TÜRKİYE CUMHURİYETİ
MARMARA ÜNİVERSİTESİ
FEN - EDEBİYAT FAKÜLTESİ

PEKİŞTİRMELİ ÖĞRENME (REINFORCEMENT LEARNING) ile VIZDOOM OYUNUNUN OTOMASYONU

HARUN BERKİN ÇETİN
MATHİLDA CEREN KESKİN
MEVLÜTHAN KESİMLİ
ÖMER PALABIYIK

LİSANS TEZİ

İSTATİSTİK ANABİLİM DALI

DÖNEM DANIŞMANI
Prof. Dr. DENİZ İNAN

2023 – İSTANBUL

BEYAN

Bu tez çalışmamızın tümüyle kendimize ait olduğunu, tezin planlanmasından tamamlanma sürecine kadar bütünüyle etik dışı davranışlarımızın olmadığını, bu tezdeki bilgilerin tümünün akademik ve etik kurallar içinde elde ettiğimizi, bu tez çalışması ile elde edilmemiş bütün bilgi ve yorumlara kaynak gösterdiğimizi ve bu kaynakları da kaynaklar listesinde belirttiğimizi, yine bu tezin çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici davranışlardan kaçınıldığını beyan ederiz.

HARUN BERKİN ÇETİN
MATHİLDA CEREN KESKİN
MEVLÜTHAN KESİMLİ
ÖMER PALABIYIK

TEŞEKKÜRLER

Lisans süreci boyunca değerli bilgilerini esirgemeyen başta Prof. Dr. Müjgan Tez olmak üzere, dönem danışmanımız ve bölüm başkanımız Prof. Dr. Deniz İnan, tez sürecinde bizlere desteklerini esirgemeyen Prof. Dr. Birsen Eygi Erdoğan lisans derslerimizde engin bilgileriyle bizleri aydınlatan Prof. Dr. Ufuk Yolcu ve Prof. Dr. Özge Cağcağ Yolcu hocalarımıza sonsuz teşekkürler.

ÖZET

Yapay zekâ teknolojisini geliştirmek için literatürde model tabanlı yaklaşım kullanılmaktadır. Model tabanlı yaklaşıma alternatif olarak makine öğrenmesinin gelişmesi ve akıllı algoritmalar oluşturulmasıyla her alanda makine öğrenmesi kullanılmaya başlanmıştır. Robotik sistemlerin geliştirilmesi sürecinde sistemin matematiksel modelinin oluşturulması ve kontrol algoritmalarının geliştirilmesi gerekmektedir. Geliştirilen matematiksel modellerin fiziksel sistemleri ifade etmesi gerekmektedir fakat model üzerindeki belirsizlik, analitik çözümün olmaması gibi durumlarda model tabanlı kontrol algoritmaları beklenen performansı üretmemektedir. Bu tez çalışmasında pekiştirmeli öğrenme ile bilgisayara "DOOM" bilgisayar oyununu kendi kendine oynayabilmesi hedeflenmiştir. DOOM oyununun seçilmesindeki amaç ise bu oyunun birincil bakış açılı oyunlarının atası sayılması ve bir milat olarak tanımlanmasıyla oyun dünyasında ayrı bir yerinin olmasıdır. Birinci kısımda pekiştirmeli öğrenmenin temellerinden bahsedilmiştir. İkinci kısımda ise DOOM ile modellemelerin nasıl yapıldığından bahsedilmiştir. Sonuç bölümünde ise pekiştirmeli öğrenmenin yakın geleceği ve istatistik bilimine olan etkilerinden bahsedilmiştir

ABSTRACT

Model-based approach is used in the literature to develop artificial intelligence technology. As an alternative to the model-based approach, machine learning has started to be used in every field with the development of machine learning and the creation of smart algorithms. In the process of developing robotic systems, it is necessary to create a mathematical model of the system and develop control algorithms. The developed mathematical models should express physical systems, but in cases such as uncertainty on the model and lack of analytical solution, model-based control algorithms do not produce the expected performance. In this thesis, it is aimed to enable the computer to play the "DOOM" computer game by itself with reinforcement learning. The purpose of choosing the DOOM game is that this game has a special place in the game world as it is considered the ancestor of the primary point of view games and is defined as a milestone. In the first part, the basics of reinforcement learning are mentioned. In the second part, how to make models with DOOM is mentioned. In the conclusion part, the near future of reinforcement learning and its effects on statistics are mentioned.

(Tezin yazım sürecinde Marmara Üniversitesi tez formatından yararlanılmıştır.
https://ebe.marmara.edu.tr/dosya/ebe/Tez_Yaz%C4%B1m_K%C4%B1lavuzu%20(1).pdf)

İÇİNDEKİLER

GİRİŞ

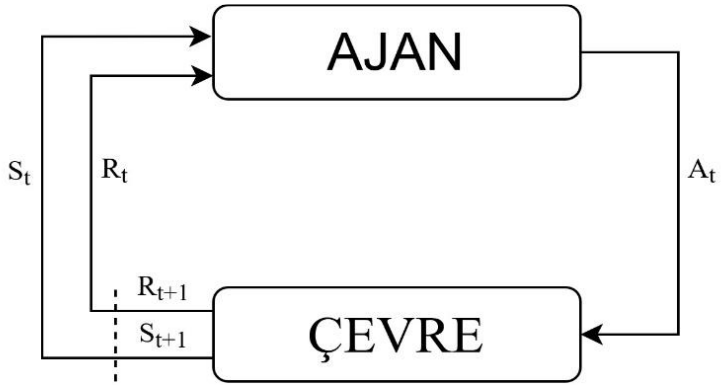
İstatistiksel birçok metodun bilgisayar bilimlerindeki gelişmelerle birleşmesi birlikte istatistiksel modeller ve çalışmalar farklı bir boyuta geçmiştir. Yakın döneme baktığımızda bunun en başında makine öğrenmesi gelmektedir. Makine öğrenmesinin bir alt kolu olan pekiştirmeli öğrenme algoritmaları robotik, otonom araçlar, finans, biyoloji vb. birçok alanda uygulanmaktadır.

Günümüzde yapay zekâ çalışmalarının büyük bir çoğunluğu makine öğrenmesi / yapay öğrenme konusunda yoğunlaşmaktadır. Makine öğrenmesinde problemlerin modellenme biçimi temel olarak danışmanlı öğrenme, danışmansız öğrenme ve pekiştirmeli öğrenme olarak üç sınıfta incelenmektedir. Makine öğrenmesi yöntemleri veri odaklı olup gerçek hayat problemlerine çözüm üretmektedirler. Makine öğrenmesi paradigması olan danışmanlı öğrenme konseptinde, makine öğrenmesi algoritması için giriş ve çıkış verisi bulunmaktadır. Çıkış verileri kategorik veya sembolik veriler olabilir. Danışmanlı öğrenmenin çözmeye çalıştığı problemler temelde, sınıflandırma ve regresyon problemine dönüştürülmektedir. Sınıflandırma probleminin çözümüyle örüntü tanıma, yüz tanıma, parmak izi tanıma, plaka tanıma, spam mail tanımlama, karar verme, tıbbi teşhis, hava durumu tahmin gibi birçok alanda uygulama yapılmaktadır. Regresyon probleminin çözümüyle süreç modelleme, zaman serisi tahmini, kontrol teorisi vb. uygulama alanı bulunmaktadır. Diğer bir makine öğrenmesi paradigması danışmansız öğrenmedir. Bu problem tipi temelde kümeleme problemleriyle uğraşmaktadır. Danışmansız öğrenme modelinde gerçek hayattan veya bir süreçten toplanmış giriş verilerimiz bulunmaktadır. Giriş verilerimize karşılık düşen etiket verileri bulunmadan, geliştirilen makine öğrenmesi modelleri veri içerisindeki kümelenmeleri belirlemede, anomali belirleme, karar verme, örüntü çıkarma vb. birçok problemi çözmektedir. Diğer bir paradigma olan pekiştirmeli öğrenmede, yapay modelimiz için giriş ve çıkış verileri bulunmamaktadır, yapay modelimizin bulunduğu ortam ile etkileşime girerek belirlenen hedefi yerine getirmesi için ödül toplaması ve toplanan bu ödüllerin maksimum veya minimum seviyede oluşturulmasını hedeflemektedir. Makine öğrenmesinin problemlerine çözüm olarak, makine öğrenmesi algoritmaları kullanılmaktadır. Bu algoritmalar sayesinde danışmanlı, danışmansız ve pekiştirmeli öğrenme ile problemler çözülmektedir. Bu tez çalışmasında bölüm ikide ifade edilen model tabanlı, robot kolun tork kontrolü problemine pekiştirmeli öğrenme modeli geliştirilerek çözüm oluşturulacaktır. Günümüzde pekiştirmeli öğrenme finans, robotik, havacılık vb. birçok kritik uygulamada birçok başarı kaydetmiştir. Pekiştirmeli öğrenme optimal kontrol teorisinin çözemediği birçok problemi çözmektedir. (Sutton, Richard S.)

1. PEKİŞTİRMELİ ÖĞRENME YAKLAŞIMI

1.1 Pekiştirmeli Öğrenmenin Temelleri

Pekiştirmeli (takviyeli) öğrenme, bilinmeyen bir dinamik ortamda oluşturulan yapay ajanın belirlenen görevi yerine getirmek için çevre ile etkileşime girmesi ve etkileşim sonucunda ödül değerlerini toplamasıdır. Pekiştirmeli öğrenme yaklaşımında yapay ajan çevre ile etkileşim sonucunda topladığı ödül değerini maksimum veya minimum seviyeye çekmeyi hedeflemektedir. Yapay ajan, çevreyle her etkileşim sonucunda çevre içerisinde bir durumda bulunmaktadır. Ajan bulunduğu durumda bir dizi aksiyonlar gerçekleştirip bir karar almaktadır. Ajan aldığı kararlar sonucunda çevre içerisinde yeni bir duruma geçmektedir ve bu durum sonucunda ajan çevreden pozitif veya negatif ödül toplamaktadır. Toplanan ödüllerin maksimum seviyeye geçmesi ve yapay ajanın vermiş olduğu sıralı kararlar sonucunda hedeflenen işi yapmayı öğrenmesi beklenmektedir. Bu durum döngüsel olarak şekil 2.1'de görülmektedir. (Sutton, Richard S.)



Şekil 2.1 : Pekiştirmeli Öğrenme Ajan ve Ortam İlişkisi

Şekil 1.1'de Rt yapay ajanın çevreden almış olduğu ödülü ifade etmektedir. At yapay ajanın çevreye göndermiş olduğu aksiyonları ifade etmektedir. St yapay ajanın çevre içerisinde bulunduğu durumları ifade etmektedir. Yapay ajan herhangi bir t anında, St durumunda bulunurken At aksiyonunu gerçekleştirerek Rt+1 ödülünü almakta ve buna bağlı olarak St+1 durumuna geçmektedir.

1.1.1 Pekiştirmeli Öğrenmede Kullanılan Kavramlar

- ✓ **Ajan:** Çevreyi algılayan/keşfeden ve buna göre hareket edebilen yapı.

- ✓ **Çevre:** Ajanın içinde bulunduğu ve bağlı olduğu yer, çevre ajanın öğrenme gerçekleştireceği ve etkileşime girebileceği biçimdedir. Ajanın içinde bulunduğu çevre kısmen gözlemlenebilir veya tamamen gözlemlenebilirdir. Pekiştirmeli öğrenmede ajanın içinde bulunduğu ortam ve gözlemler rastsal olabilir.

- ✓ **Aksiyon:** Aksiyonlar, ajan tarafından ortam içerisinde alınan kararlar olarak düşünülebilir ve $a \in A$ ile ifade edilir. Ajanın alacağı aksiyonlar sürekli değerli veya ayrık değerli olabilmektedir.

- ✓ **Ödül:** Ajanın gerçekleştirdiği eylemler sonucunda çevreden almış olduğu geribildirimlerdir. Ödül pozitif veya negatif olabilir.

- ✓ **Durum:** Durum ajanın çevre içerisinde ne şekilde bulunduğu bilgisini tutar ve $s \in S$ olarak ifade edilir. Çevrenin içerisinde, ajan için gerekli minimum bilgiyi taşımakta olup, çevreye ve problemin tanımına göre sürekli zamanlı ve ayrık zamanlı olarak ifade edilmektedir.

- ✓ **Yörünge:** Ajanın pekiştirmeli öğrenme algoritması sonucunda gerçekleştirdiği hedefe bağlı olarak, içinde bulunduğu durum ve bir sonraki durumlar arasındaki geçişlerin hepsini kapsamaktadır. Pekiştirmeli öğrenme ile ajanın belirlenen hedefi öğrenmesi ile, $\{s1, s2,...,sN\}$ durumlar kümesi yörüngemizi oluşturmaktadır.

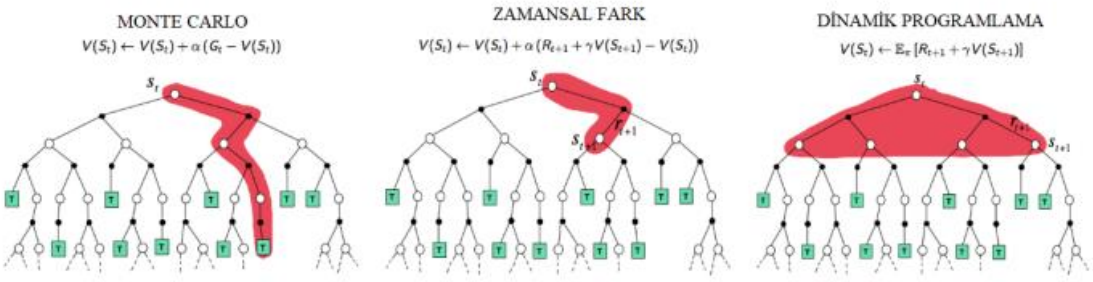
- ✓ **Politika:** Politika, t anında stdurumunda bulunan ajanın aksiyon kümesi Atiçerisinden nasıl bir aksiyon seçeceğini belirlemektedir. Politika mevcut durum $\boldsymbol{st}$ile ataksiyonunu birbirine haritalamaktadır. Ajanın takip edeceği politika stratejisi, deterministik veya stokastik olarak modellenebilmektedir.

1.1.2 Pekiştirmeli Öğrenme Modelleri

Pekiştirmeli öğrenme algoritmaları çok çeşitli olarak sınıflandırılmaktadır. Pekiştirmeli öğrenmeyi model tabanlı ve modelden bağımsız olarak iki sınıfa ayırmak mümkündür. Model tabanlı algoritmalar çevre hakkında öncül bir bilgi veya çevreyi ifade eden matematiksel model üzerinde oluşturulmuştur. Model tabanlı pekiştirmeli öğrenme optimal kontrol metodu olan model öngörülü kontrole (MPC) benzerlik göstermektedir. Modelden bağımsız pekiştirmeli öğrenme algoritmaları, çevre hakkında hiçbir bilgiye sahip olmamaktadır. Çevre genellikle rastgele süreç olan Markov karar süreci ile modellenerek algoritmalar geliştirilmektedir. Yapay ajan çevreyi keşfederek çevre hakkında bilgi toplayarak optimal kararlar vermektedir. Bu süreç pekiştirmeli öğrenme yaklaşımını deneme, yanılma ve ödül toplama olarak oluşturmaktadır. Pekiştirmeli öğrenme günümüzde birçok alanda uygulanmakta olup, başarı elde etmesinin en temel sebeplerinden biri çevre hakkında bir modele ihtiyaç duymamasıdır. Bu sayede birçok mühendislik problemini çevre içerisinde etkileşime girerek çözmektedir. Bu tez çalışmasında modelden bağımsız pekiştirmeli öğrenme algoritmaları üzerinde durulacaktır. Pekiştirmeli öğrenmenin bir diğer sınıflandırması sürekli veya ayrık durum/aksiyon çiftleri üzerindedir. Bu sınıflandırmada ajanın içerisinde bulunduğu çevrenin durumları sürekli zamanlı veya ayrık zamanlı olabilmektedir. Ajanın çevre içerisinde etkileşime girmek için uyguladığı aksiyonlar sürekli ve ayrık olabilmektedir. Ajanın içinde bulunduğu ortam sonlu boyutlu ayrık aksiyon ve durum çiftinden oluşuyorsa pekiştirmeli öğrenme algoritmaları tablo yöntemlerini temel alan algoritmalarla çözüm üretmektedir. Ajanın etkileşimde bulunduğu çevre veya aksiyon çiftinin sürekli değerler alması durumunda tablo yönteminde önerilen algoritmalar etkili olmamaktadır. Bu tez çalışmasında sürekli ve ayrık aksiyon ve durum çiftleri ile oluşturulan algoritmalar üzerinde durulacaktır. Pekiştirmeli öğrenmenin diğer bir sınıflandırılması, değer tabanlı ve politika tabanlı algoritmalardır. Değer tabanlı pekiştirmeli öğrenme algoritmaları, yapay ajanın çevre içerisinde bulunduğu bir st durumunda belirli bir at eyleminde bulunmanın değerlerini öğrenmektedir. Değer tabanlı yaklaşımda algoritmaların uygulamış olduğu politika, ajanın çevreye uyguladığı eylemlerin getirisinin hangisi en yüksek ise onun seçmesi şeklindedir. Böyle bir yaklaşım aç gözlü yaklaşım olarak isimlendirilmektedir. Ajan bulunduğu durumda uyguladığı eylemlerin hangisi en büyük getiriyi sağlıyorsa bu aksiyonu seçmektedir. Böylece ajanın politikası aç gözlü yaklaşım olarak isimlendirilmektedir. Politika tabanlı pekiştirmeli öğrenme algoritmalarında, ajanın içinde bulunduğu st durumlarının değerlerini öğrenmek yerine uygun politikayı keşfetmeyi hedeflemektedir. Bu yaklaşımda ajanın takip ettiği aç gözlü yaklaşımdan farklı olarak her bir durumda uygun kararlar vermesini sağlamak amaçlanmaktadır. Böylece ajan içinde bulunduğu ortam içerisinde politikayı öğrenerek bir davranış biçimi geliştirmektedir. Değer tabanlı yaklaşım ve politika tabanlı yaklaşım ile uygulanan probleme göre algoritmalar farklılık göstermektedir. Değer tabanlı ve politika tabanlı yaklaşımların beraber kullanıldığı pekiştirmeli öğrenme algoritmaları mevcuttur. Böyle bir durumda ajan st durumunda olmanın getirdiği değer ve uygun politikaları öğrenmektedir. Pekiştirmeli öğrenmede algoritmaların diğer bir sınıflandırma biçimi açık politika ve kapalı politikadır. Açık politikada ajan,

eğitim esnasında her adımda uyguladığı politikayı değiştirmektedir böylece en güncel politika üzerinden kararlar almaktadır. Kapalı politikada ajan çevre içerisinde bilgi topladıktan sonra politikasını güncellemektedir. Kapalı politika yaklaşımı ajanın daha kararlı öğrenmesini sağlamaktadır. Ajanın her adımda güncellediği politika, uygun politika olmayıp yanlış bir karar almasına sebep olabilir. Pekiştirmeli öğrenme yaklaşımlarından model tabanlı yaklaşımda ortamın tam bir matematiksel modeline sahip olduğumuz için kapsamlı bir arama algoritmasıyla, yani tüm olası sonuçları araştırarak sonuca varabiliriz. Arama işlemi işlemsel karmaşıklık sebebiyle basit uygulamalarda örneğin tic-tac-toe oyununda uygulanabilmektedir. Arama karmaşıklığı ortamın durumları ve aksiyonlarına bağlı olarak artmaktadır. s = 100 durumlu bir ortam hakkında olası durum uzayımızın boyutu 2 100 ve her bir durum için olası aksiyonlar düşünüldüğünde çevre modelinin bildiği varsayımıyla arama yöntemiyle sonuç bulmak birçok problem için pratik değildir. Çevre hakkında matematiksel bir modele sahip olduğumuz durumda, çözüm için kullanılan diğer bir yaklaşım ise dinamik programlamadır. Dinamik programlamada olası durum ve aksiyonlara bağlı olarak kapsamlı arama yapmak yerine optimal politikayı iteratif olarak çözmemizi sağlar. Dinamik programlama optimal çözüm için bir metodoloji geliştirmemizi sağlar ve iteratif olarak çözümü hesaplar. Çevre hakkında kesin bir bilgiye sahipsek, dinamik programlama ve kapsamlı arama yaklaşımları optimal politikayı garanti etmektedir. Pekiştirmeli öğrenme yaklaşımlarının çoğu, çevre hakkında detaylı bir öncül bilgi olmadığını ve bazı algoritmalarda çevrenin tüm durumlarına erişilemediğini (POMDP) varsaymaktadır. Fiziksel dünyada kurulan sistemler düşünüldüğü zaman çoğunlukla zamanla değişen, belirlenemeyen dinamikler, gürültü vb. problemler karşımıza çıkmaktadır. Bu sebeplerden birçok fiziksel sistem stokastik davranışlar sergilemektedir. Dinamik programlama yaklaşımı deterministik veya stokastik olmasına karşılık matematiksel modele ihtiyaç duymaktadır. Pekiştirmeli öğrenmenin modelden bağımsız yaklaşımı sayesinde dinamik programlamanın yaklaşık bir çözümü oluşturulmaktadır. Dinamik programlama bir önceki bölümde ifade edilen optimal kontrol problemi içinde uygulanmaktadır. Optimal politikayı yani kontrol işaretini belirli bir amaç fonksiyonunu minimize ederek oluşturmaktadır. Pekiştirmeli öğrenmenin modelden bağımsız yaklaşımında çevre ile etkileşime girerek oluşturulan ödül fonksiyonu sayesinde, yapay ajanın topladığı ödülü maksimize veya minimize etmeyi amaçlarız. Modelden bağımsız pekiştirmeli öğrenmede iki temel yaklaşım olan, Monte Carlo ve zamansal fark metotları kullanılmaktadır. Monte Carlo yaklaşımında, bir bölüm sonunda veya ortam içerisinde son duruma ulaşıncaya kadar algoritma kendini güncellememektedir. Bölümün sonuna ulaşıldığında algoritmanın güncellemesi oluşturulacaktır. Zamansal fark metodunda belirli bir zaman aralığında algoritmanın güncellemesi gerçekleşmektedir. Zamansal fark algoritmaları genellikle TD ile ifade edilmektedir ve matematiksel olarak $TD(0)$, $TD(1)$ şeklinde ifade edilir. İfade biçimine bağlı olarak en genel gösterim $TD(\lambda)$, $\lambda \in [0,1]$ olarak gösterilmektedir. $TD(0)$ yaklaşımı zamansal fark olarak sadece bir adım ileriyi incelediğimizi ifade etmektedir. $TD(1)$ yaklaşımı Monte Carlo yaklaşımına eşdeğerdir yani bölüm sonunda güncelleme oluşturduğumuz anlamındadır. Böylece $TD(\lambda)$ ifadesinde λ terimi, algoritmanın gelecekte nasıl bir ufuk inceleyeceğini belirtmektedir. $TD(0)$ ve $TD(1) = MC$ metodu arasındaki farklar,

MC metodu yüksek varyansa sahipken TD metodu düşük varyansa sahiptir. Bunun sebebi MC metodunun bir bölüm boyunca bütün aksiyon durum çifti üzerinden güncelleme yaparken, TD metodu tek bir adımda güncelleme kuralına sahip olmasıdır. MC metodu parametreleri güncellemek için bir bölümü göz önüne aldığı için düşük yanlılık değerine sahipken, TD metodu anlık güncellemeler üzerinden ilerlediği için yüksek yanlılık değerine sahiptir. Bu tez çalışmasında incelenen algoritmalar TD metodunu kullanarak oluşturulmuştur. TD metodu temelde Sarsa ve Q öğrenme algoritmaları olarak iki temel algoritmaya sahiptir. Model tabanlı algoritmalar ve modelden bağımsız yaklaşım şekil 2.2'de görülmektedir. (Sutton, Richard S.)



Şekil 2.2: Pekiştirmeli Öğrenme Metotları.

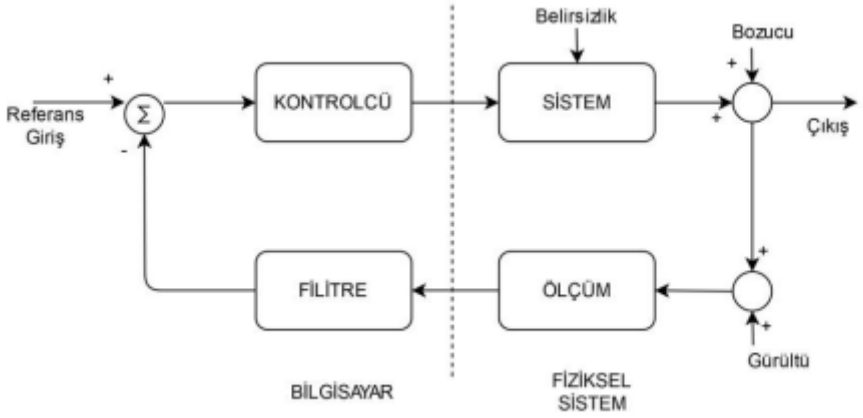
1.1.3 Pekiştirmeli Öğrenme Keşif ve Sömürü

Modelden bağımsız pekiştirmeli öğrenme yaklaşımında ajanın çevre içerisinde hiçbir bilgiye sahip olmadığı varsayılmaktadır. Ajanın ortamla etkileşime girip kararlar alması için ortamı keşfetmesi gerekmektedir. Ortamdaki durumlar ve durumlara göre alınan ödül değerleri üzerinden karar vermesi ve optimum politikayı oluşturması için ortamda ajanın gezmesi gerekmektedir. Her durum ve eylem çifti için sıfır değeri tahminiyle başladığımızı varsayalım. Pozitif ödül veren bir eylemde bulunduğumuz anda aç gözlü politikayı izleyen bir algoritma, bu eylemi hiç bozmadan uygulamaya devam edecektir. Uygulanan eylemin optimal olup olmadığı belirli olmadan pekiştirmeli öğrenme algoritmamız lokal minimum noktaya takılıp kalacaktır. Bu durumda yapay ajanın karar vermeden keşif ve sömürü konseptine uyması gerekmektedir. Ajanın bulduğu optimal politikadan daha alt optimal politikalar bulunması olasıdır. Ajan ortam içerisinde başlangıçta rastsal değerle gezinerek arama işlemi yapmalı ve ortamı iyice keşfetmelidir. Ortam hakkındaki neredeyse bütün dinamiklere erişip, değerlere erişim sağladıktan ve optimal kararlar almaya başladıktan sonra algoritmamızın keşif yaklaşımını bırakıp sömürü yaklaşımına yönelmesi, yani önceden edindiği bilgileri kullanması gerekmektedir. Hatırlanacağı üzere modelden bağımsız yaklaşımlar optimal politikayı model tabanlı algoritma olan DP gibi garanti etmemektedir. Modelden bağımsız yaklaşımda optimal politikayı oluşturmak için ajanın ortam ile etkileşimde bulunması ve en optimal kararları alması beklenmektedir. Algoritmanın keşif aşamasında

rastgele davranışlar sergileyip ortam içerisinde stokastik bir biçimde gezmesi, kötü ve iyi olan tüm sonuçları toplaması gerekmektedir. Sömürü aşamasında algoritmamız en yüksek değerleri olan kararlar üzerinden harekete geçmelidir. Pekiştirmeli öğrenmede algoritmalarımızın keşif ve sömürü arasındaki dengeyi iyi kurması beklenmektedir. Statik bir ortam içerisinde algoritmamızın başlangıç anında maksimum keşif, öğrenme aşamasından sonra tecrübelerinden yola çıkarak uygun politikayı takip etmesi beklenmektedir. Keşif ve sömürü arasındaki dengeyi matematiksel olarak oluşturmak için $\varepsilon -$ greedy yaklaşımı kullanılmaktadır. Bu yaklaşımda ajan başlangıçta $\varepsilon \in [0,1]$ değeriyle olasılıksal bir aksiyon gerçekleştirerek maksimum düzeyde rastgele eylem uygulamakta böylece bulunduğu ortamı keşfetmektedir. Zaman ilerledikçe ε değeri üstel bir azalma göstererek, daha az keşif yöneliminde olup sömürü aşamasına geçmektedir. Böylece öğrenme gerçekleştikçe algoritmanın tecrübelerini kullanması sağlanmaktadır. (Kolter, J. Zico.)

1.1.4 Pekiştirmeli Öğrenme ve Kontrol Teorisi

Kontrol teorisinin temel problemi, fiziksel bir sistem için uygun kontrol işareti üreterek hedeflenen referans işaretini takip etmektir. Klasik bir kapalı döngü kontrol sisteminin blok şeması şekil 2.3'te görülmektedir.



Şekil 2.3: Genel Kontrol Yapısı.

Şekil 2.3'te genel kontrol yapısıyla fiziksel sistemleri kontrol etmek mümkündür. Kontrol teorisinde fiziksel sistemin matematiksel bir modeli varsa model tabanlı kontrol sistemi oluşturmak mümkündür. Fiziksel sistemin modelinin bilinmediği durumlarda, klasik kontrol teorisi yaklaşımında sistem tanılama teknikleri kullanılmaktadır. Fiziksel sistem üzerine etki eden bozucu girişler, gürültü ve modellenemeyen dinamikler oluşturulan kontrolcünün performansını etkilemektedir. Optimal kontrol teorisi bir önceki bölümde incelenerek belirli bir performans kriterini sağlayacak optimal kontrol olarak düşünülebilir. Optimal kontrol teorisinde fiziksel sistemin modelinin olması ve optimal kararlar verilmesi model tabanlı pekiştirmeli öğrenme konseptiyle benzerlik göstermektedir. Pekiştirmeli öğrenmenin optimal kontrol teorisine göre avantajı ödül fonksiyonunu özgürce yazabiliyor olmamızdır.

Optimal kontrol teorisinde, amaç fonksiyonu belirli bazı fonksiyonlardan oluşmaktadır ve çözüm bu fonksiyonlara göre oluşturulmaktadır. Kontrol teorisinde oluşturulan kontrol işareti $u(t)$ pekiştirmeli öğrenmede aksiyonları $a(t)$ oluşturmaktadır. Kontrol teorisinde tasarlanan kontrolcü, pekiştirmeli öğrenmedeki politikaya karşılık düşmektedir. Kontrol teorisinde oluşturulan kontrolcüler, fiziksel sisteme ve kontrol performansına göre değişmektedir. Optimal bir kontrolcü oluşturmak için literatürde, model ön görülü kontrol MPC-NMPC, lineer kuadratik düzeltici LQR vb. metotlar önerilmiştir. Kontrolcünün gürbüz veya adaptif olması gerektiği durumlarda Hinf, MRAC vb. kontrol metotları önerilmiştir. Endüstriyel birçok sistemde PID kontrolcü kullanılmaktadır. PID kontrolcünün avantajı, ampirik bir şekilde parametrelerinin ayarlanıyor olması ve fiziksel sistemin matematiksel modelinden bağımsız olmasıdır. Pekiştirmeli öğrenmede oluşturulan ortam/çevre fiziksel sistemin dinamik denklemlerine karşı düşmektedir ve genellikle lineer olmayan zamanla değişen stokastik diferansiyel denklem formatında ifade edilmektedir. Fiziksel dünyada birçok sistem model tabanlı kontrol algoritmalarıyla otomatik bir şekilde kontrol edilmektedir. Kontrol teorisi günümüzde, otonom araçlar, roket sistemleri, uçaklar, robotlar vb. birçok alanda endüstrinin ve sanayinin temelini oluşturmaktadır. Kontrol teorisi, matematiksel modeller belirli olduğunda analitik çözümler ürettiği için çok güçlüdür.

1.2 Rastgele Süreçler ve Markov Süreci

Pekiştirmeli öğrenme yaklaşımıyla, fiziksel sistemler ve durumları arasındaki geçişleri rastgele süreç olan Markov karar süreci ile modellemektedir. Mühendislik ve bilimsel uygulamalarda rastgele olaylar bir dizi biçiminde karşımıza çıkmaktadır. Olasılık uzayındaki her bir olaya belirli bir kurala göre bir fonksiyon atanmasıyla oluşturulan fonksiyon topluluğu rastgele süreç veya stokastik süreç olarak adlandırılmaktadır. Rastgele süreci ifade eden fonksiyon topluluğunda, basit (PDF) veya ortak olasılık yoğunluk fonksiyonları (JPDF), zamanla değişmiyorsa bu süreç durağan olarak isimlendirilmektedir. Bazı durumlarda rastgele süreci ifade eden fonksiyonlar aynı istatistiksel özellikleri sergilemektedir. Bu durum rastgele sürecin ergodiklik özelliğini taşıması anlamındadır. Bu gibi özel durumlarda sürece ilişkin gerçeklemelerde rastgele süreci ifade eden tek bir örnek fonksiyonun bilinmesiyle tüm sürecin istatistiksel özellikleri modellenebilmektedir. 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 sürekli zamanlı süreç olarak dört sınıfta incelenmektedir. Bu tez kapsamında ayrık değerli ayrık zamanlı rastgele süreç olan Markov süreci ve Markov sürecinin genişletilmiş durumları incelenecektir.

1.2.1 Markov Ödül Süreci

Pekiştirmeli öğrenme yaklaşımında, yapay ajanın içinde bulunduğu ortamın matematiksel modeli bilinmediği varsayılmaktadır. Yapay ajanın bulunduğu ortam ve durumlar arasındaki geçiş Markov sürecinin genişletilmiş hali olan Markov ödül süreçleriyle modellenmektedir.

Markov ödül süreciyle modellenen ortamda, yapay ajanın her bir durum içinde bulunmasıyla ödüller hesaplanmış olur. Markov ödül süreci ayrık durumlu ayrık değerli olduğu için sonlu sayıda, sayılabilir durum kümesine sahiptir. Çok büyük boyutlu MRP için, iteratif çözüm metotları, dinamik programlama, Monte Carlo metodu ve zamansal fark metodu çözüm için kullanılmaktadır.

1.2.2 Markov Karar Süreci

Markov karar süreci, Markov ödül sürecinin $a \in A$ aksiyon kümesiyle genişletişmiş halidir ve matematiksel olarak, S Markov özelliği sağlayan sonlu ve ayrık durum kümesi, A aksiyon kümesi, Pss' $a = P[St+1 = s' | St = s, At = a]$ koşullu geçiş olasılığı, $Rs\ a = E[Rt+1|St = s, At = a]$ ödül fonksiyonu ve $\gamma \in [0,1]$ azaltma faktörüyle tanımlanmaktadır. Markov karar sürecinde tanımlanan iki önemli yaklaşım, politika ve değer fonksiyonudur. MDP'de politika ajanın içinde bulunduğu st durumuna bağlıdır, gelecekteki veya geçmişteki durumlara bağlı değildir. Politika zamandan bağımsızdır. Yapay ajan Markov karar süreci ile modellenen çevre içerisinde herhangi bir s durumundan başlayıp belirli bir politikayı π takip ederek alacağı ödülleri ifade etmektedir. Markov karar sürecinde ajanın ortam içerisinde aksiyon ve ödüller toplaması sonucunda Markov karar sürecinin iki farklı çözüm yaklaşımı oluşmaktadır. Durum-değer ve aksiyon-değer fonksiyonları.
Markov karar süreci ile modellenen çevre içerisinde bu belirli politikayı takip ederek, $St = s\ At = a$ uygulaması sonucunda almış olduğu değerleri ifade eder.

1.3 Markov Karar Süreci Çözüm

Markov karar sürecini $M = (S, A, P, R, \gamma)$ parametreleriyle matematiksel olarak ifade edilebilir. Burada S durumları, A aksiyonları, P durum geçiş olasılığı matrisini, R ödül fonksiyonunu ve son olarak γ azaltma faktörünü belirtmektedir. Markov karar sürecinde P, R bilinmiyor olabilir veya durumlar sürekli zamanlı olup hesaplanamayabilir. MDP çözümü için denklem 4.34 ve 4.32 tanımlanmıştı, deterministik bir politika π: $S \rightarrow A$ durumları aksiyonlara haritalamaktadır. Bu durumda yapay ajanın bir durumda nasıl bir karar alacağı ve uygulayacağı matematiksel olarak tanımlıdır ve deterministiktir. Stokastik bir politika π: $S\boldsymbol{x}A \rightarrow [0,1]$ ile tanımlanmakta olup karar vericinin belirli bir durumda uyguladığı kararları belirli bir olasılık değerine haritalamaktadır. MDP'nin çözümü için yapay ajanın belirli bir politikayı izleyerek durum-değer yaklaşımının optimum değerini ve aksiyon-değer yaklaşımının optimum değerini hesaplaması gerekmektedir. Değer fonksiyonunun Bellman denklemi ile genişletilip, ajanın belirli bir durumda, bir politika π takip ederek oluşturacağı değer fonksiyonu denklem 2.35'te görülmektedir.

$$V\pi(s) = R(s) + \lambda \sum P(s'\, s \in S\, |s, \pi(s))\, V\pi(s') \tag{2.35}$$

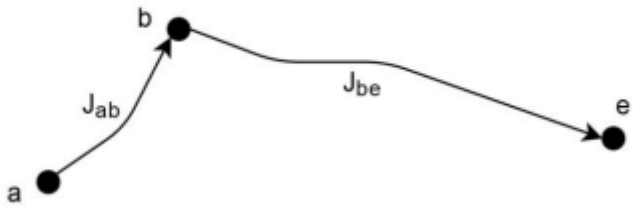
Optimal durum değer fonksiyonu, optimal politika $\pi*$ olarak belirlenir ve denklem 2.36'daki gibi hesaplanır.

$$* (s) = R(s) + \gamma \max a \in A \sum P(s' |s, a) V^* (s') \quad (2.36)$$

Markov karar sürecinin uygun politika ile değer tabanlı çözümü denklem 4.35 ile ifade edilmektedir. Markov karar sürecinin değer tabanlı çözümü denklem 4.36'da ifade edilmektedir. Denklem 2.36 Markov karar sürecindeki durumların değerleri üzerinden bir çözüm oluşturmaktadır. Dikkat edilecek olursa gelecekteki indirimli ödüllerin toplamını hesaplarken aksiyonların maksimum olduğu değeri işleme koymaktadır. Değer tabanlı yaklaşımda kullanılan politika aç gözlü yaklaşım ($\varepsilon - greedy$) olmaktadır. Denklem 2.35'te politika tabanlı yaklaşımda her bir durumda belirli bir politikayı takip ederek çözüm oluşturulur. Markov karar sürecinde R, P bilinmediği durumlarda yaklaşık çözümler oluşturulmaktadır. Denklem 2.35 ve 2.36 Markov karar sürecinin, değer tabanlı çözümlerini oluşturmaktadır. Pekiştirmeli öğrenmenin model tabanlı yaklaşımında diğer bir çözüm metodu politika tabanlı çözümlerdir. Bu algoritmalar ajanın bulunduğu s durumlarının değerleriyle ilgilenmek yerine bir davranışı π bulmayı hedeflemektedir. 2.35 ve 2.36 yaklaşımları değer tabanlı yaklaşımlar olup model tabanlı pekiştirmeli öğrenme için önerilen Markov karar sürecinin çözümlerini oluşturmaktadır. Pekiştirmeli öğrenmede $P(s' |st , at) \forall st \in S$ için durum geçiş matrisi bilinmemektedir. Böylece Markov karar süreci değer tabanlı çözüm algoritmaları iteratif olarak çözülmektedir. Çözüm için zamansal fark ve Monte Carlo metodu kullanılmaktadır. Bu tez çalışmasında zamansal fark yaklaşımıyla durum değer tabanlı ve aksiyon değer tabanlı Markov karar süreci çözümleri yaklaşık olarak hesaplanacaktır. Bu yaklaşımlar değer tabanlı çözüm algoritması olan Q-öğrenme ve Sarsa algoritmalarının temelini oluşturmaktadır. (Silver, David.)

1.4 Dinamik Programlama

Sıralı karar verme problemleri ve optimizasyon problemlerinin iteratif olarak alt parçalara bölünmesi durumunda dinamik programlama kullanılmaktadır. Büyük ve karmaşık problemleri daha küçük ve alt parçalara bölerek, alt parçaların çözümü oluşturulur ve bu çözümler bellekte depolanarak aynı çözüm tekrardan hesaplanmaz. Hesaplanan alt çözümler ile optimal çözüm oluşturulur. Dinamik programlama, diğer matematiksel programlama teknikleri gibi genel bir metoda sahip değildir, her problem için farklı bir uygulaması mevcuttur. Dinamik programlamanın altındaki temel kavram optimallik prensibidir. Dinamik programlama, sıralı karar problemleri ve optimizasyon problemlerini çözmektedir. İkinci bölümde anlatılan optimal kontrol problemi dinamik programlamayla iteratif olarak alt problemlere ayrıştırılarak çözülebilir. Şekil 2.5'te çoklu karar verme ve optimal yol çıkarma problemi görülmektedir.



Şekil 2.5: Optimal Yol Problemi.

Şekil 2.5'te $a \rightarrow b$ gitmenin maaliyeti Jab olarak, $b \rightarrow e$ gitmenin maaliyeti Jbe olarak ifade edilirse, optimal maliyet denklem 2.37'deki gibi olacaktır.

$$Jae* = Jab + Jbe \tag{2.37}$$

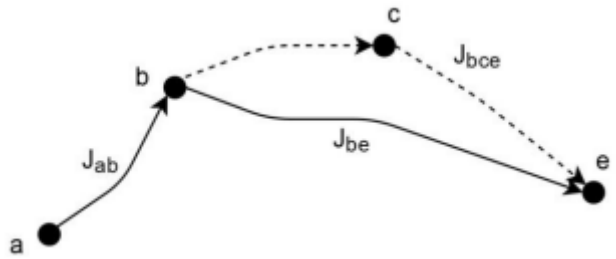
Eğer $a \rightarrow e$ optimal maliyetimiz $Jae*$ ise $b \rightarrow e$ de optimal olmak zorundadır. Eğer $b \rightarrow c \rightarrow e$ yolu $b \rightarrow e$ yolundan daha optimal ise bir ikilem oluşmaktadır. Matematiksel olarak denklem 2.38 ile ifade edilir.

$$Jbce < Jbe \tag{2.38}$$

Denklem 2.38 denklem 2.37 yerine yazılarak denklem 2.39 elde edilir.

$$Jab + Jbce < Jab + Jbe = Jae* \tag{2.39}$$

Denklem 2.39'dan görüleceği üzere optimallik prensibine aykırı bir durum oluşturmaktadır. Eğer $Jae*$, $a \rightarrow e$ optimal bir yol ise daha optimal bir alt yol oluşturulamaz. Bu durum şekil 2.6'da görülmektedir



Şekil 2.6: Optimal Yol Problemi Çelişkisi

Optimallik prensibi ayrık zamanlı sistemler için, $\pi* = \{\pi 0*, \pi 1*, \ldots, \pi N-1*\}$ optimal kararlar dizi olmalıdır. St durumları erişilebilir olduğu varsayımıyla yani matematiksel bir modelin oluşturulmasıyla, St durumundan t'ye bağlı olarak alt problemlerin geçiş maliyetleri minimum yapılmalıdır. Böylece optimallik prensibi alt optimal problemlere bölünmektedir. Dinamik programlama oluşturulan alt optimal problemlerin çözümü için ileri yönlü ve geriye doğru olarak iki farklı biçimde

uygulanmaktadır. Dinamik programlama durumlar arasındaki geçişler deterministik ise deterministik dinamik programlama, durumlar arası geçişler rastsal ise stokastik dinamik programlama olarak isimlendirilmektedir. Dinamik programlama sıralı karar verme problemlerinde kullanılmakta olup model tabanlı pekiştirmeli öğrenme problemlerinde kullanılmaktadır. Dinamik programlama ile Markov karar sürecinin çözülmesi için durum geçiş matrisinin bilinmesi gerekmektedir. (Sutton , Richard)

1.5 Bellman Denklemi

Modelden bağımsız pekiştirmeli öğrenmede ajanın içinde bulunduğu ortamı Markov karar süreciyle modellenir. Amacımız Markov karar sürecinin çözümü olan optimal politika veya optimal değer fonksiyonlarını hesaplamaktır. Optimal değer fonksiyonu $V*(s)$ olarak denklem 2.25 ile ifade edilmektedir. Yapay ajanın optimal değer ifadesini hesaplaması için her bir durumun değerlerini oluşturması gerekmektedir. Bu durum denklem 2.40'taki gibidir.

$$V(s) = E[Gt \mid St = s] \tag{2.40}$$

Denklem 2.40'ta $Gt = Rt+1 + \gamma Rt+2+\ldots = \sum \gamma\ kRt+k+1 \infty\ k=0$ ifade etmekte olup gelecekteki ödül değerlerini ifade etmektedir. Denklem 2.40'ın hesaplamasında iteratif bir yaklaşım oluşturan Bellman denklemi, denklem 2.41'deki gibidir.

$$\begin{aligned} V(s) &= E[Gt \mid St = s] \\ &= E[Rt+1 + \gamma Rt+2 + \gamma\ 2Rt+3+\ldots \mid St = s] \\ &= E[Rt+1 + \gamma(Rt+2 + \gamma Rt+3+\ldots)\mid St = s] \\ &= E[Rt+1 + \gamma Gt+1 \mid St = s] \\ &= E[Rt+1 + \gamma V(St+1) \mid St = s] \end{aligned} \tag{2.41}$$

Denklem 2.41 ile durum değer fonksiyonu ifade edilmektedir. Denklem 4.41'in beklenen değer ifadesi $E[]$bir sonraki durumlarla ifade edilmektedir. $E[Rt+1] \rightarrow Rt+1$ olmaktadır böylece beklenen değer sadece denklemin sağ tarafındaki$E[\gamma V(St+1)]$ terim ile hesaplanmaktadır. Beklenen değer yani ortalama hesabının denklem 2.41'e eklenerek iteratif olarak denklem 2.42 elde edilmektedir. Bu denklem Bellman denklemi olarak ifade edilmektedir

$$V(s) = Rs + \gamma \sum Pss\ 'V(s\ ) \tag{2.42}$$

Denklem 2.42 her bir durumun gelecekteki azaltılmış ödül değerleriyle birlikte durum-değer fonksiyonu yazılmaktadır. Bellman denklemi aksiyon değer fonksiyonuyla birlikte denklem 2.43'teki gibi ifade edilmektedir.

$$q(s, a) = Rs\ a + \gamma \sum Pss'\ a\ q(s\ ,\ a\ ) \tag{2.43}$$

Denklem 2.43 ve denklem 2.42 sayesinde Markov karar süreciyle modellenen ortam içerisinde değer tabanlı pekiştirmeli öğrenme yaklaşımı iteratif olarak hesaplanabilir. Modelden bağımsız pekiştirmeli öğrenme yaklaşımında Pss' , $Pss'\ a$ durum geçiş olasılıkları matrisi bilinmemektedir. Yapay ajanın bu durumda Markov karar sürecinin çözümünü oluşturamaz. Zamansal fark veya Monte Carlo yaklaşımıyla denklem 2.43 ve denklem 2.42 yaklaşık olarak hesaplanabilir. Bu tez çalışmasında zamansal fark algoritmalarıyla denklem 2.42 ve denklem 2.43'ün iteratif çözümleri incelenecektir. (Silver, David.2015)

1.6 Pekiştirmeli Öğrenme Politika Tabanlı Çözüm

Pekiştirmeli öğrenmenin politika tabanlı çözümü, MDP'ye değer tabanlı yaklaşıma alternatif olarak durumların değerlerini oluşturmak yerine bir politika bulmayı hedeflemektedir. Bir önceki bölümde MDP'nin çözümü için durum-değer ve aksiyon-değer fonksiyonları incelenmişti. Bu yaklaşımlarda yapay ajanın ortam içerisinde almış olduğu ödüllerle $V\pi(s)$, $q\pi(s,\ a)$ fonksiyonları güncellendi. Güncelleme işlemi için ajanın oluşturacağı bir politika incelenmedi. Örneğin durum değerlerini maksimum yapan bir politika seçimi yapıldı. Deterministik bir politika için bu durum denklem 4.30'de ifade edilmektedir. Pekiştirmeli öğrenmenin çözümü için önerilen politika tabanlı yaklaşım, yapay ajanın ortam içerisinde bir davranış geliştirmesini sağlamaktadır. Ajanın davranış biçimi deterministik veya stokastik olabilir. Deterministik bir politika denklem 2.44'te görülmektedir.

$$\pi_\theta(s) = \max_{a \in A} \theta(s, a) \tag{2.44}$$

Denklem 2.44'e benzer şekilde politikanın stokastik olması durumunda denklem 2.45 ile ifade edilen politika oluşturulabilir.

$$\pi_\theta(a|s) = \frac{exp\,\theta\,(s,a)}{\sum_{a' \in A} exp\,\theta\,(s,a')} \tag{2.45}$$

Denklem 2.44 ve 2.45 ile oluşturulan politika, Markov karar sürecinin çözümü için yazılan denklem 2.30 ile ifade edilen politikadan farklıdır. Dikkat edilmelidir ki θ burada keyfi bir parametredir. Bir

politikanın değeri denklem 2.22 ile azaltılmış gelecek ödüllerin beklenen değerini ifade etmektedir. Bu durum denklem 2.46'da görülmektedir.

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] \tag{2.46}$$

Pekiştirmeli öğrenmenin politika tabanlı çözümü için oluşturulan optimizasyon problemi denklem 2.46 genişletilerek denklem 2.47'deki gibi ifade edilmektedir.

$$V_\theta(s) = E[\sum_{t=1}^{\infty} \gamma^t r_t | s_{t+1} \sim P(s'|s_t, \pi_\theta(s_t)), s_1 = s] \tag{2.47}$$

Denklem 2.47'den ifade edildiği gibi politika uzun bir ufuk boyunca hesaplanabilmektedir. Denklem 2.47 ile ifade edilen politika değerinin analitik olarak gradyanı $\nabla\theta V\theta(s)$ bilinmemektedir. Optimizasyon probleminin çözülmesi için $\nabla\theta V\theta(s)$ ifadesinin yaklaşık çözümü gerekmektedir. Basit bir politika arama algoritması tablo 2.1'deki gibidir.

Tablo 2.1: Basit Politika Arama Algoritması

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.*

Tablo 2.1 ile ifade edilen basit politika arama algoritmasının dışında diğer bir politika arama algoritması politika gradyanı ve takviye (reinforce) metodudur. Bu metot birçok politika tabanlı pekiştirmeli öğrenme algoritmasının temelini oluşturmaktadır.τ = ($s1$, $a1$, $s2$, $a2$, . . . , sT, aT) durum aksiyon çifti olarak izlediğimiz yörüngeyi ifade etmek üzere politikanın değer fonksiyonu denklem 2.48'deki gibi ifade edilebilir.

$$V_\theta(s) = E[R(\tau); \theta] = \int p(\tau; \theta)\, R(\tau) d\tau \tag{2.48}$$

Takviye algoritmasının oluşturulması için denklem 2.48'in düzenlenmesi gerekmektedir. Bu düzenleme öncelikle denklem 2.48'in gradyan hesabı üzerinden denklem 2.49'daki gibi ifade edilmektedir.

$$\nabla_\theta V_\theta(s) = \nabla_\theta \int p(\tau;\theta)\, R(\tau) d\tau$$
$$= \int \nabla_\theta p(\tau;\theta)\, R(\tau) d\tau$$
$$= \int \frac{p(\tau;\theta)}{p(\tau;\theta)} \nabla_\theta p(\tau;\theta)\, R(\tau) d\tau$$
$$= \int p(\tau;\theta) \nabla_\theta \log p\,(\tau;\theta)\, R(\tau) d\tau$$
$$= E[\nabla_\theta \log p\,(\tau;\theta) R(\tau)] \quad (2.49)$$

Denklem 2.49 ile $V\theta(s)$ fonksiyonunun gradyanı $\nabla\theta V\theta(s)$ hesaplanabilir. Denklem 2.49'a dikkat edilmelidir ki bir beklenen değer hesabı yapılmakta böylece örneklem üzerinden bir tahmin oluşturulmaktadır. Politika arama için önerilen politika gradyanı ve takviye algoritmasının ikinci düzenlemesi denklem 2.49 üzerinden yapılmaktadır. Bu durum denklem 2.50'de ifade edilmektedir.

$$\nabla_\theta \log p\,(\tau;\theta) = \nabla_\theta \log\Big(\prod_{t=1}^{T} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)\Big)$$
$$= \nabla_\theta \sum_{t=1}^{T} (\log p\,(s_{t+1}|s_t, a_t)) + \nabla_\theta \log \pi_\theta\,(a_t|s_t)$$
$$= \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\,(a_t|s_t) \quad (2.50)$$

Denklem 2.50 kullanılarak politika optimizasyonu algoritması tablo 2.2'de görülmektedir. Denklem 2.50'de gradyan geçiş olasılıklarına bağlı olmadığı için, eşitliğin sağındaki ilk kısım sıfır olmaktadır.

Tablo 2.2: Politika Gradyanı ve Takviye Algoritması

ALGORİTMA : Politika Gradyanı ve Takviye Algoritması		
1	:	*Stokastik bir politika* π_θ *için M adımda* τ *yörüngeyi oluşturun.*
2	:	*Gradyan değerini yaklaşık olarak hesaplayın.* $g_\theta \leftarrow \frac{1}{M}(\sum_{i=1}^{M}(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\,(a_t^{(i)}|s_t^{(i)}))\, R(\tau^{(i)}))$
3	:	*Parametreleri güncelle.* $\theta \leftarrow \theta + \alpha g_\theta$

Tablo 2.2'de önerilen algoritma, gradyanın örneklem üzerinden hesaplanması olup gerçek gradyan değerini hesaplamamaktadır. Dikkat edilmelidir ki ajanın ortam içindeki örnekleminin çok iyi olup politikayı en iyi şekilde belirlemesi gerekmektedir. Modelden bağımsız pekiştirmeli öğrenme

yaklaşımında ortam hakkında bir bilgi oluşmadığı için ajanın ortamı keşfetmesi ve sonrasında keşiflerinden edindiği tecrübeleri kullanması gerekmektedir. Değer tabanlı veya politika tabanlı algoritmalar ortamı bir süre gezip bir bilgi toplaması gerekmektedir. Bir önceki bölümde ifade edilen $\varepsilon - greedy$ yaklaşımı denklem 2.51 ile pekiştirmeli öğrenme algoritmaları için kullanılmaktadır. (Kolter, J. Zico)

$$\pi(s) = \begin{cases} \max_{a \in A} \hat{Q}(s,a) & 1-\varepsilon \\ \text{rastgele aksiyon} & \text{diğer durumlar} \end{cases} \tag{2.51}$$

1.7 Pekiştirmeli Öğrenme Değer Tabanlı Çözüm

Markov karar süreci, pekiştirmeli öğrenme için matematiksel bir altyapı sağlamaktadır. Markov karar süreci bir önceki bölümde detaylıca incelenmiştir. Markov karar sürecinin çözümü optimal durum-değer veya aksiyon-değer fonksiyonlarının çözülmesiyle elde edilmektedir. Bu durum denklem 2.52'de ifade edilmektedir.

$$\begin{aligned} V_*(s) &= \max_a \; R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s') \\ q_*(s,a) &= R_s^a + \gamma \textstyle\sum_{s' \in S} P_{ss'}^a \max_{a'} \; q_*(s', a') \end{aligned} \tag{2.52}$$

Denklem 2.52'deki $V*(s)$ durum-değer fonksiyonunun optimal çözümünü ifade etmekte olup MDP'nin çözümünü oluşturmaktadır. Benzer şekilde $q*(s, a)$ aksiyondeğer fonksiyonunun optimal bir çözümü olup MDP'nin çözümüdür. Denklem 2.52 ile ifade edilen yaklaşımlarda yapay ajan MDP ile modellenen ortam içerisinde nasıl bir ödül değeri alacağını ve bu ödül değerlerine göre fonksiyonları nasıl güncelleyeceği ifade edilmektedir. Bu çözüm yaklaşımları MDP'nin değer tabanlı çözümlerini oluşturmaktadır. Denklem 2.52'de ifade edilen $Pss'\ a$, $Rs\ a$bilinmemektedir. Durum geçiş matrisi ve ödül fonksiyonu bilinmediği için MDP'nin kapalı formda çözümü hesaplanamamaktadır. Yapay ajan bulunduğu ortam içerisinde gözlemlerde bulunarak st+1 durumunu $P(s'|st, at)$ dağılım fonksiyonundan örneklemiş olmaktadır. Elimizde var olan bu gözlem değerleri ve rt ortam içerisinde durumlar üzerinden yazılan ödül fonksiyonu ile yaklaşık hesaplama yapılmaktadır. Bu durum denklem 2.53'te ifade edilmektedir.

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

(2.53)

Denklem 2.53 st durumunun değerini t anındaki ödül ve bir sonraki st+1 durumu ve bir azaltma faktörüyle $rt + \gamma V^{\wedge} \pi$ (st+1)'deki gibi güncellemektedir. Bu durumdaki güncelleme bir sonraki durum değerinin pozitif veya negatif olmasına karşılık t anındaki durumu etkilemektedir. Denklem 2.53'e güncelleme yaparak daha etkili bir güncelleme kuralı olan denklem 2.54 elde edilmektedir.

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

(2.54)

Denklem 2.54'e dikkat edilecek olursa t anındaki $V^{\wedge} \pi$ (st) değerlerinin güncellemesi gene $V^{\wedge} \pi$ (st) 'ye ilave olarak zamansal fark kısmının eklenmesiyle oluşmaktadır. Zamansal fark yaklaşımında ajanın almış olacağı gelecekteki indirimli ödül değerine $\gamma V^{\wedge} \pi$ (st+1) bağlı olarak güncelleme gerçekleşecektir. $\gamma V^{\wedge} \pi$ (st+1) − $V^{\wedge} \pi$ (st) ifadesi zamansal fark olarak isimlendirilmekte olup gelecekteki $V^{\wedge} \pi$ (st+1) değeriyle, şimdiki $V^{\wedge} \pi$ (st) arasında bir denge kurmaktadır. Gelecekte tahmin edilen değer ile şimdiki değer arasındaki fark ne kadar iyi bir karar verdiğimizi ifade etmektedir. Böylece ajanın t anında elde etmiş olduğu ödül değerleri denklem 2.54 üzerinden denklem 2.53'e göre çok daha başarılı bir şekilde güncellenecektir. Denklem 2.53'teki güncelleme denklem 2.54'e göre çok daha başarısız olmasının sebebi içinde bulunduğumuz durum ödülü ve tahmin edilen gelecekteki ödül ile güncelleme yapıyor olmamızdır. Denklem 2.54'te ise içinde bulunduğumuz durum değeri güncellenirken, içinde bulunduğumuz durum değerine ilave olarak gelecekte ne kadar doğru kararlar verdiğimizi ifade eden zamansal fark ile güncellenmektedir. Denklem 2.54 modelden bağımsız pekiştirmeli öğrenme yaklaşımının zamansal fark yaklaşımıyla çözülmesini sağlamaktadır. Zamansal fark yaklaşımının aksiyon-değer ve durum-değer yaklaşımlarına uygulanmaktadır. Zamansal fark algoritması tablo 2.3'te ifade edilmektedir.

Tablo 2.3: Zamansal Fark Algoritması

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	:	$t \leftarrow t + 1$	

Zamansal fark algoritması MDP'yi oluşturmadan $V^{\wedge} \pi\ (st) \approx V\ \pi\ (st)$ yaklaşık olarak hesaplamamızı sağlar. (Kolter, J. Zico)

1.7.1 Q Öğrenme Algoritması

Q öğrenme algoritması zamansal fark algoritmasıyla beraber aksiyon-değer tabanlı bir algoritma olup kapalı politikadır. Hatırlanacağı üzere kapalı politika yaklaşımında ajanın takip ettiği politika değeri güncellenmemektedir. Q öğrenme algoritması değer tabanlı yaklaşıma benzerlik göstermektedir. Değer tabanlı yaklaşıma alternatif olarak Q öğrenme algoritmasında aksiyon ve durumlar üzerinden denklem 4.55'teki gibi ifade edilmektedir.

$$Q\pi\ (s, a) = R(s) + \gamma\ \Sigma\ P(s'|s, a) Q\pi\ s' \in S\ (s', \pi(s')) \tag{4.55}$$

Değer tabanlı çözüm algoritması olan Q öğrenme algoritması denklem 4.55'te görüldüğü gibidir. Denklemde ajanın takip ettiği politika $\pi(s')$ ile ifade edilmektedir. Denklem 4.55'in optimal çözümü denklem 4.56'daki gibidir.

$$Q*(s, a) = R(s) + \gamma\ \Sigma\ P(s'|s, a)\ \max\ a' \in A\ Q*\ s' \in S\ (s', a') \tag{4.56}$$

Denklem 4.56'da ajanın sahip olduğu politika $\pi(s')$ Q değerleri üzerinden maksimum değeri üreten aksiyon seçimi olarak oluşturulabilir. Denklem 4.56 Q öğrenme algoritmasının temelini oluşturmaktadır. Böylece ajan MDP içerisinde durum ve aksiyon değer fonksiyonuyla ifade edilmektedir. Denklem 4.56'da $P(s'|s, a)$, $R(s)$ bilinmediği durumda, yani MDP tam olarak bilinmiyor sadece $t$anındaki ölçümler elimizde varsa, MDP'nin çözümü olan optimal durum aksiyon - değer fonksiyonunun $Q*(s, a)$ bulunması için denklem 4.56'nın zamansal fark yaklaşımıyla güncellenmesi gerekmektedir. Bu durum denklem 4.57'de ifade edilmektedir.

$$\hat{Q}^*(s,a) = (1-\alpha)\hat{Q}^*(s,a) + \alpha(r + \gamma \max_{a' \in A} \hat{Q}^*(s',a'))$$
$$\hat{Q}^*(s,a) = \hat{Q}^*(s,a) + \alpha(r + \underbrace{\gamma \max_{a' \in A} \hat{Q}^*(s',a') - \hat{Q}^*(s,a)}_{ZamansalFark}) \quad (4.57)$$

Denklem 4.57 ile oluşturulan Q öğrenme algoritması, ajanın bulunduğu ortam içerisinde yeterince durum ve aksiyonları uygulamasıyla gerçek değerlerine $Q*(s, a) \approx Q\hat{} *(s, a)$ yakınsamaktadır. Q öğrenmenin bir avantajı MDP elimizde bulunmadan aksiyon seçimleri gerçekleştirerek optimal politika değerlerini öğrenebiliyor olmamızdır. Bu denklem 4.58'de ifade edilmiştir.

$$\pi^*(s) = \arg \max_{a \in A} \hat{Q}^*(s,a) \quad (4.58)$$

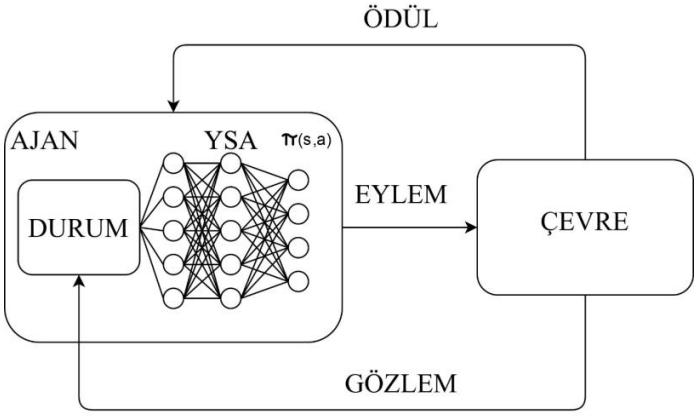
Denklem 4.57 ile ifade edilen Q öğrenme algoritması tablo 4.4'te görülmektedir.

Tablo 2.4: Q Öğrenme Algoritması

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	:	$s \leftarrow s'$

1.8 Derin Pekiştirmeli Q Öğrenme Algoritması

Bir önceki bölümde ifade edilen Q öğrenme ve Sarsa öğrenme algoritmaları literatürde tablo metodu olarak isimlendirilmektedir. Bu metotlar ayrık durum ve aksiyon uzayı için uygulanabilirdir. Gerçek hayatta kontrol probleminin, durum uzayı ve kontrol işareti süreli zamanlı olarak ifade edilmektedir. VizDoom oyun panelinde, kontrol işaretimiz olan tork değerleri $\tau \in \Re N$, N serbestlik dereceli sistem için reel sayılar uzayında tanımlı olup sonsuz değer almakta ve robotun eklem değişkenleri olan N serbestlik dereceli sistem için reel sayılar uzayından yani sonsuz değerler almaktadır. Pekiştirmeli öğrenmede pratik bir uygulama olarak sürekli durum ve aksiyon uzaylarına ayrıklaştırma yaparak Q öğrenme ve Sarsa öğrenme algoritmaları uygulanabilir, fakat pratikte ayrıklaştırmadan dolayı oluşacak hatalar kontrol uygulamalarında sorun teşkil etmektedir. Pekiştirmeli öğrenmenin tablo metodunun bir diğer problemi hesaplama maliyetini oluşturmaktır. Ayrık durum ve aksiyon uzayımızın boyutunun artması hesapsal karmaşıklığa ve hesaplama maliyetinin artmasına sebep olmaktadır. Makine öğrenmesinde kullanılan derin yapay sinir ağları evrensel fonksiyon yaklaşımlarıdır. Yapay sinir ağlarının fonksiyon yakınlaşıcı özelliği ile pekiştirmeli öğrenmenin Q öğrenme algoritması birleştirilerek derin pekiştirmeli öğrenme yöntemleri geliştirilmiştir. Derin pekiştirmeli öğrenme algoritması, derin yapay sinir ağları ve Q öğrenme algoritmasının birleştirilmesiyle elde edilmektedir. Derin pekiştirmeli öğrenme algoritmasıyla sürekli durumların bulunduğu kontrol problemleri çözülmektedir. Derin pekiştirmeli öğrenme algoritmasının dezavantajı kontrol probleminde kontrol işaretimiz $u(t)$, zamanın sürekli bir fonksiyonudur. Kontrol işaretimizin pekiştirmeli öğrenmedeki karşılığı olan aksiyonlarımız $a(t)$ zamanın birer sürekli fonksiyonu olarak kontrol problemlerimizde karşımıza çıkmaktadır. Aksiyonlarımızın ayrık değerler alması sebebiyle, pekiştirmeli öğrenme ile kontrol problemlerini çözmek için önerilen derin Q öğrenme algoritması yeterli sonuç üretememektedir. Derin Q öğrenme algoritmasının blok diyagramı şekil 4.9'da görülmektedir.



Şekil 2.4: Derin Q Öğrenme Algoritması

1.9 Derin Deterministik Politika Gradyanı Algoritması

Derin deterministik politika gradyanı algoritması, deterministik politika gradyanı algoritması ve derin Q öğrenme algoritmasının birleştirilmesiyle oluşturulmuştur. Derin Q öğrenme algoritması sürekli durum ve ayrık aksiyon kümesi üzerinde çalışmaktadır. Birçok kontrol problemi sürekli durum ve sürekli aksiyon uzayına sahiptir. Derin Q öğrenme algoritması kapalı politika bir algoritma olup ε – greedy yaklaşımıyla aksiyonları seçmektedir. Sürekli aksiyon uzayında bu yaklaşım uygulanmamaktadır. Sürekli aksiyon uzayına derin Q öğrenme algoritmasını uygulamak için aksiyon uzayının ayrıklaştırılması gerekmektedir. Fakat aksiyon uzayını ayrıklaştırmak birçok problem oluşturur. Örneğin 7DOF bir robot kol için, her bir eklem uygun aksiyon değerleri $ai \in \{-k, 0, k\}$ olsun böylece aksiyon uzayımızdaki eleman sayısı 37 = 2187 olmaktadır. Aksiyon uzayındaki üç elemanın çözünürlüğünün yetmediği durumlarda daha fazla aksiyon eklemek gerekmektedir ve bu durumda 7DOF bir sistem için üstel olarak büyümektedir.

2. VizDOOM OYUNU ÜZERİNDEN PEKİŞTİRMELİ ÖĞRENME

Resim 3.1: Doom Oyunu

2.1 VizDoom Oyununa Giriş

Bu bölümde, tezin odak noktası olan VizDoom oyununun genel yapısı ve temel mekanikleri tanıtılmaktadır. VizDoom, Doom adlı popüler birinci şahıs nişancı oyununun yapay zekâ araştırmaları için özelleştirilmiş bir sürümüdür. Oyun, ajanların düşmanları yenmeleri ve belirli hedefleri tamamlamaları gereken bir senaryo sunmaktadır. VizDoom, yapay zekâ algoritmalarını test etmek ve eğitmek için ideal bir ortam sağlamaktadır.

2.2 Oyunun Temel Yapısı

VizDoom, popüler bir video oyunu olan Doom'un yapay zeka araştırmalarında kullanılan bir simülatörüdür. Oyun, birinci şahıs bakış açısıyla kontrol edilen bir karakterin, düşmanları alt etmek, görevleri tamamlamak ve bölümleri keşfetmek gibi hedefleri olan bir aksiyon-macera oyunudur. Oyunun temel yapısı, aşağıdaki unsurlardan oluşur:

1. Oyun Motoru: VizDoom, orijinal Doom oyununun motorunu temel alır ve bir yapay zeka simülatörü olarak kullanılır. Oyun motoru, grafik işleme, fizik simülasyonu, yapay zeka düşmanların davranışları ve oyun mekaniği gibi temel oyun unsurlarını içerir. Bu motor, araştırmacıların oyunu kontrol etmek, çevreyi gözlemlemek ve etkileşimde bulunmak için gerekli işlevleri sağlar.

2. Haritalar ve Senaryolar: VizDoom, çeşitli haritalar ve senaryolar sunar. Haritalar, oyuncunun keşfedebileceği oyun dünyasını temsil eder ve farklı mekanikler, engeller ve düşmanlar içerir. Senaryolar ise belirli hedeflere sahip görevleri ifade eder. Bu senaryolar, oyuncunun belli bir sürede belirli düşmanları alt etmesi, nesneleri toplaması veya hedefleri tamamlaması gibi belirli görevleri yerine getirmesini gerektirebilir.

3. Karakterler ve Düşmanlar: Oyunda kontrol edilen karakter, oyuncunun gözünden deneyimi yaşayan ana karakterdir. Düşmanlar ise yapay zeka tarafından kontrol edilen düşman karakterlerdir. Düşmanlar, belirli davranış kalıplarına sahip olabilir, oyuncunun saldırılarına tepki verebilir ve stratejik olarak hareket edebilir. Oyundaki düşmanlar, oyuncuya zorluk seviyeleri ve oyun deneyimi sunan önemli bir unsurdur.

2.3 Oyun Senaryoları ve Haritalar

VizDoom, çeşitli senaryoları ve haritaları içeren geniş bir oyun koleksiyonuna sahiptir. Her senaryo farklı zorluk seviyeleri, hedefler ve oyun dinamikleri sunarak araştırmacılara çeşitli deneyler yapma imkanı sağlar. Bu bölümde, VizDoom oyununda sunulan temel senaryolar ve haritalar hakkında genel bir bakış sunulacaktır.

2.3.1. Oyun Senaryoları

1. Deathmatch: Deathmatch senaryosu, oyuncuların birbirleriyle çatıştığı rekabetçi bir çok oyunculu moddur. Oyuncular, haritadaki diğer oyunculara karşı mücadele eder ve en yüksek skoru elde etmek için stratejilerini kullanır.

2. Takım Deathmatch: Takım Deathmatch senaryosu, oyuncuların takımlar halinde rekabet ettiği bir çok oyunculu moddur. Oyuncular, takımlarıyla birlikte hareket ederek diğer takımlara karşı savaşır ve zafer için takım çalışması ve koordinasyon stratejilerini kullanır.

3. Capture the Flag: Capture the Flag senaryosu, oyuncuların bayrağı düşürmeden rakip takımın bayrağını ele geçirmeye çalıştığı bir moddur. Oyuncular, düşman üssüne girip bayrağı almak ve kendi üslerine geri getirmek için taktiksel becerilerini kullanır.

4. Tek Oyunculu Görevler: VizDoom ayrıca tek oyunculu görevler sunar. Bu görevler, oyuncunun tek başına belirli hedefleri tamamlamak için ilerlemesini gerektirir. Örnek olarak, belirli bir düşmanı etkisiz hale getirmek, bir nesneyi bulmak veya bir labirentte çıkışı bulmak gibi görevler yer alır.

2.3.2. Haritalar

VizDoom'da çeşitli haritalar bulunur. Haritalar, farklı boyutlara, mimari özelliklere ve ortamlara sahiptir. Örneğin, dar koridorlardan oluşan bir harita, geniş açık alanlara sahip bir harita veya labirent şeklinde bir harita gibi farklı tiplerde haritalar mevcuttur. Bu haritalar, araştırmacıların farklı oyun senaryolarını ve çevreleri keşfetmelerini sağlar.

VizDoom'un geniş senaryo ve harita seçenekleri, araştırmacıların farklı oyun dinamikleri, zorluk seviyeleri ve stratejiler üzerinde çalışmalarını sağlar. Bu senaryolar ve haritalar, yapay zeka algoritmalarının farklı koşullar altında performansını değerlendirmek ve geliştirmek için değerli bir ortam sağlar.

2.4 Oyun Kontrolleri

VizDoom, oyuncuların oyun dünyasıyla etkileşime girmek ve karakterlerini kontrol etmek için çeşitli kontrol seçenekleri sunar. Oyun kontrolleri, oyuncunun karakterin hareketini, nişan alma yeteneklerini, saldırılarını ve diğer etkileşimleri gerçekleştirmesini sağlar. İşte VizDoom'da kullanılan temel oyun kontrolleri:

1. Hareket Kontrolleri: Oyuncu, karakterin oyun dünyasında hareket etmesini sağlamak için klavye veya oyun kumandasını kullanabilir. İleri, geri, sağa ve sola hareket etmek için ilgili tuşlar veya kontrol çubukları kullanılabilir. Ayrıca zıplama, eğilme ve koşma gibi özel hareketler de oyuncunun kontrolünde olan seçenekler arasındadır.

2. Nişan Alma ve Ateş Etme Kontrolleri: VizDoom'da oyuncunun doğru nişan alabilmesi ve düşmanlara saldırabilmesi için fare veya kontrol çubuğu kullanılabilir. Fare veya kontrol çubuğuyla nişan alınırken hassasiyet ayarları yapılabilir. Silah seçimi ve ateş etme işlemleri ise ilgili tuşlar veya düğmeler kullanılarak gerçekleştirilebilir.

3. Etkileşim Kontrolleri: VizDoom'da oyuncunun çevreyle etkileşimde bulunabilmesi için çeşitli etkileşim kontrolleri bulunur. Bunlar arasında kapıları açma/kapama, anahtarları toplama, nesneleri kullanma gibi işlemler yer alır. Bu etkileşimler genellikle ilgili tuşlar veya düğmelerle gerçekleştirilir.

4. Kamera Kontrolü: Oyuncunun karakterin bakış açısını kontrol etmesi için kamera kontrolü sunulur. Fare veya kontrol çubuğu kullanılarak kameranın yönü değiştirilebilir, yukarı ve aşağı hareket ettirilebilir.

2.5 VizDoom'un Yapay Zekâ Araştırmalarında Kullanımı

VizDoom, yapay zeka araştırmalarında önemli bir araç olarak kullanılmaktadır. Oyun, karmaşık ve gerçek dünya benzeri bir ortam sağlayarak yapay zeka algoritmalarının öğrenme ve karar verme yeteneklerini geliştirmek için ideal bir platform sunmaktadır. İşte VizDoom'un yapay zeka araştırmalarında kullanımının bazı örnekleri:

1. Reinforcement Learning (Pekiştirmeli Öğrenme): VizDoom, pekiştirmeli öğrenme algoritmalarının geliştirilmesi ve değerlendirilmesi için yaygın olarak kullanılan bir platformdur. Oyunda yapay zeka ajanı, çevreyi algılar, durumlar arasında geçiş yapar ve ödül alma stratejileri geliştirir. Bu, ajanların oyun içindeki hedefleri yerine getirmek için stratejilerini optimize etmelerini sağlar.

2. Derin Öğrenme: VizDoom, derin öğrenme algoritmalarının uygulanmasında yaygın olarak kullanılan bir oyun ortamıdır. Derin sinir ağları, oyundan elde edilen görsel girdileri analiz ederek karakterin durumunu ve çevresini anlamaya çalışır. Bu, ajanların karmaşık görsel verileri işlemesini ve anlamlı kararlar almasını sağlar.

3. Nesne Algılama ve Tanıma: VizDoom, nesne algılama ve tanıma algoritmalarının geliştirilmesi için de kullanılmaktadır. Oyundaki nesneleri algılayarak ve tanıyarak, yapay zeka ajanları çevrelerindeki önemli nesneleri tanımlayabilir ve bu bilgileri stratejilerini optimize etmek için kullanabilir.

4. Duygu Analizi ve Karar Verme: VizDoom, yapay zekanın duygusal tepkilerini ve karar verme süreçlerini incelemek için de kullanılır. Ajanlar, çevresel koşullara ve olaylara tepki verirken duygusal durumlarını yönetebilir ve bu durumları karar verme süreçlerine dahil edebilir.
VizDoom'un yapay zeka araştırmalarında kullanımı, daha karmaşık ve gerçekçi oyun senaryolarının tasarlanması ve yapay zeka algoritmalarının gerçek dünya problemlerine uyarlanması için önemli bir adımdır. Araştırmacılar, VizDoom üzerinde deneyler yaparak yeni algoritmalar geliştirebilir, performanslarını değerlendirebilir ve yapay zekanın farklı becerilerini keşfedebilir.

2.6 Tezin Odak Noktası

Bu tezin odak noktası, VizDoom oyununda pekiştirmeli öğrenme algoritmalarının uygulanması ve performanslarının değerlendirilmesidir. Tez, VizDoom'un yapay zeka araştırmalarında

kullanılabilirliğini incelemekte ve bu oyun ortamında pekiştirmeli öğrenme yöntemlerinin etkinliğini göstermeyi amaçlamaktadır.

Tezin amacı, VizDoom oyununda bir yapay zeka ajanı eğiterek, çevresini algılamasını, hareket etmesini ve oyun içi hedefleri başarmasını sağlamaktır. Bu amaçla, pekiştirmeli öğrenme algoritmaları, özellikle Q-learning ve Deep Q-Networks (DQN) gibi derin öğrenme temelli yaklaşımlar kullanılacaktır. Bu algoritmalar, ajanın oyun ortamını keşfetmesini ve en iyi stratejileri geliştirmesini sağlamak için kullanılacaktır.

Tezin yöntemi, VizDoom oyununda deneyler yaparak, ajanın performansını değerlendirmeyi içerecektir. Farklı öğrenme hızları, keşif-exploit dengesi ve hiperparametre ayarlamaları gibi faktörlerin performansa etkisi incelenecektir. Ayrıca, farklı oyun senaryoları ve haritalar kullanılarak ajanın genelleyebilme yeteneği de test edilecektir.

Bununla birlikte, tezde ayrıca VizDoom'un yapay zeka araştırmalarında kullanımının genel bir değerlendirmesi de yapılacaktır. VizDoom'un avantajları ve sınırlamaları ele alınacak ve diğer oyun ortamlarıyla karşılaştırılacaktır. Ayrıca, VizDoom'un yapay zeka alanında gelecekteki potansiyel uygulamaları ve geliştirmeleri de tartışılacaktır.

Bu tez, VizDoom'un yapay zeka araştırmalarında nasıl kullanılabileceğini ve pekiştirmeli öğrenme algoritmalarının oyun performansına etkisini anlamamıza katkıda bulunacaktır. Ayrıca, VizDoom'un avantajlarını ve sınırlamalarını göstererek, bu oyun ortamının yapay zeka alanında daha fazla keşif için potansiyelini ortaya koyacaktır.

2.7 VizDoom Oyununda Pekiştirmeli öğrenme

Bu bölümde, VizDoom oyununda reinforcement learning kullanarak bir botun nasıl öğrenildiğini ve oyun performansının nasıl artırıldığını inceleyeceğiz. VizDoom, popüler bir birinci şahıs nişancı oyunudur ve karmaşık görsel ve oyun stratejileri gerektiren bir ortam sağlar. Reinforcement learning, VizDoom oyununda ajanların oyun içi ödülleri maksimize etmek ve stratejileri geliştirmek için kullanılan güçlü bir yöntemdir.

Bu bölümün amacı, VizDoom oyununun temelini anlamak, oyunun özelliklerini ve ajanın hedeflerini belirlemektir. Ayrıca, reinforcement learning algoritmalarının oyun performansını geliştirmedeki etkisini ve VizDoom oyununda kullanılan belirli algoritmalara odaklanacağız.

VizDoom oyunu, ajanların birinci şahıs bakış açısıyla karmaşık bir ortamda hareket etmesini gerektirir. Ajan, düşmanları yenmek, hedefleri tamamlamak ve oyun içi ödülleri maksimize etmek için stratejiler geliştirmelidir. Oyunun dinamik yapısı ve yüksek boyutlu görsel girdiler, reinforcement learning algoritmalarının uygulanmasını zorlaştırır.

Reinforcement learning, VizDoom oyununda ajanların öğrenme sürecini analiz etmek ve oyun performansını artırmak için kullanılır. Ajanlar, oyun içi ödüller ve cezalar alarak, çevreyle etkileşime

girer ve stratejilerini geliştirir. Bu süreçte, ajanın durum-aksiyon değerlerini tahmin etmek için Q-learning veya Deep Q-Networks (DQN) gibi algoritmalar kullanılır.
Bu bölüm, VizDoom oyununda reinforcement learning'in nasıl uygulandığını ve ajanların oyun performansını nasıl geliştirdiğini ayrıntılı bir şekilde inceleyecektir. Tezin odak noktası olan VizDoom ve reinforcement learning kombinasyonu, ilgi çekici sonuçlar ve gelecekteki araştırma yönlendirmeleri sunabilir.

2.8 Oyun Ortamının Tanımlanması

VizDoom, birinci şahıs nişancı türünde popüler bir video oyunudur ve reinforcement learning çalışmalarında sıkça kullanılan bir platformdur. Bu bölümde, VizDoom'un temel yapısı ve oyun ortamının tanımı detaylı bir şekilde incelenecektir.

2.8.1. Oyunun Temel Yapısı

VizDoom, yapay zeka araştırmalarında kullanılmak üzere özel olarak tasarlanmış bir oyun platformudur. Oyun, ajanların oyun içi ödüllerini maksimize etmek için stratejiler geliştirmesi gereken birinci şahıs nişancı türündedir. Oyuncular, çeşitli silahlarla donatılmış olarak düşmanları alt etmeye, hedefleri tamamlamaya ve oyun içi ödüller kazanmaya çalışır.
VizDoom, farklı senaryolar ve haritalar içeren çeşitli oyun modları sunar. Her senaryo, oyuncunun hedeflerini ve düşmanların davranışlarını belirleyen özel kurallara sahiptir. Bu senaryolar, farklı zorluk seviyeleri ve stratejik zorluklarla oyunculara çeşitlilik sunar.

2.8.2. Oyun Ortamının Tanımı

VizDoom oyun ortamı, birinci şahıs bakış açısıyla sunulan 3D bir dünyayı simüle eder. Oyuncu, oyun içinde dolaşırken çevresel bilgileri algılar ve çeşitli aksiyonlar gerçekleştirir. Oyunun girdileri, görsel veriler, ses efektleri ve oyun durumuyla ilgili diğer bilgilerden oluşur.
Oyun ortamı, oyuncunun aksiyonlarını algılayan ve oyuncuya görsel geri bildirimler sunan bir arabirim sağlar. Ajanlar, çevredeki düşmanları, hedefleri, nesneleri ve oyun içi ödülleri algılayarak kararlarını şekillendirir. Bu algılama süreci, oyun ortamının durumunu temsil eden bir durum vektörü oluşturur.
VizDoom oyun ortamı, ajanların aksiyonlarını uygulayabildiği bir arabirim de sağlar. Aksiyonlar, oyun içindeki karakterin hareketini kontrol etmek, silahları ateşlemek, nesneleri toplamak gibi çeşitli eylemleri içerir. Aksiyonlar, oyun ortamı tarafından kabul edilir ve sonuçları ortama yansıtılır.
Bu bölümde, VizDoom oyununun temel yapısı ve oyun ortamının tanımı ayrıntılı olarak incelenmiştir. Bu bilgiler, tezin odak noktası olan reinforcement learning algoritmalarının VizDoom oyununda nasıl

uygulandığını anlamak için temel bir zemin oluşturur. Sonraki bölümlerde, bu bilgileri kullanarak reinforcement learning yöntemlerinin VizDoom oyununda nasıl başarılı sonuçlar elde edebileceği araştırılacaktır.

2.9 Veri Toplama ve Ön İşleme

Reinforcement learning algoritmalarının etkin bir şekilde çalışabilmesi için, oyun ortamından yeterli ve temsilci veri toplanması gerekmektedir. Bu bölümde, VizDoom oyununda veri toplama süreci ve ön işleme adımları detaylı bir şekilde incelenecektir.

2.9.1. Veri Toplama Süreci

Veri toplama süreci, ajanın VizDoom oyununda oynayarak deneyim biriktirmesini içerir. Ajan, belirli bir politika doğrultusunda oyun ortamında aksiyonlar gerçekleştirir ve gözlem verileri toplar. Bu veriler, ajanın durumu, gerçekleştirilen aksiyonlar, elde edilen ödüller ve sonraki durumları içerir.
Veri toplama süreci genellikle bir deneme-yanılma sürecidir. Ajan, başlangıçta rastgele aksiyonlar gerçekleştirerek oyunu keşfeder ve deneyim biriktirir. Daha sonra, toplanan veriler kullanılarak reinforcement learning algoritmaları eğitilir ve ajanın stratejisi geliştirilir.

2.9.2. Veri Ön İşleme

Veri toplandıktan sonra, veri ön işleme adımları uygulanır. Bu adımlar, toplanan verileri uygun bir formata dönüştürmek ve algoritmanın kullanabileceği şekilde hazırlamak için yapılır. Veri ön işleme, aşağıdaki adımları içerebilir:

1. **Durum Uzayının Dönüştürülmesi:** Oyun ortamından alınan gözlem verileri, genellikle yüksek boyutlu ve karmaşık olabilir. Bu nedenle, verilerin boyutunu azaltmak veya daha temsili bir formata dönüştürmek için yöntemler kullanılabilir. Örneğin, görüntü verileri için görüntü işleme teknikleri uygulanabilir.

2. D**urum-aksiyon İlişkisinin Oluşturulması:** Veri setindeki her gözlem için, durum-aksiyon çiftleri oluşturulur. Durum, oyundaki ajanın mevcut durumunu temsil ederken, aksiyon, oyuncunun gerçekleştirdiği aksiyonu temsil eder. Bu durum-aksiyon çiftleri, reinforcement learning algoritmalarının eğitiminde kullanılır.

3. **Ödül Sinyalinin Hesaplanması:** Oyun ortamından elde edilen veri setinde, her aksiyon sonrası bir ödül sinyali bulunur. Bu ödül sinyali, oyuncunun başarısını veya performansını temsil eder. Ödül sinyalleri, veri setinde ilgili durum-aksiyon çiftleriyle eşleştirilir.

Veri toplama süreci ve ön işleme adımları, reinforcement learning algoritmalarının verimli bir şekilde çalışması için kritik öneme sahiptir. Doğru veri toplama stratejileri ve veri ön işleme adımları, ajanın başarılı bir şekilde öğrenmesini ve oyun ortamında optimal stratejiler geliştirmesini sağlar.

2.10 Eğitim Süreci ve Algoritmalar

Reinforcement learning algoritmaları, VizDoom oyununda ajanların stratejilerini geliştirmek için kullanılır. Bu bölümde, VizDoom oyununda reinforcement learning algoritmalarının eğitim süreci ve kullanılan bazı temel algoritmalar detaylı bir şekilde incelenecektir.

2.10.1. Eğitim Süreci

Reinforcement learning algoritmaları, ajanın deneyim biriktirerek ve toplanan verileri kullanarak stratejilerini geliştirdiği bir eğitim sürecini takip eder. Bu süreç genellikle aşağıdaki adımları içerir:

1. Başlangıç Durumu: Eğitim süreci başlangıcında ajan, oyun ortamında başlangıç durumunda yer alır. Bu durum, ajanın stratejisini geliştirmeye başladığı noktadır.

2. Aksiyon Seçimi: Ajan, mevcut durumunu gözlemleyerek bir aksiyon seçer. Bu aksiyon, mevcut stratejisi doğrultusunda belirlenir. Bazı algoritmalar, ajanın keşfetme ve sözde rasgelelik yoluyla yeni aksiyonlar denemesini teşvik edebilir.

3. Aksiyon Uygulama ve Gözlem: Seçilen aksiyon oyun ortamına uygulanır ve sonraki durumu gözlenir. Bu durum, ajanın hareketinin sonucunda ortaya çıkan yeni durumu temsil eder.

4. Ödül ve Hedef Durum: Aksiyon uygulandıktan sonra, ajan bir ödül alır ve hedef durumu belirler. Ödül, ajanın performansını ve başarısını temsil ederken, hedef durum, ajanın ilerlemesini hedeflediği durumu gösterir.

5. Öğrenme ve Strateji Güncelleme: Toplanan veriler kullanılarak reinforcement learning algoritması güncellenir ve ajanın stratejisi geliştirilir. Algoritma, ajanın mevcut durum-aksiyon çiftleriyle ödül sinyallerini eşleştirir ve stratejideki hata oranını azaltmak için parametreleri günceller.

Bu adımlar, ajanın eğitim sürecinde tekrarlanır ve ajanın stratejisi zamanla iyileşir. Eğitim süreci, hedef duruma ulaşma, optimum stratejilerin keşfi ve ajanın performansının artması gibi amaçları içerir.

2.10.2. Kullanılan Algoritmalar

VizDoom oyununda reinforcement learning için çeşitli algoritmalar kullanılabilir. Bazı temel algoritma örnekleri şunlardır:

1. Q-Learning: Q-learning, Q değerlerini güncelleyerek ve aksiyon değerlendirmeleriyle bir tabloya dayanan bir algoritmadır. Ajan, en yüksek Q değerine sahip aksiyonu seçerek stratejisini geliştirir.
2. Deep Q-Networks (DQN): DQN, derin sinir ağları kullanarak Q-learning algoritmasını genişletir. Bu algoritma, ajanın gözlem verilerini girdi olarak alır ve aksiyon değerlendirmelerini tahmin etmek için derin sinir ağını kullanır.

3. Proximal Policy Optimization (PPO): PPO, doğrudan Q değerlerini tahmin etmek yerine politika optimizasyonu üzerine odaklanan bir algoritmadır. Ajan, mevcut politikasını iteratif olarak günceller ve daha iyi performans gösteren bir politika elde eder.

Bu algoritma örnekleri, VizDoom oyununda kullanılan temel reinforcement learning algoritmalarından sadece birkaç tanesidir. Oyunun karmaşıklığına ve hedeflere bağlı olarak, farklı algoritmaların kullanılması uygun olabilir.

2.11 Performans Değerlendirmesi

VizDoom oyununda reinforcement learning algoritmalarıyla eğitilen ajanların performansının değerlendirilmesi önemli bir adımdır. Bu bölümde, ajanların performansını ölçmek ve değerlendirmek için kullanılan bazı metrikler ve yöntemler incelenecektir.

2.11.1 Skor Tabanlı Değerlendirme

Bir performans metriği olarak skor, ajanın oyun içindeki başarısını yansıtır. VizDoom oyununda, ajanın elde ettiği toplam skor, oyun içindeki hedeflere ulaşma, düşmanları yenme ve ödüllerin toplanması gibi faktörlerle ilişkilidir. Skor tabanlı değerlendirme, ajanların stratejilerini ve performanslarını karşılaştırmak için yaygın bir yaklaşımdır.

2.11.2 Başarı Oranı

Başarı oranı, ajanın belirli bir hedefi başarma yeteneğini ölçer. VizDoom oyununda, başarı oranı, ajanın belirli bir görevi tamamlama, hedefe ulaşma veya belirli bir performans eşiğini aşma gibi hedeflere ulaşma yeteneğini değerlendirmek için kullanılabilir. Başarı oranı, ajanların performansını değerlendirmek ve farklı algoritmaların etkinliğini karşılaştırmak için önemli bir metriktir.

2.11.3 Hız ve Verimlilik

Reinforcement learning algoritmaları, ajanın oyun ortamında hızlı bir şekilde öğrenmesini sağlamak için verimli olmalıdır. Eğitim süresi, bir ajanın başarılı stratejileri geliştirmesi için gereken zamanı temsil eder. Performans değerlendirmesi sırasında, farklı algoritmaların eğitim sürelerini ve verimliliklerini karşılaştırmak önemlidir.

2.11.4 Diğer Metrikler

Performans değerlendirmesi için kullanılan diğer metrikler arasında öğrenme hızı, keşfetme-istismar dengesi, ödül toplama hızı, aksiyon seçme doğruluğu ve hedefe ulaşma süresi gibi faktörler bulunabilir. Bu metrikler, ajanların performansını daha ayrıntılı ve kapsamlı bir şekilde değerlendirmek için kullanılabilir.

Performans değerlendirmesi, reinforcement learning algoritmalarının etkinliğini değerlendirmek ve geliştirmek için kritik öneme sahiptir. Bu bölümde tartışılan metrikler ve yöntemler, ajanların performansını objektif bir şekilde ölçmek ve farklı algoritmaların karşılaştırılmasını sağlamak için kullanılabilir.

3. SONUÇLAR ve TARTIŞMA

3.1 Sonuçlar

Reinforcement learning algoritmalarının VizDoom oyununda kullanılmasıyla ilgili olarak yapılan deneylerden elde edilen sonuçlar oldukça olumlu olmuştur. Eğitim sürecinde ajan, deneme yanılma yoluyla oyunu öğrenmiş ve stratejilerini geliştirmiştir. Skor tabanlı değerlendirme ve başarı oranı metrikleri kullanılarak yapılan değerlendirmeler, ajanın zaman içinde daha yüksek skorlar elde ettiğini ve hedeflerini daha etkili bir şekilde gerçekleştirdiğini göstermektedir.

Ayrıca, farklı reinforcement learning algoritmalarının karşılaştırılmasıyla ilgili olarak yapılan analizler, her bir algoritmanın avantajlarını ve dezavantajlarını ortaya koymuştur. Örneğin, Q-learning tabanlı algoritmaların hızlı öğrenme yeteneklerine sahip olduğu, ancak daha karmaşık oyun ortamlarında daha yavaş performans gösterebileceği belirlenmiştir. Deep Q-Networks gibi derin öğrenme temelli algoritmaların ise daha karmaşık oyunlar için daha uygun olduğu gözlenmiştir.

3.2 Karşılaşılan Zorluklar

Bu çalışma sürecinde, VizDoom oyununda reinforcement learning kullanarak bir botun öğrenmesi ve gelişimiyle ilgili bazı zorluklarla karşılaşılmıştır. Aşağıda, bu zorluklar ve yapılan değerlendirmeler sunulmaktadır.

1. **Karmaşık Oyun Ortamı**: VizDoom oyunu, birinci şahıs bakış açısıyla oynanan ve çeşitli görevleri içeren bir oyun olduğundan karmaşık bir oyun ortamı sunmaktadır. Bu karmaşıklık, ajanın öğrenme sürecini etkileyebilir ve daha fazla veri ve hesaplama gücü gerektirebilir. Ayrıca, oyunun dinamik yapısı ve hızlı tepki gerektiren durumları, ajanın stratejilerini geliştirme sürecinde zorluklar yaratabilir.

2. **Veri ve Hesaplama Gücü**: Reinforcement learning algoritmaları genellikle büyük miktarda veriye ve hesaplama gücüne ihtiyaç duyar. VizDoom oyununda da benzer bir durum söz konusudur. Ajanın optimal stratejileri öğrenmesi için yeterli sayıda oyun deneyimi gerekmektedir. Ayrıca, derin öğrenme tabanlı algoritmaların eğitimi, yoğun hesaplama gücü gerektiren işlemleri içerir.

3. **Yanlış Stratejilere Sapma**: Ajanın reinforcement learning sürecinde yanlış stratejilere sapması veya takılıp kalması gibi durumlarla karşılaşılabilmektedir. Özellikle başlangıçta rastgele aksiyon seçimiyle başlayan ajan, doğru stratejileri öğrenene kadar

bazen hatalı adımlar atabilir. Bu durum, öğrenme sürecinin daha uzun sürmesine ve yanlış stratejilerin düzeltilmesine yönelik ek çalışmaların yapılmasını gerektirebilir.

4. **Hiperparametre Ayarı**: Reinforcement learning algoritmalarının performansı, kullanılan hiperparametrelerin doğru bir şekilde ayarlanmasına bağlıdır. Hiperparametreler, öğrenme hızı, keşif ve sözlüğün boyutu gibi faktörleri içerir. Bu hiperparametrelerin doğru bir şekilde ayarlanması, ajanın öğrenme sürecinin hızını, kararlılığını ve performansını etkiler. Ancak, optimal hiperparametre ayarının bulunması zorlu olabilir ve deneysel çalışmalar gerektirebilir.

Bu zorluklara rağmen, bu çalışma VizDoom oyununda reinforcement learning kullanarak olumlu sonuçlar elde etmeyi başarmıştır. Karşılaşılan zorluklar, araştırmacıların daha karmaşık oyunlarda daha etkili öğrenme algoritmaları ve stratejileri geliştirmesi gerektiğini vurgulamaktadır. Ayrıca, veri ve hesaplama gücüne erişim, algoritma parametrelerinin ayarlanması ve hatalı stratejilerin düzeltilmesi gibi konularda daha fazla çalışma ve iyileştirme yapılması gerekmektedir.
Bu zorluklar, VizDoom oyununda reinforcement learning araştırmalarının gelecekteki çalışmalar için bir yol gösterici niteliktedir. Daha derin ve karmaşık öğrenme algoritmalarının kullanımıyla, bu zorlukların üstesinden gelinebilir ve oyun yapay zekası alanındaki gelişmelere katkıda bulunulabilir.

3.3 Tartışma

Sonuçlar bize, VizDoom oyununda reinforcement learning kullanarak bir botun etkili bir şekilde öğrenme ve strateji geliştirme yeteneğine sahip olduğunu göstermektedir. Ancak, bazı zorluklar ve sınırlamalar da belirlenmiştir. Örneğin, oyunun karmaşıklığı, ajanın öğrenme sürecini etkileyebilir ve daha fazla veri ve hesaplama gücü gerektirebilir. Ayrıca, ajanın bazen yanlış stratejilere sapabileceği ve takıldığı durumlarla karşılaşabileceği de gözlemlenmiştir.
Tartışma bölümünde, bu çalışmanın sınırlamaları ve gelecekte yapılacak çalışmalar için öneriler ele alınmaktadır. Örneğin, daha karmaşık oyunlarla deneyler yapılabilir ve farklı hiperparametre ayarları ile daha iyi sonuçlar elde etmek için optimizasyon çalışmaları yapılabilir. Ayrıca, derin öğrenme ve doğal dil işleme tekniklerinin birleştirildiği daha karmaşık bir modelin kullanılması da düşünülebilir.
Sonuç olarak, bu çalışma, VizDoom oyununda reinforcement learning kullanarak bir botun öğrenme ve strateji geliştirme yeteneğini incelemiş ve olumlu sonuçlar elde etmiştir. Bu çalışma, oyun yapay zekası alanında daha ileri araştırmalar ve geliştirmeler için bir temel oluşturmuştur.

3.4 Öneriler ve Gelecek Çalışmaları

Bu çalışmanın sonuçları ve tartışmaları doğrultusunda, VizDoom oyununda reinforcement learning kullanımıyla ilgili bazı öneriler ve gelecek çalışma alanları aşağıda sunulmaktadır:

1. **Daha Derin ve Karmaşık Modellerin İncelenmesi**: Bu çalışma, Q-learning, Deep Q-Networks ve Proximal Policy Optimization gibi popüler reinforcement learning algoritmalarını kullanmıştır. Ancak, daha derin ve karmaşık öğrenme modellerinin VizDoom oyununda performansı nasıl etkilediği ayrıntılı bir şekilde incelenebilir. Örneğin, Recurrent Neural Networks (RNN) veya Transformer tabanlı modeller gibi farklı yapılar ve mimariler üzerinde araştırmalar yapılabilir.

2. **Hiperparametre Ayarının İyileştirilmesi**: Hiperparametrelerin doğru bir şekilde ayarlanması, reinforcement learning algoritmalarının performansını etkileyen kritik bir faktördür. Gelecek çalışmalarda, otomatik hiperparametre ayarlama tekniklerinin kullanılması veya daha kapsamlı hiperparametre arama stratejilerinin uygulanması düşünülebilir. Bu şekilde, algoritmanın öğrenme hızı, kararlılığı ve genel performansı daha iyi optimize edilebilir.

3. **Çoklu Görev Öğrenme ve Genelleme Yeteneği**: VizDoom oyununda, farklı senaryolar ve haritalar bulunmaktadır. Gelecek çalışmalarda, bir ajanın birden fazla görevi aynı anda öğrenebilme yeteneği ve bu öğrenmenin genelleme becerisi üzerinde durulabilir. Aynı zamanda, bir haritadan diğerine geçiş yapabilme ve daha genel bir öğrenme yeteneği kazanabilme hedeflenmelidir.

4. **Daha İyi Veri Toplama ve Ön İşleme Stratejileri**: Veri toplama süreci, öğrenme performansını etkileyen önemli bir faktördür. Gelecek çalışmalarda, veri toplama stratejileri ve ön işleme yöntemleri üzerinde daha fazla çalışma yapılabilir. Örneğin, dengeleme teknikleri, deneyim tekrarını artıran yöntemler veya hedefe yönelik veri toplama stratejileri gibi yaklaşımlar kullanılabilir.

5. **Diğer Oyun Ortamlarında Uygulama**: VizDoom, oyun yapay zekası araştırmaları için popüler bir platformdur. Ancak, farklı oyun ortamlarında da reinforcement learning algoritmalarının kullanımı incelenebilir. Örneğin, Atari oyunları veya StarCraft gibi strateji oyunları gibi farklı oyunlar üzerinde çalışmalar yapılabilir.

Bu öneriler ve gelecek çalışma alanları, VizDoom oyununda reinforcement learning kullanımının daha ileriye taşınmasına ve oyun yapay zekası araştırmalarındaki gelişmelere katkıda bulunmaya yöneliktir. Gelecekteki çalışmalar, bu önerileri dikkate alarak daha kapsamlı ve etkili sonuçlar elde etme potansiyeline sahiptir.

4. KAYNAKÇA

Kolter, J. Zico., Introduction to Reinforcement Learning, Carnegie Mellon University, (2016).

Kolter, J. Zico., Markov Decision Processes, Carnegie Mellon University, (2016).

Kolter, J. Zico., *Reinforcement Learning*, Carnegie Mellon University, (2016).

Pavone, Marco., Optimal and Learning-based Control - Intro to reinforcement learning, Stanford University, (2018).

Hastie, Friedman, and Tibshirani, The Elements of Statistical Learning, 2001

Bishop, Pattern Recognition and Machine Learning, 2006

Ripley, Pattern Recognition and Neural Networks, 1996

Duda, Hart, and Stork, Pattern Classification, 2nd Ed., 2002

Tan, Steinbach, and Kumar, Introduction to Data Mining, Addison-Wesley, 2005.

Scholkopf and Smola, Learning with Kernels, 2002

Mardia, Kent, and Bibby, Multivariate Analysis, 1979

Computational Statistics (online book)

Sutton and Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

Bertsekas and Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, 1996.

Zhang, Xiaoqin., Ma, Huimin., "Pretraining Deep Actor-Critic Reinforcement Learning Algorithms With Expert Demonstrations", Dept. of EE, Tsinghua University, 2018.

Szepesvári, Csaba., Algorithms for Reinforcement Learning, Morgan & Claypool Publishers, 2010.

VizDoom Scenarios Description

Build a Doom AI Model with Python

Automating Doom with Deep Q-Learning: An Implementation in Tensorflow

An introduction to Policy Gradients with Cartpole and Doom

Reinforcement Learning : Markov-Decision Process

Epsilon-Greedy Algorithm in Reinforcement Learning