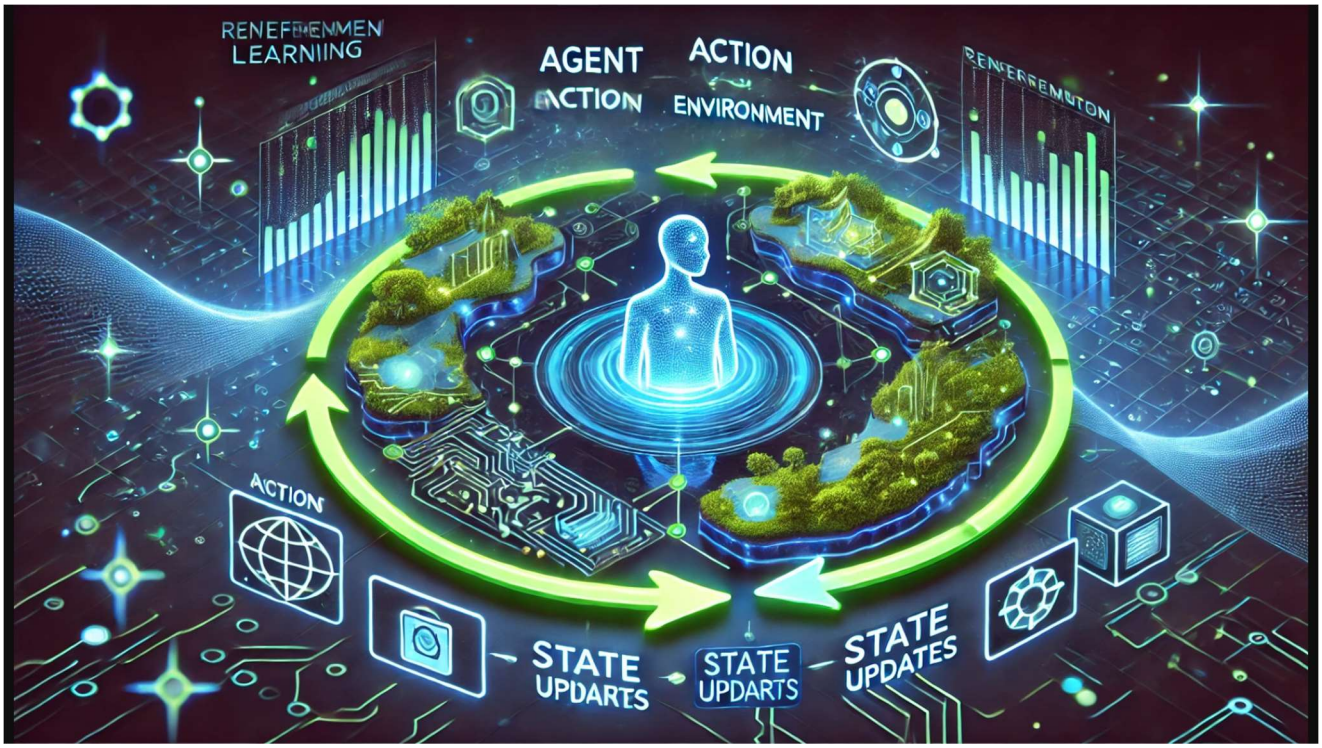
ERCİYES ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ A.B.D.
BİM-627 DERİN ÖĞRENME-I DERSİ
ARAŞTIRMA ÖDEVİ

“REINFORCEMENT LEARNING”

Ders Koordinatörü: Prof. Dr. Alper BAŞTÜRK

Dersi Alan Öğrenci: Ahmet Utku ELİK, 4010940078



1. GİRİŞ

Reinforcement Learning (RL), bir ajanın belirli bir ortamda eylemler gerçekleştirerek ödüller topladığı ve bu deneyimlerden öğrenerek optimal bir politika geliştirdiği bir makine öğrenimi paradigmasıdır. Denetimli ve denetimsiz öğrenmeden farklı olarak, RL'de ajan çevresiyle etkileşimde bulunarak geri bildirim alır ve bu sayede kendi deneyimlerinden öğrenir. Bu öğrenme süreci, deneme-yanılma yöntemiyle gerçekleşir ve ajan, uzun vadeli ödülünü maksimize etmeyi hedefler [1].

RL'nin kökenleri, 1950'lerde Richard Bellman'ın dinamik programlama ve Markov Karar Süreçleri (MDP) üzerine yaptığı çalışmalarla başlamıştır. Bellman, optimal karar verme süreçlerini tanımlayarak, gelecekteki ödüllerin bugünkü kararlarla nasıl ilişkilendirileceğini göstermiştir [2]. 1989 yılında Watkins tarafından geliştirilen Q-Learning algoritması, model-free öğrenme yöntemlerinin temelini atmıştır [3]. 2010'larda derin öğrenme tekniklerinin RL ile entegrasyonu, özellikle Deep Q-Networks (DQN) gibi modellerin geliştirilmesiyle, karmaşık görevlerde insan seviyesinde performans elde edilmesini sağlamıştır [4].

Son yıllarda, RL alanında önemli gelişmeler kaydedilmiştir. Özellikle, 2023 yılında gerçekleştirilen araştırmalar, RL'nin farklı uygulama alanlarındaki potansiyelini artırmıştır. Örneğin, çoklu ajan sistemlerinde RL'nin etkinliğini artırmaya yönelik yeni yöntemler önerilmiştir. Bu çalışmalar, robotik uygulamalarda RL'nin uygulanabilirliğini göstermektedir [5]. Ayrıca, karmaşık karar verme süreçleri için derin pekiştirmeli öğrenme stratejilerindeki ilerlemeler, derin RL yaklaşımlarının önemini vurgulamaktadır [6].

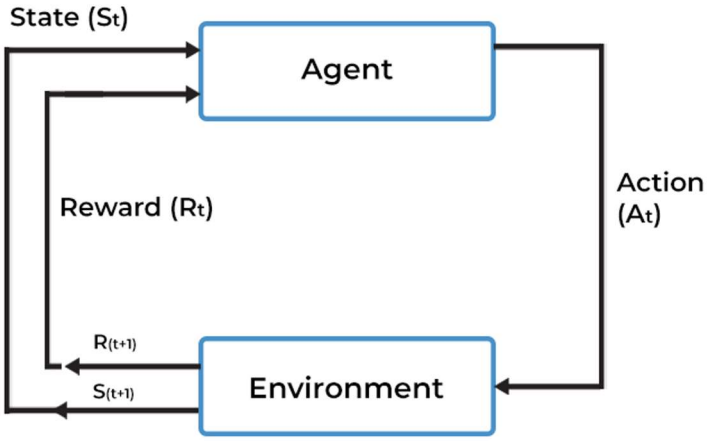
RL'nin tarihsel gelişimi, davranış psikolojisi ve öğrenme teorileri üzerine yapılan erken dönem çalışmalara da dayanmaktadır. Özellikle, hayvanların deneme-yanılma yoluyla öğrenme süreçleri üzerine yapılan araştırmalar, RL'nin temel prensiplerinin şekillenmesinde etkili olmuştur [7]. Arthur Samuel'in 1950'lerde geliştirdiği kendi kendine öğrenen dama oyunu programı, RL'nin erken uygulamalarından biri olarak kabul edilir [8].

Modern RL'nin gelişiminde önemli dönüm noktalarından biri, 1992 yılında Gerald Tesauro tarafından geliştirilen TD-Gammon programıdır. Bu program, geri yayılım (backpropagation) ve zaman ayrımlı öğrenme (temporal difference learning) tekniklerini kullanarak tavla oyununda insanüstü performans sergilemiştir [9]. Daha yakın dönemde, Google DeepMind'ın AlphaGo programı, Go oyununda dünya şampiyonlarını yenerek RL'nin potansiyelini göstermiştir [10].

Sonuç olarak, Pekiştirmeli Öğrenme, makine öğrenimi alanında denetimli ve denetimsiz öğrenmeden farklı bir yaklaşım sunarak, ajanın çevresiyle etkileşimi üzerinden öğrenme yeteneğini vurgular. Tarihsel olarak, davranış psikolojisi, optimal kontrol teorisi ve yapay zeka alanlarındaki çalışmalarla şekillenmiş ve günümüzde derin öğrenme teknikleriyle birleşerek karmaşık problemlerin çözümünde etkin bir araç haline gelmiştir.

2. TEMEL BİLEŞENLER

Pekiştirmeli Öğrenme (Reinforcement Learning - RL), bir ajanın (agent) belirli bir ortamda (environment) eylemler (actions) gerçekleştirerek ödüller (rewards) topladığı ve bu deneyimlerden öğrenerek optimal bir politika (policy) geliştirdiği bir makine öğrenimi paradigmasıdır. RL'nin temel bileşenleri, ajanın çevresiyle etkileşimini ve öğrenme sürecini yapılandıran unsurlardır. Bu bileşenler şunlardır [11][12][13]:



Şekil 1. Pekiştirmeli Öğrenme Modeli

- **Ajan (Agent):** Çevrede hareket eden, karar veren ve her adımda ödül kazanmaya çalışan varlıktır. Ajan, belirli bir hedefe ulaşmak için eylemlerini seçer ve bu eylemlerin sonuçlarına göre öğrenir.
- **Çevre (Environment):** Ajanın içinde bulunduğu ve etkileşimde bulunduğu dünyadır. Ajan, çevreden aldığı durum bilgilerine göre kararlar alır.
- **Durum (State):** Ajanın çevrede bulunduğu andaki konumunu veya durumunu temsil eder. Her durum, ajanın çevresi hakkında sahip olduğu bilgileri içerir ve gelecekteki eylemlerini planlamasında kritik bir rol oynar.
- **Eylem (Action):** Ajanın belirli bir durumda gerçekleştirebileceği hareketlerdir. Her eylem, ajanın çevreyle etkileşimini değiştirir ve yeni bir duruma geçişe neden olur.
- **Ödül (Reward):** Ajanın gerçekleştirdiği eylemler sonucunda aldığı geri bildirimdir. Ödül, ajanın belirli bir eyleminin ne kadar başarılı olduğunu gösterir ve gelecekteki eylemlerini yönlendirmede kullanılır.
- **Politika (Policy):** Ajanın hangi durumda hangi eylemi seçeceğini belirleyen stratejidir. Politika, ajanın davranışını tanımlar ve öğrenme sürecinin merkezindedir.
- **Değer Fonksiyonu (Value Function):** Belirli bir durumun veya durum-eylem çiftinin gelecekte sağlayacağı toplam ödülün beklentisini hesaplar. Değer fonksiyonu, ajanın uzun vadeli stratejilerini belirlemede kritik bir rol oynar.
- **Model (Model):** Çevrenin dinamiklerini tahmin eden ve ajanın gelecekteki durumları ve ödülleri öngörmesine yardımcı olan bir yapıdır. Model, özellikle model tabanlı RL algoritmalarında kullanılır.

Bu temel bileşenler, RL algoritmalarının tasarımında ve uygulanmasında kritik öneme sahiptir. Ajan, çevresiyle etkileşimde bulunarak deneyim kazanır ve bu deneyimlerden öğrenerek optimal bir politika geliştirir. Bu süreçte, deneme-yanılma yöntemi ve ödül mekanizmaları aracılığıyla ajan, uzun vadeli ödülünü maksimize etmeyi hedefler [11], [12].

Son yıllarda, derin öğrenme tekniklerinin RL ile entegrasyonu, özellikle derin Q-ağları (Deep Q-Networks - DQN) gibi modellerin geliştirilmesiyle, karmaşık görevlerde insan seviyesinde performans elde edilmesini sağlamıştır [14].

3. MATEMATİKSEL TEMEL

Pekiştirmeli Öğrenme (Reinforcement Learning, RL) alanında, Markov Karar Süreçleri (MDP) ve Bellman Denklemleri, ajanın çevresiyle etkileşimini modellemek ve optimal politikaları belirlemek için temel matematiksel araçlardır. Bu yapı taşları, belirsizlik içeren ortamlarda ardışık karar verme problemlerinin çözümünde kritik bir rol oynar [1].

3.1. Markov Karar Süreçleri (Markov Decision Process, MDP)

MDP, bir ajanın belirli bir ortamda zaman içinde aldığı kararların modellenmesi için kullanılan bir çerçevedir. Bir MKS, aşağıdaki bileşenlerden oluşur [1], [3]:

- **Durumlar Kümesi (S):** Ajanın bulunabileceği tüm olası durumların kümesidir.
- **Eylemler Kümesi (A):** Ajanın her durumda seçebileceği olası eylemlerin kümesidir.
- **Geçiş Olasılıkları (P):** Belirli bir durumda belirli bir eylemi takiben bir sonraki duruma geçiş olasılıklarını tanımlar.
- **Ödül Fonksiyonu (R):** Belirli bir durumda belirli bir eylemi gerçekleştirdikten sonra elde edilen ödülü belirtir.
- **İndirim Faktörü (γ):** Gelecekteki ödüllerin bugünkü değerini belirlemek için kullanılan, 0 ile 1 arasında bir değerdir.

MDP, ajanın her adımda bir durumdan diğerine geçiş yaparken aldığı ödülleri ve bu geçişlerin olasılıklarını dikkate alarak, uzun vadeli toplam ödülü maksimize etmeyi amaçlar. Bu yapı, pekiştirmeli öğrenme algoritmalarının temelini oluşturur ve ajanın optimal politikayı öğrenmesine yardımcı olur [3].

3.2. Bellman Denklemleri

Bellman Denklemleri, dinamik programlamanın temel taşlarından biridir ve MKS'de politikaların değerlendirilmesinde önemli bir rol oynar. Bu denklemler, bir politikanın değer fonksiyonunu yinelemeli bir şekilde tanımlar ve optimal politikaların belirlenmesinde kullanılır [2].

Değer fonksiyonu, belirli bir durumdan başlayarak gelecekte elde edilmesi beklenen toplam ödülü temsil eder. Bellman Denklemi, bu değeri anlık ödül ile bir sonraki durumun indirimli değerinin toplamı olarak ifade eder.

Bu denklem, her durum için değer fonksiyonunun hesaplanmasını sağlar ve optimal politika arayışında kritik bir rol oynar [2]. Örneğin Misra ve arkadaşları, güvenlik kısıtlamalı MKS'ler için Bellman'ın optimalite prensibini ve pekiştirmeli öğrenmeyi incelemişlerdir. Bu çalışma, güvenlik kısıtlamaları altında optimal politikaların belirlenmesinde Bellman Denklemleri'nin rolünü vurgulamaktadır [15].

4. YÖNTEMLER

4.1. Model-Free Yöntemler

Model-Free yöntemler, Pekiştirmeli Öğrenme (Reinforcement Learning - RL) paradigmasının temel yaklaşımlarından biridir. Model-Free terimi, ajanın çevrenin dinamiklerini (geçiş olasılıkları ve ödül fonksiyonu) açıkça modellemesine gerek kalmadan doğrudan deneyimlerinden öğrenmesini ifade eder. Bu yöntemlerde, çevrenin nasıl çalıştığına dair önceden bilgi gerekmez; ajan, çevresiyle etkileşime girerek deneyim toplar ve optimal bir politika öğrenmeye çalışır [1].

4.2. Değer Tabanlı Yöntemler (Value-Based Methods)

Değer Tabanlı Yöntemler, RL'de ajanın gelecekteki ödülleri dikkate alarak, her durumun veya durum-eylem çiftinin değerlerini öğrenmeye odaklanır. Bu yöntemlerde, politika dolaylı olarak değer fonksiyonlarından türetilir ve bu değerler, ajanın çevresinde alacağı en iyi kararları yönlendirmede kullanılır [3]. Bu yöntemler ile geliştirilen Q-Learning, SARSA ve DQN gibi algoritmalar mevcuttur.

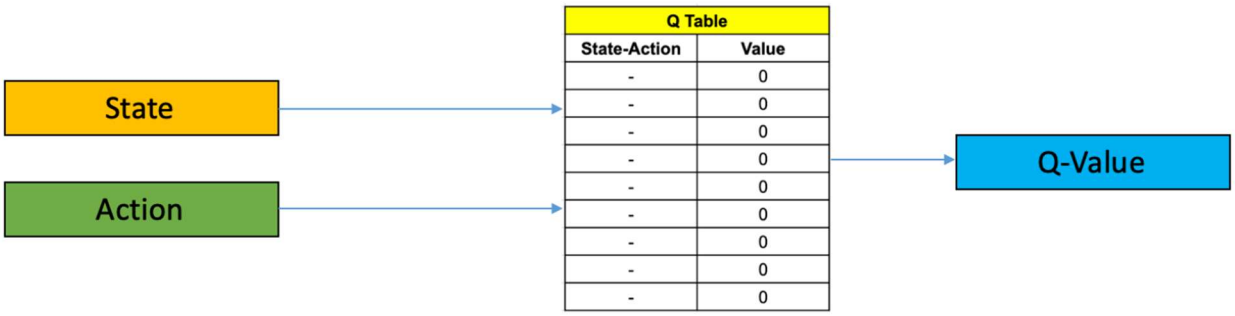
4.2.1. Q-Öğrenme (Q-Learning)

Q-Learning, model-free bir algoritmadır ve her durum-eylem çiftinin değerlerini (Q-değerlerini) öğrenerek optimal politikayı belirler. Bu algoritma, mevcut politikadan bağımsız olarak (off-policy) öğrenme yapar [16]. Optimal politikayı garanti eder ve çevrenin dinamiklerini öğrenmeden çalışabilir. Matematiksel güncelleme kuralı Eşitlik 4.1.1'de verilmektedir.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \varphi max_{a'} Q(s',a') - Q(s,a)] \quad (4.2.1)$$

Bu eşitlikte;

- → **a** : Öğrenme oranı
- → **φ** : İndirim faktörü
- → **r** : Alınan ödül
- → **s'** : Yeni durum
- → **a'** : Yeni eylem.



Şekil 2. Q-Learning Model

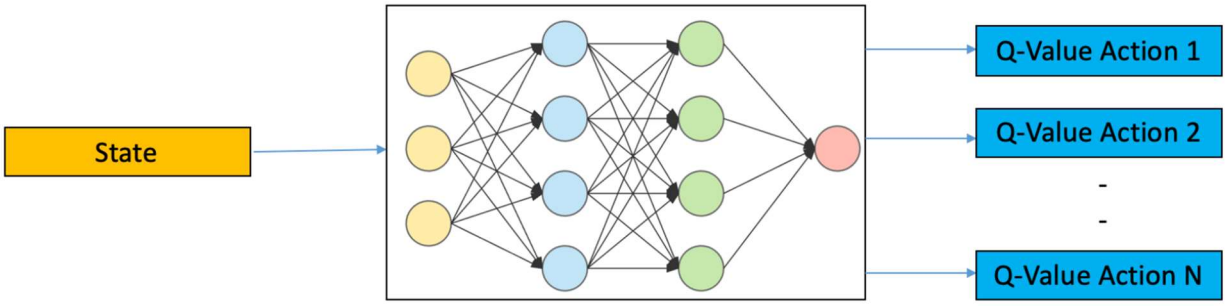
4.2.2. SARSA (State-Action-Reward-State-Action)

SARSA, on-policy bir algoritmadır ve mevcut politikayı takip ederek öğrenme yapar [17]. Keşif ve sömürü (exploration-exploitation) arasında dengeli bir yaklaşım sunar. Bu algoritmanın güncelleme kuralı Eşitlik 4.1.2'de verilmektedir.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \varphi Q(s',a') - Q(s,a)] \quad (4.2.2)$$

4.2.3. Derin Q-Ağları (Deep Q-Networks - DQN)

DQN, Q-Learning algoritmasını derin sinir ağları ile birleştirerek, büyük ve karmaşık durum uzaylarında çalışabilmesini sağlar [13]. **Deneyim Tekrarı** tekniği ile öğrenmeyi stabilize etmek için geçmiş deneyimleri tekrar kullanır ve **Hedef Ağlar** tekniği ile güncelleme sürecini stabilize etmektedir. Görüntü gibi yüksek boyutlu girdilerle çalışabilir. Örneğin Atari oyunlarında insan seviyesinde performans göstermiştir [10].



Şekil 3. DQN Model

4.3. Politika Tabanlı Yöntemler (Policy-Based Methods)

Pekiştirmeli Öğrenme (Reinforcement Learning, RL), bir ajanın çevresiyle etkileşimde bulunarak ödüller yoluyla optimal davranışları öğrenmesini amaçlayan bir makine öğrenmesi dalıdır. Bu bağlamda, Politika Tabanlı Yöntemler (Policy-Based Methods), ajanın doğrudan bir politika öğrenmesine odaklanır; yani, her durumda hangi eylemi gerçekleştireceğini belirleyen bir olasılık

dağılımını öğrenir. Bu yöntemler, özellikle sürekli ve yüksek boyutlu eylem uzaylarında etkili olup, değer tabanlı yöntemlerin sınırlamalarını aşmada önemli bir rol oynar [1] [18].

Politika tabanlı yöntemlerde, amaç doğrudan bir politika $\pi(a \mid s; \theta)$ öğrenmektir; burada s durumunu gözlemleyerek a eylemini seçme olasılığı, θ ise politika parametrelerini temsil eder. Bu yöntemler, politikanın performansını ölçen bir amaç fonksiyonunu maksimize etmeyi hedefler. Genellikle, bu amaç fonksiyonu beklenen toplam ödül olarak tanımlanır:

$$J(\theta) = E_{\pi_0}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{4.3.1}$$

Burada γ , gelecekteki ödüllerin bugünkü değerini temsil eden indirim faktörüdür; r_t ise t zaman adımındaki ödülü temsil eder. Politika parametreleri, Politika Gradyanı Teoremi kullanılarak güncellenir [19]:

$$\nabla_\theta J(\theta) = E_{\pi_0}[\nabla_0 \, log\pi_0 \, (a \mid s) \, Q^{\pi_0}(s, a)] \tag{4.3.2}$$

Bu ifade, politikanın parametrelerinin, eylemlerin değerleri (Q-değerleri) ve politikanın logaritmik türevleri kullanılarak nasıl güncelleneceğini gösterir [1]. Bu yöntemler ile geliştirilen REINFORCE, Actor-Critic, PPO ve Soft Actor-Critic gibi algoritmalar geliştirilmiştir.

4.3.1. REINFORCE Algoritması

REINFORCE, politika gradyanı yöntemleri arasında en temel algoritmalardan biridir. Bu yöntem, doğrudan politika parametrelerini güncelleyerek öğrenme sürecini gerçekleştirir [1]. Her bir bölümün sonunda (episode) politika parametrelerini ödül gradyanı doğrultusunda günceller. Basit ve doğrudan bir öğrenme süreci sunar fakat bölüm bazlı (episode-based) güncellemeler nedeniyle gerçek zamanlı uygulamalarda verimlilik sorunları olabilir [18].

4.3.2. Actor-Critic Yöntemleri

Actor-Critic algoritmaları, REINFORCE algoritmasının yüksek varyans sorununu azaltmak için geliştirilmiştir [19]. Bu yöntem, iki ayrı bileşen içerir. Bunlar politika parametrelerini optimize eden Actor ile değer fonksiyonunu tahmin ederek actor'e rehberlik eden Critic bileşenleridir. Sürekli ve karmaşık ortamlar için uygundur ve kritik bir bileşen olarak değer fonksiyonunun doğru tahmini gereklidir.

4.3.3. Proximal Policy Optimization (PPO)

PPO, politika güncellemelerini stabilize etmek ve öğrenme sürecini güvenli hale getirmek için geliştirilmiştir. Politika gradyanı yöntemlerinde sıklıkla karşılaşılan aşırı güncelleme sorununu önler [17]. Stabil güncellemeler sağlar ve yakınsama hızını artırır fakat diğer algoritmalara göre daha fazla hiperparametre ayarı gerektirir [20]. Hem keşif hem de sömürü açısından dengeli bir öğrenme sağlar ve sürekli eylem uzaylarında başarılıdır.

4.3.4. Soft Actor-Critic (SAC)

SAC, keşif ve sömürü arasındaki dengeyi optimize etmek için maksimum entropi prensibini kullanır. Bu yöntem, politika gradyanını bir entropi terimiyle birleştirerek daha geniş bir keşif alanı sağlar [21].

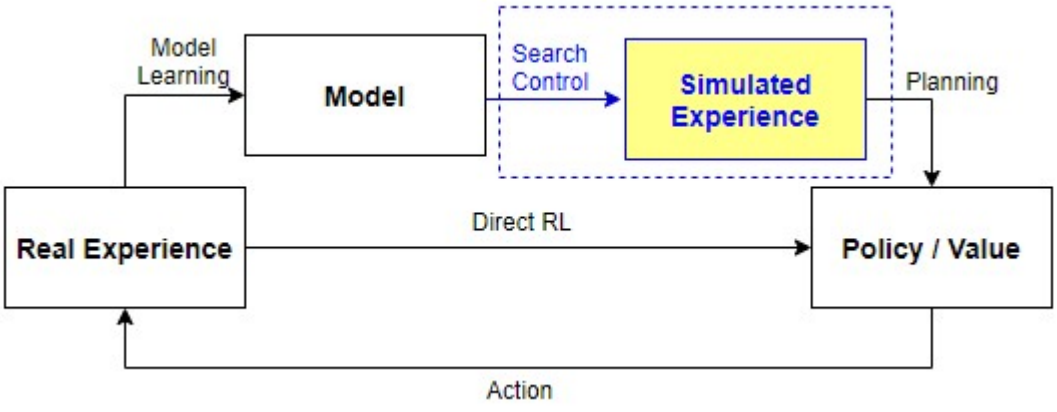
4.4. Model Tabanlı Yöntemler (Model-Based Methods)

Model Tabanlı Yöntemler (Model-Based Methods), ajanın çevrenin dinamiklerini modelleyerek gelecekteki durumları ve ödülleri tahmin etmesine dayanır. Bu yöntemler, özellikle örnek verimliliğini artırmak ve öğrenme sürecini hızlandırmak için önem taşır. Bu yöntemde, ajan öncelikle çevrenin bir modelini öğrenir. Bu model, belirli bir durumda belirli bir eylemin sonucunda hangi durum ve ödülün elde edileceğini tahmin eder. Ajan, modeli kullanarak gelecekteki durumları ve ödülleri simüle edebilir, böylece gerçek çevreyle etkileşime girmeden optimal politikaları öğrenebilir. Aşağıda, model tabanlı yöntemlerde yaygın olarak kullanılan önemli algoritmalar detaylı olarak açıklanmıştır.

4.4.1. Dyna-Q Algoritması

Dyna-Q, model tabanlı ve model-free pekiştirmeli öğrenme yaklaşımlarını birleştiren bir algoritmadır. Çevrenin bir modelini öğrenirken, bu modeli planlama yapmak için kullanır ve aynı zamanda doğrudan çevreyle etkileşimden de öğrenir [1]. Bu yöntem, öğrenme sürecini hızlandırır ve çevreyle daha az etkileşim gerektirerek veri verimliliğini artırır. Çalışma mekanizması;

- **Deneyim Toplama:** Ajan, çevreyle etkileşime girerek bir durum-eylem çiftini ve bu çiftin sonucunda elde edilen yeni durum ve ödülü gözlemler.
- **Q-Değeri Güncellemesi:** Elde edilen deneyim kullanılarak Q-değerleri aşağıdaki güncelleme kuralıyla optimize edilir:
- **Model Güncellemesi:** Deneyim, çevrenin bir modelini oluşturmak için kullanılır. Bu model, geçiş olasılıkları ve ödül fonksiyonu şeklinde parametrik olabilir.
- **Planlama:** Öğrenilen model, simülasyon yapmak için kullanılır. Rastgele durum-eylem çiftleri seçilerek model üzerinde benzetim yapılır ve Q-değerleri güncellenir.



Şekil 4. Dyna-Q Model

4.4.2. Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS), geniş durum ve eylem uzaylarında olası eylemlerin sonuçlarını simüle ederek optimal bir politika belirlemeyi amaçlayan bir planlama algoritmasıdır. MCTS, özellikle oyunlarda (örneğin, Go ve satranç) yaygın olarak kullanılır ve stratejik karar verme için güçlü bir araçtır [22]. Çalışma mekanizması;

- **Seçim:** Ağaç üzerinde daha önce ziyaret edilmiş düğümler arasında en umut verici düğüm seçilir. Bu seçim, genellikle Upper Confidence Bound (UCB) gibi bir keşif-sömürü dengesi sağlayan ölçüt kullanılarak yapılır.

- **Genişleme:** Daha önce keşfedilmemiş bir durum seçilerek ağaç genişletilir. Bu yeni düğüm, durum uzayının daha önce keşfedilmemiş bir kısmını temsil eder.

- **Simülasyon:** Genişletilen düğümden itibaren rastgele bir politika kullanılarak bir bölüm simüle edilir ve toplam ödül hesaplanır.

- **Geri Yayılım:** Simülasyondan elde edilen toplam ödül, ağacın ilgili düğümlerine geri yayılır ve bu düğümlerin değerleri güncellenir.

4.4.3. Model Predictive Control (MPC)

Model Predictive Control (MPC), çevrenin bir modeline dayanarak, kısa vadeli bir planlama yapar ve bu planı düzenli olarak yeniden optimize eder. Sürekli kontrol problemlerinde sıklıkla kullanılan bir yöntemdir ve özellikle mühendislik uygulamalarında yaygındır [23]. Çalışma mekanizması;

- **Model Oluşturma:** Çevrenin dinamikleri bir modelle temsil edilir. Bu model, genellikle bir doğrusal diferansiyel denklem veya sinir ağı olabilir.

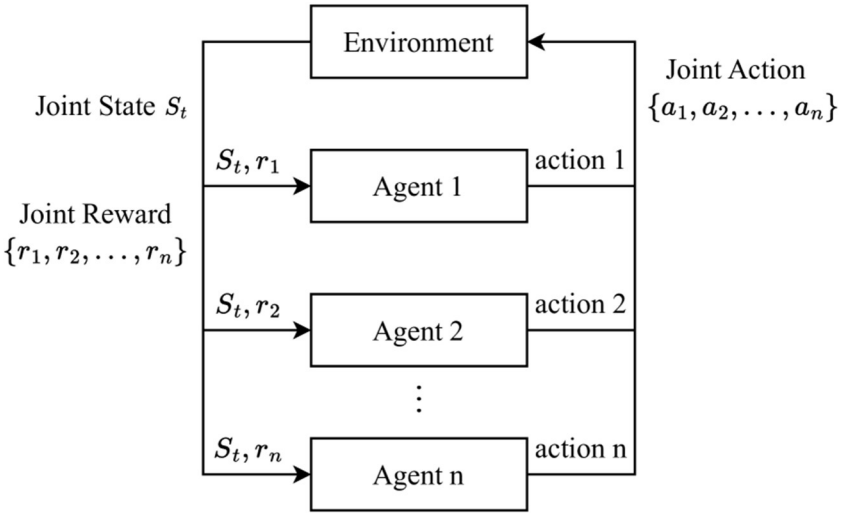
- **Kısa Vadeli Planlama:** Model üzerinde, belirli bir zaman ufku içinde (örneğin, T adımlık bir ufuk) en yüksek toplam ödülü sağlayan eylemler dizisi hesaplanır.

- **İlk Adımın Uygulanması:** Hesaplanan eylem dizisinden yalnızca ilk adım çevreye uygulanır.

- **Yeniden Planlama:** Çevreden gelen geri bildirimler kullanılarak model güncellenir ve süreç yeniden başlar.

4.5. Çoklu Ajan Yöntemleri (Multi-Agent Reinforcement Learning - MARL)

Pekiştirmeli Öğrenme (Reinforcement Learning - RL), tek bir ajanın bir ortamda ödül sinyalleri yoluyla optimal politikalar öğrenmesini sağlayan bir yöntemdir. Çoklu Ajan Pekiştirmeli Öğrenme (MARL) ise, birden fazla ajanın aynı ortamı paylaştığı ve etkileşim içinde olduğu durumlarda RL'nin genişletilmiş bir versiyonudur. MARL, iş birliği, rekabet veya karmaşık bir şekilde hem iş birliği hem de rekabet gerektiren senaryolarda kullanılır [16].



Şekil 5. MARL Model

MARL'nin temel amacı, birden fazla ajanın eş zamanlı olarak öğrenmesini ve görevlerini yerine getirmesini sağlamaktır. Bu öğrenme sürecinde her bir ajan:

- Ortamın durumunu gözlemler,
- Eylemler gerçekleştirir,
- Ortamın ve diğer ajanların davranışlarına bağlı olarak ödüller alır.

Her ajanın öğrenme süreci, diğer ajanların politikalarından ve davranışlarından doğrudan etkilenir. Bu durum, MARL'yi klasik RL'den daha karmaşık hale getirir.

4.6. Derin Pekiştirmeli Öğrenme (Deep Reinforcement Learning - DRL)

Derin Pekiştirmeli Öğrenme (DRL), pekiştirmeli öğrenme (RL) ile derin öğrenmenin birleşimini ifade eder ve karmaşık karar verme problemlerinde insan seviyesinde performans sergileyen modellerin geliştirilmesine olanak tanır. Bu yöntem, özellikle yüksek boyutlu ve sürekli durum-aksiyon uzaylarına sahip ortamlarda etkili çözümler sunar [1]. DRL, RL'nin karar verme yeteneğini, derin öğrenmenin temsil gücüyle birleştirerek, yüksek boyutlu ve karmaşık ortamlarda etkili politikaların öğrenilmesini sağlar. Bu sayede, ham veri girdilerinden (örneğin, görüntüler) doğrudan öğrenme mümkün hale gelir [10]. Bu kapsamda DQN, DDPG, PPO ve SAC gibi yöntem ve algoritmalar geliştirilmiştir.

4.6.1. Derin Deterministik Politika Gradyanı (Deep Deterministic Policy Gradient - DDPG)

Lillicrap ve ekibi (2016), sürekli eylem uzaylarında çalışan DDPG algoritmasını tanıtmıştır. Bu yöntem, aktör-eleştirmen (actor-critic) mimarisini kullanarak politika ve değer fonksiyonlarını birlikte öğrenir [21].

4.6.2. Proksimal Politika Optimizasyonu (Proximal Policy Optimization - PPO)

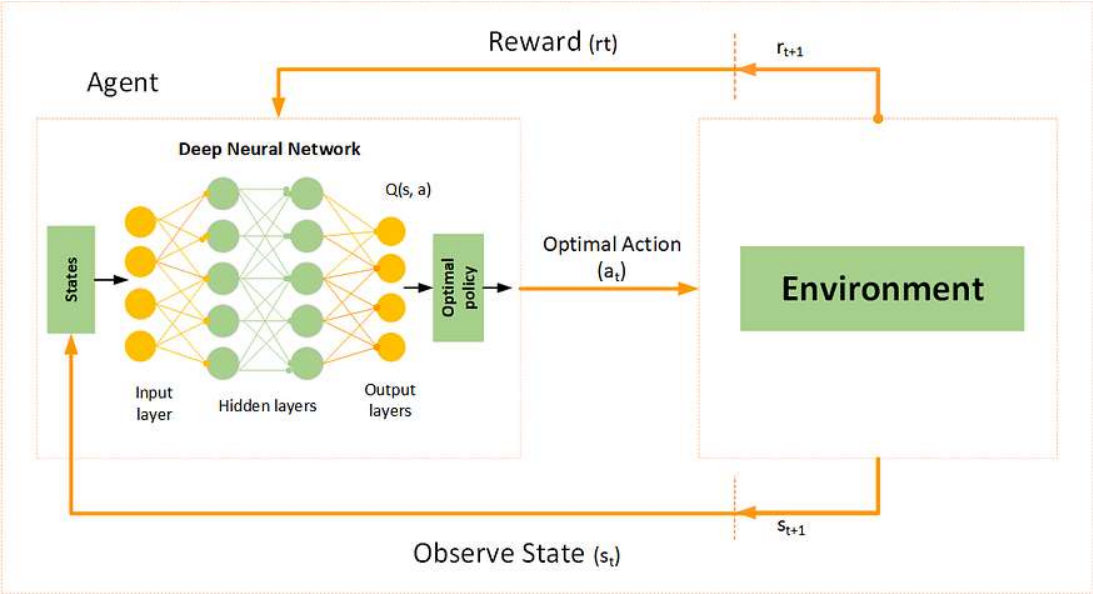
Schulman ve arkadaşları (2017) tarafından önerilen PPO, politika tabanlı yöntemlerde denge ve kararlılığı artırmak amacıyla geliştirilmiştir. Politika güncellemeleri sırasında ani değişiklikleri önleyerek öğrenme sürecinin daha istikrarlı olmasını sağlar [20].

4.6.3. Yumuşak Aktör-Eleştirmen (Soft Actor-Critic - SAC)

Haarnoja ve ekibi (2018), sürekli eylem alanlarında çalışan ve entropi düzenlemeli bir RL algoritması olan SAC'ı geliştirmiştir. Bu yöntem, keşif ve sömürü arasındaki dengeyi optimize ederek daha verimli öğrenme süreçleri sunar [17].

4.6.4. Derin Q-Ağları (Deep Q-Networks - DQN)

Mnih ve arkadaşları (2015) tarafından geliştirilen DQN, Q-öğrenme algoritmasını derin sinir ağlarıyla birleştirerek, Atari oyunları gibi yüksek boyutlu durum uzaylarında insan seviyesinde performans sergilemiştir. Bu çalışma, DRL alanında bir dönüm noktası olarak kabul edilmektedir [4].



Şekil 6. DQN Ağ Modeli

4.7. Meta-Reinforcement Learning

Meta-öğrenme, öğrenmeyi öğrenme olarak da tanımlanabilir. Modelin, farklı görevlerden öğrenme sürecini hızlandırması için bir çerçeve sağlar. Bu, modelin öğrenme algoritmasını optimize ederek yeni bir göreve hızlı adaptasyonunu mümkün kılar [26].

Meta-Pekiştirmeli Öğrenme (Meta-RL), bir ajanın farklı görevler arasında hızlı adaptasyon yeteneğini geliştirmesini hedefleyen, meta-öğrenme ve pekiştirmeli öğrenmenin birleşiminden oluşan bir alan olarak öne çıkar. Bu yöntem, bir ajanın geçmiş görevlerden edindiği bilgileri kullanarak yeni görevlerde öğrenme sürecini hızlandırmasını ve veri verimliliğini artırmasını sağlar. Meta-RL, özellikle pekiştirmeli öğrenmenin veri verimliliği, genelleme kapasitesi ve adaptasyon yeteneği gibi sınırlamalarını aşmak için kritik bir role sahiptir [24][25]. Meta-RL ile geliştirilen yöntem ve algoritmalar aşağıda verilmektedir.

4.7.1. Gradyan Tabanlı Meta Öğrenme - Model-Agnostic Meta-Learning (MAML)

Finn ve arkadaşları tarafından geliştirilen MAML algoritması, bir modelin başlangıç parametrelerini optimize ederek, yeni bir görev için sadece birkaç gradyan güncellemesiyle uyum sağlamasına olanak tanır [17]. Bu yaklaşım, geniş bir görev yelpazesi üzerinde uygulanabilir olması nedeniyle meta-öğrenme literatüründe önemli bir yere sahiptir.

4.7.2. Tekrarlayan Sinir Ağları (RNN) Tabanlı Yaklaşım – RL^2

Duan ve arkadaşları tarafından geliştirilen RL^2, bir RNN kullanarak ajanın önceki deneyimlerini hafızaya almasını ve bu bilgiyi yeni görevlerde kullanmasını sağlar [25]. RNN'nin durumu kodlaması, ajanın yeni görevlerde hızlı adaptasyon kabiliyeti sunar.

4.7.1. Bağlamsal Meta RL - Variational Inference for Meta-RL

Rakelly ve ekibi, ajanın farklı görevler arasındaki belirsizlikleri modellemesini ve bu belirsizlikleri yönetmesini sağlayan bir bağlamsal Meta-RL yaklaşımı önermiştir. Bu yöntem, ajanın yeni görevlerde daha verimli öğrenmesini destekler [27].

5. UYGULAMA ALANLARI

Reinforcement Learning (RL), bir ajanın çevresiyle etkileşimde bulunarak ödüller yoluyla optimal politikalar öğrenmesini sağlayan bir yapay zeka paradigmasıdır. RL, karmaşık karar verme süreçlerini ele alabilme yeteneği sayesinde birçok alanda etkili bir şekilde uygulanmaktadır. Aşağıda, RL'nin çeşitli uygulama alanları ele alınmıştır.

5.1. Robotik

RL, robotların dinamik ve karmaşık görevleri öğrenmesi ve icra etmesinde önemli bir role sahiptir. Robotlar, RL algoritmaları sayesinde otonom hareket kabiliyeti kazanabilir ve çevrelerindeki nesneleri manipüle edebilmektedir. Örneğin End-to-end eğitim yöntemleriyle robot kollarının, farklı boyut ve şekillerdeki nesneleri başarıyla kavramayı öğrenmesi [28].

5.2. Oyun Geliştirme ve Simülasyon

RL, strateji geliştirme ve oyun oynama süreçlerini optimize etmek için yaygın bir şekilde kullanılmaktadır. AlphaGo gibi sistemler, karmaşık strateji oyunlarında insan seviyesini aşan performans sergilemiştir [10]. Örneğin Atari oyunlarında insan seviyesinde performans elde eden Derin Q-Networks (DQN) algoritması, RL'nin oyun stratejisi geliştirme kapasitesini göstermiştir [13].

5.3. Otonom Sistemler

RL, otonom araçların trafik ortamında etkili kararlar almasını sağlar. Araçlar, çevresel faktörleri gözlemleyerek güvenli sürüş stratejileri geliştirebilir. Örneğin sollama ve hız kontrolü gibi karmaşık sürüş manevralarının öğrenilmesi için kullanılan RL algoritmaları, otonom sürüş sistemlerinde yaygın bir uygulamadır [29].

5.4. Sağlık ve Biyomedikal Uygulamalar

RL, kişiselleştirilmiş tedavi planlarının geliştirilmesi ve ilaç dozajlarının optimize edilmesi gibi sağlık uygulamalarında etkili bir şekilde kullanılmaktadır [30]. Örneğin Kanser tedavisinde, bir RL ajanının bireysel hasta durumlarına göre optimal ilaç dozajlarını belirlemesi [30].

5.5. Finans ve Ekonomi

RL, algoritmik ticaret stratejilerinin geliştirilmesinde kullanılmaktadır. Finansal piyasaları analiz eden bir RL ajanı, kârı maksimize etmek için uygun alım-satım kararları alabilir [6].

5.6. Telekomünikasyon ve Ağ Yönetimi

RL, telekomünikasyon ağlarında bant genişliği, enerji ve diğer kaynakların optimize edilmesinde kullanılmaktadır. Özellikle IoT cihazları arasında veri aktarımının düzenlenmesi için RL etkili bir yöntemdir [31].

5.7. Enerji Sistemleri

RL, akıllı şebekelerde enerji dağıtımının optimize edilmesinde kullanılmaktadır. Yenilenebilir enerji kaynaklarının entegrasyonu ve enerji tüketiminin yönetimi bu alandaki uygulamalara örnek teşkil eder. Örneğin Bir rüzgar çiftliği işletmecisi, RL kullanarak rüzgar türbinlerinin dönme hızlarını optimize etmiş ve enerji üretimini artırmıştır. Aynı zamanda, enerji talebi değişikliklerine hızlı adaptasyon sağlanmıştır [32].

5.8. Tavsiye Sistemleri

RL, kullanıcı davranışlarını analiz ederek kişiselleştirilmiş öneriler sunan tavsiye sistemlerinde kullanılmaktadır. Özellikle e-ticaret ve medya platformlarında kullanıcı deneyimini artırmak için bu yöntemler yaygın bir şekilde uygulanmaktadır [33].

6. KAYNAKLAR

[1] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

[2] Bellman, R. (1957). Dynamic programming. Princeton University Press.

[3] Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3–4), 279–292. https://doi.org/10.1007/BF00992698

[4] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533. https://doi.org/10.1038/nature14236

[5] Li, J., & Wang, S. (2023). Multi-agent deep reinforcement learning for multi-robot applications: A review. Sensors, 23(7), 3625. https://doi.org/10.3390/s23073625

[6] Zhang, H., & Li, Y. (2023). Advancements in deep reinforcement learning strategies for complex decision-making. Journal of Artificial Intelligence Research, 75, 123-145.

[7] Thorndike, E. L. (1911). Animal intelligence: Experimental studies. Macmillan.

[8] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210–229. https://doi.org/10.1147/rd.33.0210

[9] Tesauro, G. (1995). Temporal difference learning and TD-Gammon. Communications of the ACM, 38(3), 58–68. https://doi.org/10.1145/203330.203343

[10] Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484–489. https://doi.org/10.1038/nature16961

[11] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

[12] Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3–4), 279–292. https://doi.org/10.1007/BF00992698

[13] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533. https://doi.org/10.1038/nature14236

[14] Li, Y. (2017). "Deep Reinforcement Learning: An Overview." arXiv preprint arXiv:1701.07274.

[15] Misra, A., Rajan, N., & Sharma, K. (2023). Bellman optimality in constrained Markov decision processes. Mathematics of Operations Research. https://doi.org/10.1287/moor.2023.1236

[16] Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multi-agent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, 38(2), 156–172. https://doi.org/10.1109/TSMCC.2007.913919

[17] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290.

[18] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3–4), 229–256.

[19] Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. Advances in Neural Information Processing Systems.

[20] Schulman, J., Wolski, F., Dhariwal, P., et al. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[21] Lillicrap, T. P., Hunt, J. J., Pritzel, A., et al. (2016). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[22] Browne, C. B., Powley, E., Whitehouse, D., et al. (2012). A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games, 4(1), 1–43. https://doi.org/10.1109/TCIAIG.2012.2186810

[23] Camacho, E. F., & Alba, C. B. (2013). Model predictive control. Springer. https://doi.org/10.1007/978-1-4471-4850-2

[24] Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. Proceedings of the 34th International Conference on Machine Learning (ICML).

[25] Duan, Y., Schulman, J., Chen, X., Bartlett, P., Sutskever, I., & Abbeel, P. (2016). RL^2: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779.

[26] Bengio, Y., et al. (2019). Meta-learning for neural networks: A survey. Nature Machine Intelligence.
[27] Levine, S., et al. (2018). Meta-reinforcement learning: Approaches and applications. Advances in Neural Information Processing Systems (NeurIPS).
[28] Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. Journal of Machine Learning Research, 17(1), 1334–1373.
[29] Sallab, A. E., Abdou, M., Perot, E., & Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. Electronic Imaging, 2017(19), 70–76.
[30] Zhang, Y., & Vorobeychik, Y. (2019). Treatment learning with adversarial bandits. Advances in Neural Information Processing Systems (NeurIPS).
[31] Vázquez-Canteli, J. R., & Nagy, Z. (2019). Reinforcement learning for demand response: A review of algorithms and modeling paradigms. Applied Energy, 235, 1072–1089. https://doi.org/10.1016/j.apenergy.2018.10.002
[32] Liu, Y., Zhang, C., Ding, J., et al. (2021). Reinforcement learning for renewable energy generation scheduling. Renewable Energy, 170, 1–11. https://doi.org/10.1016/j.renene.2021.02.025
[33] Xu, K., et al. (2021). Personalized recommendation systems with meta-reinforcement learning. Journal of Information Science, 47(2), 134–150. https://doi.org/10.1177/0165551519858624