotik ve güvensiz olduğu düşünülmektedir. Robotik manipülasyonun gerçek dünyadaki uygulamaları, güvenlik standartlarını karşılamayı ve çevre ile daha yapılandırılmış bir etkileşim kullanmayı gerektirir. Bu işte eğitilmiş araçlar, bu tür garantileri sağlamak için mücadele eder ve gerçek robotlarda denetimsiz kullanımları, kazara hasar riskini azaltan uyumlu nesnelerle sınırlıdır. Bunu akılda tutarak, ayrık eylem alanları örneğin önceden tanımlanmamış eylem temelleri ve güvenlik sınırlarına sahip piksel bazında eylem alanı, görev çözme yeteneklerini geliştirecek daha karmaşık ilkeleri öğrenme yeteneğinin azalmasına rağmen, daha belirleyici davranışları nedeniyle şu anda gerçek dünya uygulamaları için daha uygun olabilir. Bu nedenle, sürekli uçtan uca kontrol ile gerçek dünyadaki robotik manipülasyon görevlerini çözmek için uygulanabilir olmadan önce RL için bir güvenlik teorisinin geliştirilmesi gerektiğine inanılmaktadır. Ablasyon çalışmaları bazı beklenmedik sonuçları getirmiştir. Özellikle gösterimlerin kullanılması, erken aşamalarda daha hızlı öğrenmeye rağmen, yeni sahnelerde ulaşılabilir başarı oranını %7 oranında azaltmıştır. Performanstaki bu önemli düşüşün, en sonunda yerel bir optimal politikaya yakınsamaya yol açan alt-optimal kodlanmış politika tarafından getirilen bir önyargıdan kaynaklandığı tartışılabilir. Tamamen sıfırdan keşfetmesi gereken ajanın, küresel olarak optimal olan ve bu tür önyargılardan etkilenmeyen bir politikaya yaklaşma şansı daha yüksektir. Bu nedenle, deneysel sonuçlar, mümkünse RL için gösterilerin kullanılmasından vazgeçilmesi gerektiğini ve bunun yerine müfredat öğrenimi gibi daha iyi garantileri olan diğer yöntemlerin uygulanması gerektiğini göstermektedir.

Proprioseptif gözlemlerin eklenmesi, başarı oranında %2'lik küçük bir artış sağlar. Bu gözlemler kolaylıkla elde edilebildiği için kullanımların faydalı olduğu düşünülmektedir. Benzer şekilde görsel gözlemlerde renk özniteliklerin kullanılması başarı oranını %10,5 oranında artırmaktadır ki bu çok önemli kabul edilmektedir. Bu nedenle eğitim için kullanılan bir ortam gerçekçi oluşturmayı destekliyorsa, bunların eklenmesi yararlıdır. Öznitelik çıkarıcı parametrelerin paylaşımı için yapılan analizler bazı ilginç sonuçları da ortaya çıkarmıştır. Her gözlem yığını için ayrı öznitelik çıkarıcılar kullanıldığında, ilk öğrenme, tek bir paylaşılan öznitelik çıkarıcının kullanımından çok daha hızlıdır. Bu mevcut olana kıyasla geçmiş gözlemler için farklı öznitelik çıkarıcılar çok daha fazla sayıda birleşik öğrenilebilir parametreye sahiptir. Bununla birlikte, her iki yaklaşım da çok benzer bir nihai başarı oranına ulaşabilir. Son yıllarda RL'nin büyük potansiyeline ve önemli ilerlemelerine rağmen, gerçek dünyadaki robotik manipülasyon görevleri için uygulanabilirliği hala sınırlıdır. DRL tarafından öğrenilen uçtan uca politikaların gerçek robotik sistemlere sağlam bir şekilde entegre edilebilmesi için ele alınması gereken birkaç zorluk vardır. Modelden bağımsız RL algoritmalarının örnek verimliliğini artırma girişimleri olmasına rağmen, deneyim tekrarına sahip politika dışı algoritmalar bile en uygun politikayı öğrenmek için binlerce kez geçiş gerektirir. Hiperparametrelere duyarlılık, RL'nin büyük ölçekli kullanımını sağlamak için ele alınması gereken bir diğer önemli sorundur. Hiperparametrelerin her görev için optimizasyonu, her denemenin uzun eğitim süresi nedeniyle çok zaman alan bir prosedürdür. Benzer şekilde, RL'de tekrar üretilebilirlik, birçok robotun çalıştığı ortamların yüksek stokastikliği nedeniyle sürekli görevler için çok zordur. Yüksek doğruluklu fizik ve işleme ile ucuz paralel simülasyonlar, yakın gelecekte bu sorunların bazılarını hafifletebilir. Bu nedenle, DRL'nin robotik manipülasyon alanında umut verici bir geleceğe sahip olacağına inanılmaktadır.

VII. SONUÇ VE GELECEK ÇALIŞMALAR

Bu çalışmada federe derin öğrenme ve federe derin pekiştirmeli öğrenme yaklaşımları kullanılarak robot kolların cisimleri tutabilirliğini öğretmeye çalışılmıştır. Federe öğrenme bilindiği üzere birden fazla alt sistemin sunucu üzerinde aynı anda eğitilme olanağı sağlamaktadır. Aynı zamanda eğitim gerçekleştirilirken alt sistemlerin verileri sunucuya gönderilmez bu da öğreticiye veri güvenliğini sağlar. Aynı zamanda alt sistemler aynı anda öğrendiklerinden birbirleriyle sinir ağlarının hatalarını ve ağırlıklarını paylaştıkları için öğrenme süreci daha hızlı gerçekleşmektedir. Görülmüştür ki federe derin pekiştirmeli öğrenme yaklaşımı standart federe derin öğrenme yaklaşımından daha iyi performans göstermiş ve daha hızlı çalışmıştır. Federe derin pekiştirmeli öğrenme yaklaşımının bir önemli yönü de başlangıçta ve eğitim gerçekleşirken herhangi bir veriye ihtiyaç duymadan öğrenmesidir. Bundan dolayı gelecekte İnsansız Hava Araçlarında federe pekiştirmeli öğrenme yaklaşımları kullanılabilir ve araçların daha önce hiç görmedikleri alanlarda hızlıca eğitilmesi sağlanabilir. Ağın algıdan öğrenebilmesi için öncelikle kuruluma bir kamera eklenebilir. İlk kullanım fikri bu olsa da, güncelleme oranları açısından V-REP simülasyonunda bir sınıra ulaşılmıştır. Algı eklemek öğrenmeyi iyileştirilebilir çünkü daha bilgilendirici bir durum sağlar. Deneysel kuruluma bir kamera eklemek kurulum için sonuçları iyileştirilebilse de, sahneye birden fazla nesne ekleyerek deneyin zorluğunu arttırmak, ağa nesnelerden hangisi olduğunu söyleyen daha üst düzey bir karar verme sürecinin eklenmesi gerekir. Toplu normalleştirme ve bırakma gibi düzenlileştirme teknikleri eklenerek başka bir iyileştirme yapılabilir. Bu çalışmada sunulmamasına rağmen, kısa süre içerisinde bunlarla ilgili deneyler yapılmasına rağmen, önemli bir gelişme görülmemiştir. Ancak sadece iki teknik denenmiştir ve sadece Tensorflow çerçevesi tarafından sağlanan standart değerler kullanılmıştır. Farklı parametrelerle iyileştirmeler olabilir. Ancak bu, çok fazla zaman harcanması gereken daha fazla deney gerektirmektedir. Farklı bir robotik kol kullanılarak da iyileştirmeler yapılabilir. Çalışmada Mico kolunun seçilmesinin tek nedeni, o zamanlar robotik

laboratuvarlarda mevcut olan bir robot kol olması ve gerçek dünyada da çok fazla kullanılmasıdır. Ancak bu projenin son aşamasında simülasyonda kullanılmak üzere yeni bir robotik kol, Panda robotik kol kullanıma sunulmuştur. Panda, lineer olarak hareket ettirilen iki parmaklı bir kavrayıcıya sahip 7 serbestlik dereceli bir robotik koldur. Bu robotik kolun ana avantajı, kuvvet algılamaya sahip olmasıdır, yani bir cisimle çarpışırsa bir nesneye ne kadar kuvvetle ittiğini algılayabilir ve ayrıca bir nesneyi tutarken kuvvet algılama özelliğine sahiptir. Bu geri bildirim, ödül işlevi için değerli bilgiler sağlayabilir. Diğer ilginç kısım ise 7 serbestlik dereceli olmasıdır. Bu 7. Eklem bölgesi kolun geri kalanı hala hareket edebilirken uç efektörün aynı pozisyonda kalmasına izin vermektedir. Bu ilave karmaşıklık, son efektörün nihai konumuna ulaşmasını kolaylaştırmalı, ancak aynı zamanda keşif sırasında karmaşıklığı artıracak ve muhtemelen aktör ve eleştirmen için öğrenmesini zorlaştıracaktır. Bu araştırma manipülasyon alanında yalnızca nispeten basit bir senaryoya odaklanmıştır.

VIII. KAYNAKÇA

KİTAPLAR

ELLİS, J. (2000). **Founding brothers: the revolutionary generation,** Vintage Books.

GARCİA, A. L. (2005). **Probability, Statistics, and Random Processes for Electrical Engineering,** Pearson Prentice Hall.

GERON, A. (2017). **Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems.** O'Reilly Media Inc.

GOODFELLOW, I., BENGİO, Y., COURVİLLE, A. (2017). **Deep Learning,** MIT Press.

HAGAN, M. T. (2014). **Neural Network Design Second Edition,** Stillwater.

İPLİKÇİ, S. (2017). **Optimizasyon Teknikleri,** Pamukkale Üniversitesi.

KOLTER, J. Z. (2016). **Reinforcement Learning,** Carnegie Mellon University.

KULKARNİ, P. (2012). **Reinforcement and systemic machine learning for decision making,** John Wiley & Sons.

NOCEDAL, J. and WRİGHT, S. J. (2006). **Numerical Optimization,** Springer.

RİCHARD, S. and ANDREW, G. (2015). **Introduction to reinforcement learning (Second Edition),** MIT Press.

RUSSEL, S. and NORVİG, P. (1995). **Artificial Intelligence A Modern Approach,** Prentice Hall.

SARANGAPANİ, J. (2006). **Neural Network Control of Nonlinear Discrete-Time Systems,** Taylor & Francis Group. https://doi.org/10.1201/9781420015454

SİLVER, D. (2015). **Reinforcement Learning Markov Decision Processes,** University College London.

STANDAGE, T. (2002). **The turk. The life and times of the famous eighteenth-century chess-playing machine,** Berkeley Books.

SUTTON, R. S. and BARTO, A. G. (1998). **Reinforcement learning: An introduction,** MIT Press.

SUTTON, R. S. and BARTO, A. G. (2018). **Reinforcement Learning An Introduction, Second Edition,** MIT Press.

WATKİNS, C. and DAYAN, P. (1992). **Technical Note: Q-Learning,** Springer.

WERBOS, P. (1992). **Approximate dynamic programming for real-time control and neural modeling,** Van Nostrand Reinhold.

WERBOS, P., Sİ, J., BARTO, A. G., POWELL, W. B., WUNSCH, D. (2004). **Handbook of learning and approximate dynamic programming,** Wiley-IEEE Press.

WİLLİAMS, J. K., HAUPT, S. E., PASİNİ, A., MARZBAN, C. (2009). **Artificial intelligence methods in the environmental sciences,** Springer.

YU, H. Z. (2020). **Taxonomy of Reinforcement Learning Algorithms,** Springer.

YÜCEL, İ. H. (1991). **Sanayide robot teknolojisi, uygulaması ve önemi,** T.C. Başbakanlık Devlet Planlama Teşkilatı.

MAKALELER

ADAM, S., BUSONİU, L., BABUSKA, R. (2012). "Experience Replay for Real-Time Reinforcement Learning Control", **IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews (99),** ss. 1-12. https://doi.org/10.1109/TSMCC.2011.2106494

ATKESON, C. G. and SCHAAL, S. (1997). "Learning tasks from a single demonstration", **IEEE International Conference on Robotics and Automation** sayı 2**,** ss. 1706-1712.

BHATNAGAR, S., SUTTON, R. S., GHAVAMZADEH, M., LEE, M. (2009). "Natural actor–critic algorithms", **Automatica** sayı 45, ss. 2471-2482. https://doi.org/10.1016/j.automatica.2009.07.008

BİCCHİ, A. (1995). "On the Closure Properties of Robotic Grasping", **The International Journal of Robotics Research** sayı 14, ss. 319-334. https://doi.org/doi.org/10.1177/027836499501400402

BONAWİTZ, K., EİCHNER, H., GRİESKAMP, W., HUBA, D. (2019). "Towards federated learning at scale: System design", **arXiv:1902.01046.**

BUCAK, I. and ZOHDY, M. (1999). "Application of reinforcement learning control to a nonlinear bouncing cart", **1999 American Control Conference,** ss. 5108-5113. https://doi.org/10.1109/ACC.1999.783230

BUCAK, O. and ZOHDY, M. (2001). "Reinforcement learning control of nonlinear multi-link system", **Engineering Applications of Artificial Intelligence,** ss. 563-575. https://doi.org/10.1016/S0952-1976(01)00031-8

BUCHLİ, J., THEODOROU, E., STULP, F., SCHAAL, S. (2010). "Variable Impedance Control A Reinforcement Learning Approach", **Conference: Robotics: Science and Systems VI.** https://doi.org/10.15607/RSS.2010.VI.020

CARUANA, R. (1997). "Multitask Learning", **Machine Learning** sayı 28, ss. 41-75. https://doi.org/doi.org/10.1023/A:1007379606734

CHEN, J., PAN, X., MONGA, R., BENGİO, S., JOZEFOWİCZ, R. (2016). "Revisiting Distributed Synchronous SGD", **arXiv:1604.00981.** https://doi.org/10.48550/arXiv.1604.00981

CHEN, T., GİANNAKİS, G. B., SUN, T., YİN, W. (2018). "LAG: Lazily aggregated gradient for communication-efficient distributed learning", **Advances in Neural Information Processing Systems.**

CRONİN, J., ESCANO, J. M., ROSHANY-YAMCHİ, S., CANTY, N. (2014). "Fuzzy-Based Generalized Predictive Control of a Robotic Arm", **25th IET Irish Signals & Systems Conference 2014.** https://doi.org/10.1049/cp.2014.0676

DE, S. and GOLDSTEİN, T. (2016). "Efficient Distributed SGD with Variance Reduction", **IEEE 16th International Conference on Data Mining (ICDM).** https://doi.org/10.1109/ICDM.2016.0022

DEİSENROTH, M. and RASMUSSEN, C. (2011). "PILCO: A Model-Based and Data Efficient Approach to Policy Search", **International Conference on Machine Learning,** ss. 465-472.

DİGNEY, B. (1996). "Nested Q-learning of hierarchical control structures", **IEEE international conference on neural networks** sayı 1, ss. 161-166.

DUAN, Y., CHEN, X., HOUTHOFT, R., SCHULMAN, J., ABBEEL, P. (2016). "Benchmarking Deep Reinforcement Learning for Continuous Control", **International Conference of Machine Learning.** https://doi.org/10.48550/arXiv.1604.06778

EL-GHAZALİ, T. (2020). "Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics", **ACM Computing Surveys,** ss. 1-30.

GLASCHER, J., DAW, N., DAYAN, P., O'DOHERTY, J. P. (2010). "States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning", **Neuron,** ss. 585-595. https://doi.org/10.1016/j.neuron.2010.04.016

GU, S., HOLLY, E., LİLLİCRAP, T., LEVİNE, S. (2017). "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates", **IEEE International Conference on Robotics and Automation.** https://doi.org/10.48550/arXiv.1610.00633

HASAN, A. T., HAMOUDA, A. S., ISMAİL, N., AL-ASSADİ, H. A. (2006). "An Adaptive- Learning Algorithm to Solve The Inverse Kinematics Problem of A 6 D.O.F Serial Robot Manipulator", **Advances in Engineering Software,** sayı 37, no 7, ss. 432-438.

HOFFMANN, H., THEODOROU, E., SCHAAL, S. (2012). "Behavioral experiments on reinforcement learning in human motor control. In Abstracts of the eighteenth annual meeting of neural control of movement", **IEEE Transactions on Robotics,** ss. 1360-1370.https://doi.org/10.1109/TRO.2012.2210294

IJSPEERT, A., NAKANİSHİ, J., SCHAAL, S. (2002). "Learning Attractor Landscapes for Learning Motor Primitives", **Advances in Neural Information Processing Systems** sayı 15.

INABA, M., KAGAMİ, S., KANEHİRO, F., HOSHİNO, Y. (2000). "A Platform for Robotics Research Based on the Remote-Brained Robot Approach", **International Journal of Robotics Research,** sayı 19, no 10. https://doi.org/10.1177/02783640022067878

IZAWA, J., KONDO, T., ITO, K. (2004). "Biological arm motion through reinforcement learning", **41st SICE annual conference vol. 1,** ss. 413-418. https://doi.org/10.1007/s00422-004-0485-3

JAN PETERS, S. S. (2008). "Reinforcement learning of motor skills with policy gradients". **Neural Networks,** ss. 682-697. https://doi.org/10.1016/j.neunet.2008.02.003

KAPPEN, H. (2005). "Path integrals and symmetry breaking for optimal control theory", **Journal of Statistical Mechanics: Theory and Experiment.** https://doi.org/10.1088/1742-5468/2005/11/P11011

KEHOE, B., MATSUKAWA, A., CANDİDO, S., KUFFNER, J., K. GOLDBERG. (2013). "Cloud-based robot grasping with the google object recognition engine", **IEEE International Conference on Robotics and Automation.** https://doi.org/10.1109/ICRA.2013.6631180

KEHOE, B., PATİL, S., ABBEEL, P., GOLDBERG, K. (2015). "A survey of research on cloud robotics and automation", **IEEE Transactions on Automation Science and Engineering,** sayı 12, no 2, ss. 398-409. https://doi.org/10.1109/TASE.2014.2376492

KHAN, S. G., HERRMANN, G., LEWİS, F. L., PİPE, T., MELHUİSH, C. (2012). "Reinforcement Learning and Optimal Adaptive Control: An Overview and Implementation Examples", **Annual Reviews in Control 36,** ss. 42-59.

KİM, B., KANG, B., PARK, S., KANG, S. (2008). "Learning robot stiffness for contact tasks using the natural actor-critic", **IEEE international conference on robotics and automation,** ss. 3832-3837. https://doi.org/doi.org/10.1109/ROBOT.2008.4543799

KİM, B., PARK, J., PARK, S., KANG, S. (2010). "Impedance Learning for Robotic Contact Tasks Using Natural Actor-Critic Algorithm", **IEEE Transaction on System, Man and Cybernetics, part B: Cybernetics,** sayı 4, ss. 433-443. https://doi.org/10.1109/TSMCB.2009.2026289

KİNGMA, D. P. and BA, J. L. (2015). "Adam: A Method For Stochastıc Optimization", **International Conference on Learning Representations (ICLR).** https://doi.org/10.48550/arXiv.1412.6980

KİUMARSİ, B., VAMVOUDAKİS, K., MODARES, H., LEWİS, F. L. (2018). "Optimal and Autonomous Control Using Reinforcement Learning: A Survey", **Neural Networks and Learning Systems** sayı 29, no 6, ss. 2042-2062. https://doi.org/10.1109/TNNLS.2017.2773458

KOBER, J. and PETERS, J. (2011). "Policy search for motor primitives in robotics", **Machine Learning,** ss. 171-203. https://doi.org/10.1007/s10994-010-5223-6

KOHL, N. and STONE, P. (2004). "Policy gradient reinforcement learning for fast quadrupedal locomotion", **International Conference on Robotics and Automation .**

KUAN, C. and YOUNG, K. (1998). "Reinforcement Learning and Robust Control for Robot Compliance Tasks", **Journal of Intelligent and Robotic Systems**, ss. 165-182. https://doi.org/10.1023/A:1008083631190

KUFFNER, J. (2010). "Cloud-enabled humanoid robots", **IEEE-RAS International Conference on Humanoid Robotics.**

LEVİNE, S., FİNN, C., DARRELL, T., ABBEEL, P. (2016). "End-to-end training of deep visuomotor policies", **Journal of Machine Learning Research,** ss. 1334-1373. HTTPS://DOİ.ORG/10.7746/JKROS.2019.14.1.040

Lİ, D., ZHAO, D., ZHANG, Q., CHEN, Y. (2018). "Reinforcement Learning and Deep Learning based Lateral Control for Autonomous Driving", **IEEE Computational Intelligence Magazine.** https://doi.org/10.48550/arXiv.1810.12778

LİLLİCRAP, T. P., HUNT, J. J. (2015). "Continuous control with deep reinforcement learning", **International Conference on Learning Representations .** https://doi.org/10.48550/arXiv.1509.02971

LİOUTİKOV, R., PARASCHOS, A., NEUMANN, G., PETERS, J. (2014). "Sample-based informationl-theoretic stochastic optimal control", **IEEE International Conference on Robotics and Automation (ICRA).** https://doi.org/10.1109/ICRA.2014.6907424

LİU, B., WANG, L., LİU, M. (2019). "Lifelong Federated Reinforcement Learning: A Learning Architecture for Navigation in Cloud Robotic Systems", **IEEE Robotics and Automation Letters,** ss. 4555-4562. https://doi.org/10.1109/LRA.2019.2931179

LOCHAN, K. and ROY, B. K. (2014). "Control of Two-link 2-DOF Robot Manipulator Using Fuzzy Logic Techniques: A Review", **Advances in Intelligent Systems and Computing.** https://doi.org/10.1007/978-81-322-2217-0_41

LUO, B., D. LİU, T. H., WANG, D. (2016). "Model-free optimal tracking control via critic-only Q-learning", **IEEE Trans. neural networks and learning systems,** sayı 27, no 10, ss. 2134-2144. https://doi.org/10.1109/TNNLS.2016.2585520

M. P. DEİSENROTH, (2013). "A survey on policy search for robotics", **Foundations and Trends in Robotics** sayı 2. no 1-2, ss. 1-142. https://doi.org/10.1561/2300000021

MACMAHAN, H., MOORE, E., RAMAGE, D. (2016). "Communication-Efficient Learning of Deep Networks from Decentralized Data", **Artificial Intelligence and Statistics Conference .**

MCGOVERN, A. and BARTO, A. G. (2001). "Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density", **Proceedings of the International Conference on Machine Learning** , ss. 147-163.

MCMAHAN, H. and MOORE, E. (2016). "Communication-Efficient Learning of Deep Networks from Decentralized Data", **arXiv:1602.05629.**

MNİH, V. (2015). "Human-level control through deep reinforcement learning", **Nature** sayı 518, no 7540, ss. 529-533.

MNİH, V., BADİA, A. P., MİRZA, M. (2016). "Asynchronous Methods for Deep Reinforcement Learning", **International Conference of Machine Learning,** ss. 1928-1937. https://doi.org/10.48550/arXiv.1602.01783

MNİH, V., KAVUKCUOGLU, K., SİLVER, D. (2015). "Human-level control through deep reinforcement learning", **Nature** sayı 518, ss. 529-533. https://doi.org/https://doi.org/10.1038/nature14236

MNİH, V., KAVUKCUOGLU, K., SİLVER, D. (2013). "Playing Atari with Deep Reinforcement Learning", **Deep Learning Workshop NIPS 2013.** https://doi.org/10.48550/arXiv.1312.5602

MOHAMMED, M. A., HUİ, L., NORBERT, S., KERSTİN, T. (2016). "Multiple Lab Ware Manipulation in Life Science Laboratories using Mobile Robots", **17th International Conference on Mechatronics- Mechatronika (ME).**

MOLDOVAN, T., LEVİNE, S., JORDAN, M., ABBEEL, P. (2015). "Optimism-driven exploration for nonlinear systems", **IEEE International Conference on Robotics and Automation,** ss. 3239-3246. https://doi.org/10.1109/ICRA.2015.7139645

PASTOR, P., HOFFMAN, H., ASFOUR, T., SCHAAL, S. (2009). "Learning and Generalization of Motor Skills by Learning from Demonstration", **IEEE International Conference on Robotics and Automation.** https://doi.org/10.1109/ROBOT.2009.5152385

PETERS, J. and SCHAAL, S. (2008). "Natural Actor-Critic", **Neurocomputing** sayı 71, ss. 1180-1190. https://doi.org/10.1016/j.neucom.2007.11.026

PLATT, R. (2007). "Learning grasp strategies composed of contact relative motions", **7th IEEE-RAS International Conference on, IEEE,** ss. 49-56. https://doi.org/10.1109/ICHR.2007.4813848

RİEDMİLLER, M. (2005). "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method", **European Conference on**

Machine Learning, ss. 317-328. https://doi.org/https://doi.org/10.1007/11564096_32

RİEDMİLLER, M., GABEL, T., HAFNER, R., LANGE, S. (2009). "Reinforcement learning for robot soccer", **Autonomous Robots,** sayı 27(1), ss. 55-73. https://doi.org/10.1007/s10514-009-9120-4

SCHAAL, S. (1997). "Learning from demonstration", **Advances in Neural Information Processing Systems 9,** ss. 1040-1046.

SCHAAL, S., MOHAJERİAN, P., IJSPEERT, A. (2007). "Dynamics systems vs. optimal control--a unifying view", **Progress in Brain Research** sayı 165, ss. 425-445. https://doi.org/10.1016/S0079-6123(06)65027-9

SHAH, H. and GOPAL, M. (2009). "Reinforcement learning control of robot manipulators in uncertain environments", **IEEE international conference on industrial technology,** ss. 1-6. https://doi.org/10.1109/ICIT.2009.4939504

SİLVER, D., HUANG, A., MADDİSON, C., GUEZ, A. (2016). "Mastering the game of Go with deep neural networks and tree search", **Nature,** sayı 529, no 7587, ss. 484-489. https://doi.org/10.1038/nature16961

SMART, W. D. and KAELBLİNG, L. P. (2002). "Effective Reinforcement Learning for Mobile Robots", **IEEE International Conference on Robotics and Automation,** ss. 3404-3410.

SMART, W. and KAELBLİNG, L. (2002). "Reinforcement Learning for Robot Control", **The International Society for Optical Engineering** sayı 4573, ss. 92-103. https://doi.org/10.1117/12.457434

SMİTH, V., CHİANG, C. K., SANJABİ, M., TALWALKAR, A. S. (2017). "Federated Multi-Task Learning", **Advances in Neural Information Processing Systems 30 (NIPS 2017),** ss. 4424-4434.

SMYS, S. and RANGANATHAN, G. (2019). "Robot Assisted Sensing, Control and Manufacture in Automobile Industry", **Inventive Research Organization,** ss. 180-187. https://doi.org/10.36548/jismac.2019.3.005

STİNGU, P. and LEWİS, F. (2011). "Adaptive dynamic programming applied to a 6DoF quadrotor", **IGI Global.** https://doi.org/10.4018/978-1-60960-551-3.CH005

STULP, F., THEODOROU, E., KALAKRİSHNAN, M. (2011). "Learning motion primitive goals for robust manipulation", **2011 IEEE/RSJ International**

Conference on Intelligent Robots and Systems, ss. 325–331. https://doi.org/10.1109/IROS.2011.6094877

SUTTON, R. (1999). "Policy gradient methods for reinforcement learning with function approximation", **Advances in Neural Information Processing Systems.**

TAMPUU, A., MATİİSEN, T., D.KODELJA, KUZOVKİN, I. (2017). "Multiagent cooperation and competition with deep reinforcement learning", **PLoS ONE** sayı 12, no 4, ss. 1-15. https://doi.org/10.1371/journal.pone.0172395

THEODOROU, E., BUCHLİ, J., SCHAAL, S. (2010). "Reinforcement learning of motor skills in high dimensions: A path integral approach", **IEEE international conference on robotics and automation,** s. 2397-2403.

THEODOROU, E., PETERS, J., SCHAAL, S. (2007). "Reinforcement learning for optimal control of arm movements", **Abstracts of the 37st Meeting of the Society of Neuroscience.**

TODOROV, E., EREZ, T., TASSA, Y. (2012). "Mujoco: A physics engine for modelbased control", **IEEE/RSJ International Conference on Intelligent Robots and Systems,** ss. 5026-5033. https://doi.org/10.1109/IROS.2012.6386109

TSOKALO, I. A., WU, H. (2019). "Mobile Edge Cloud for Robot Control Services in Industry Automation", **2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC).** https://doi.org/10.1109/CCNC.2019.8651759

TSURUMİNE, Y., CUİ, Y., ECHİBE, E., MATSUBARA, T. (2019). "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation", **Robotics and Autonomous Systems** sayı 112, ss. 72-83. https://doi.org/10.1016/j.robot.2018.11.004

WATKİNS, C. (1989). "Learning from Delayed Rewards", **King's College.**

WERBOS, P. (2009). "Intelligence in the brain: A theory of how it works and how to build it", **Neural Networks,** ss. 200-212.

WERBOS, P. J. (2008). "ADP: The key direction for future research in intelligent control and understanding brain intelligence", **IEEE Transactions on Systems, Man, and Cybernetics,** ss. 898-900.

WERBOS, P., PERLOVSKY, L., KOZMA, R. (2007). "Neurodynamics of cognition and consciousness. Understanding complex systems", **Springer.**

WİLLİAMS, R. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning", **Machine Learning** sayı 8, no 3-4, ss. 229-256.

XU, H., GAO, Y., YU, F., DARRELL, T. (2017). "End-to-end Learning of Driving Models from Large-scale Video Datasets", **Computer Vision and Pattern Recognition,** ss. 2174-2182. https://doi.org/10.48550/arXiv.1612.01079

YANG, T., ANDREW, G. (2018). "Applied Federated Learning: Improving Google Keyboard Query Suggestions", **arXiv:1812.02903.**

ZEHONG CAO (2019). "Reinforcement Learning from Hierarchical Critics", **IEEE Transactions on Neural Networks and Learning Systems.** https://doi.org/10.48550/arXiv.1902.03079

ZHOU, W., Lİ, Y., CHEN, S., DİNG, B. (2018). "Real-Time Data Processing Architecture for Multi-Robots Based on Differential Federated Learning", **IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation.** https://doi.org/10.1109/SmartWorld.2018.00106

ZHUO, H. H., FENG, W., XU, Q., YANG, Q., LİN, Y. (2019). "Federated Reinforcement Learning", **arXiv, 1901.08277.**

ELEKTRONİK KAYNAKLAR

ROSHEİM, M. E. "Leonardo's programmable automaton. A reconstruction. Anthrobot", http://www.anthrobot.com/press/article_leo_programmable.html, (Erişim Tarihi: 2022, 05 21).

TEZLER

GASKETT, C. (2002). "Q-Learning for Robot Control". (Doktora Tezi), The Australian National University, https://doi.org/10.25911/5d7a2a09a7dfd

DİĞER KAYNAKLAR

POLLARD, W. (1942). *ABD Patent No. 2,286,571*

RORESLUND, H. (1944). *ABD Patent No. 4,344,108.*

ÖZGEÇMİŞ

Adı-Soyadı **:** MURAT UĞUR GÜLLE

ÖĞRENİM DURUMU:

Lisans: 2017, Gazi Üniversitesi, İktisadi ve İdari Bilimler Fakültesi, Ekonometri Bölümü

Yüksek Lisans: İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği Fakültesi, Bilgisayar Mühendisliği Bölümü

MESLEKİ DENEYİM:

Veribilim Yazılım Bilgisayar San. Tic. Ltd, Yapay Zeka Uzmanı, Ocak 2021-Günümüz

YAYINLAR:

Kiani, F., Seyyedabbasi, A., Aliyev, R., Gülle, M. U., Shah, M. A. (2021). "Adapted-RRT: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms". *Neural Computing and Applications vol 33***.** ss. 15569-15599

Seyyedabbasi, A., Aliyev, R., Gülle, M. U., Kiani, F. (2021). "Hybrid algorithms based on combining reinforcement learning and metaheuristic methods to solve global optimization problems". *Knowledge-Based Systems vol 223*

Kiani, F., Seyyedabbasi, A., Aliyev, R., Gülle, M. U., Shah, M. A. (2021). "3D path planning method for multi-UAVs inspired by grey wolf algorithms". *Journal of Internet Technology vol 22,* ss. 743-755