

T.C.  
AFYON KOCATEPE ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERİN ÖĞRENME

**REINFORCEMENT LEARNING**



212923048 Meltem Şen

212923052 Elif Uz

212923053 Saffet Tolga Güveli

# İÇİNDEKİLER

- Reinforcement Learning Nedir?
- Tarihçesi
- Pekiştirmeli Öğrenmenin Avantajları
- Pekiştirmeli Öğrenmenin Dezavantajları
- Denetimli Öğrenme
- Denetimli Öğrenme vs Pekiştirmeli Öğrenme
- Denetimsiz Öğrenme
- Denetimsiz Öğrenme vs Pekiştirmeli Öğrenme
- Pekiştirmeli Öğrenmenin Unsurları
- Markov Decision Process / Markov Karar Süreci (MDP)
- Pekiştirmeli Öğrenme Nasıl Çalışır
- Pekiştirmeli Öğrenme Ne Tür Problemleri Çözebilir?
- Pekiştirmeli Öğrenme Algoritmalarının Türleri Nelerdir?
- Model Tabanlı RL
- Modelsiz RL
- Dinamik Programlama (Dynamic Programming, DP)
- Monte Carlo Yöntemleri
- Zamansal Fark (Temporal Difference, TD) Algoritmaları
- Q-Learning
- State-Action-Reward-State-Action (SARSA)
- Politika Gradyan Yöntemleri
- On-Policy & Off-Policy
- Derin Pekiştirmeli Öğrenme Nedir?
- Deep Q Network (DQN)
- Deep Deterministic Policy Gradient (DDPG)
- Ters Pekiştirmeli Öğrenme (Inverse Reinforcement Learning - IRL) Nedir?
- IRL'in Temel Adımları
- Ters Pekiştirmeli Öğrenme (IRL) Matematiksel Modeli
- IRL'in Zorlukları
- Pekiştirmeli Öğrenme Uygulama Alanları
- Geleceği

# Reinforcement Learning Nedir?

Pekiştirmeli öğrenme, bilgisayarların bir dizi karar vererek ve sonuçlardan öğrenerek bağımsız bir şekilde öğrenmelerini sağlayan bir yöntemdir. Bu öğrenme türünde, bilgisayar programları, deneme-yanılma yoluyla belirli bir bağlamda en iyi eylemleri belirler ve performanslarını optimize eder. Bilgisayar, gerçekleştirdiği eylemlere göre olumlu veya olumsuz geri bildirim alır ve bir görevi nasıl tamamlayacağı konusunda kademeli olarak öğrenme sağlar.

Başka bir deyişle, RL, maksimum ödülü elde etmek amacıyla bir ortamda en uygun davranışları öğrenmeye dayanır. Örneğin, bir bilgisayarın tavla oyununda kazanması için eğitilmesi, yalnızca tek bir doğru hamleyi değil, bir dizi olumlu kararın alınmasını gerektirir. Bu tür oyunlarda, pek çok olası eylem ve senaryo vardır ve kısa vadeli eylemlerin uzun vadede nasıl sonuç vereceği konusunda belirsizlikler bulunur.

RL, aynı zamanda daha karmaşık kontrol problemlerinde de etkili bir şekilde kullanılabilir. Örneğin, yürüyen robotlar veya sürücüsüz arabalar gibi sistemlerin doğru şekilde yönlendirilmesinde RL teknikleri büyük rol oynar. Diğer iki öğrenme çerçevesinden farklı olarak, RL mevcut bir veri kümesi yerine çevresiyle etkileşimde bulunarak veri toplar. Bir yazılım, çevresini keşfederek, etkileşime girerek ve nihayetinde çevreden öğrenerek en uygun çözümü bulmaya çalışır.

## TARİHÇESİ

Pekiştirmeli öğrenmenin (RL) kökleri davranış psikolojisine ve öğrenme teorileri üzerine yapılan ilk çalışmalara dayanmaktadır. Ancak modern gelişimi 1950'lerde Arthur Samuel ve 1980'lerde Richard Sutton gibi araştırmacıların öncü çalışmaları ile başlamıştır. Arthur Samuel, kendi kendine öğrenen bir dama oyunu programı geliştirdi. Bu çalışma, etkileşimlerden öğrenmenin temelini attı ve pekiştirmeli öğrenmenin öncülerinden biri olarak kabul edildi. Richard Sutton, Zaman Farkı (Temporal Difference) öğrenmesi ve Q-learning üzerine çalışmalarıyla RL teorilerini önemli ölçüde geliştirdi. Pekiştirmeli öğrenmenin matematiksel temelleri bu dönemde daha da sağlamlaştı.

## TARİHÇESİ

1990'larda sinir ağlarının entegrasyonu ve 2010'ların başında derin pekiştirmeli öğrenme alanında önemli atılımlar sağlandı. RL algoritmalarının sinir ağlarıyla birleştirilmesi, algoritmaların daha karmaşık problemleri çözebilmesini sağladı. İlk olarak küçük ölçekli uygulamalar ve simülasyonlarla RL'in potansiyeli test edildi. DeepMind'ın DQN Algoritması (2013-2015) atari oyunlarında insan seviyesinde performans gösterdi. Bu gelişme, derin sinir ağlarının pekiştirmeli öğrenmeye entegrasyonunun önemini vurguladı. Derin pekiştirmeli öğrenme, RL'nin gerçek dünya problemlerine uygulanabilirliğini önemli ölçüde artırdı. 2016 yılında, Google DeepMind'ın geliştirdiği AlphaGo, Go oyununda dünyanın en iyi insan oyuncularını yendi. Monte Carlo Tree Search (MCTS), Politika (Policy) ve Değer (Value) Ağları gibi teknikler kullanıldı. AlphaGo'nun başarısı, RL'in yalnızca oyunlarda değil, stratejik ve karmaşık problemlerde de etkili olduğunu gösterdi.

## TARİHÇESİ

2018 de OpenAI Five, Valve'ın Dota 2 oyununda profesyonel oyuncularını yendi. Bu başarı, çok ajanlı (multi-agent) ortamlarda RL'in etkisini gösterdi. AlphaStar, StarCraft II oyununda insan oyuncularını geride bırakarak, RL'in dinamik ve belirsiz ortamlardaki gücünü kanıtladı.

2020'den sonra pekiştirmeli öğrenme, iş dünyasında yaygınlaştı. Reklamcılıkta dinamik stratejiler, tavsiye sistemlerinde kişiselleştirme, fiyatlandırma ve envanter yönetiminde optimizasyon sağladı. Ayrıca, robotik ve otonom araçlarda belirsiz ortamlarda karar verme süreçlerini iyileştirdi.

# GÜNÜMÜZDE

Günümüzde birçok alanda etkili çözümler sunmaktadır. Otonom sistemler kapsamında otonom araçlar, drone'lar ve robotlar, RL kullanarak karmaşık çevresel koşullarda optimize edilmiş kararlar alabilmektedir. Sağlık sektöründe, kişiselleştirilmiş tedavi planları, ilaç keşfi ve klinik karar destek sistemleri RL ile geliştirilmektedir. Finans alanında, dinamik fiyatlandırma, portföy optimizasyonu ve ticaret stratejileri için RL modelleri yaygın şekilde kullanılmaktadır. Enerji yönetiminde ise akıllı şebekeler ve enerji tüketim optimizasyonu gibi sürdürülebilir enerji çözümleri, RL sayesinde daha verimli hale gelmiştir. Ayrıca, çok ajanlı RL, meta-öğrenme ve güvenli RL gibi yenilikçi yaklaşımlar, RL'in karmaşık ve gerçek dünya ortamlarına daha güvenli ve etkili bir şekilde uygulanmasına olanak tanımaktadır.

Hiyerarşik Pekiştirmeli Öğrenme (Hiyerarşik RL), karmaşık görevler, daha küçük ve yönetilebilir alt görevlere bölünerek öğrenmeyi kolaylaştırır. Bu, daha verimli ve etkili bir öğrenme süreci sağlar. Çok Ajanlı Pekiştirmeli Öğrenme (Multi-Agent RL), birden fazla ajanın aynı ortamda iş birliği yaparak veya rekabet ederek öğrenmesini sağlar. Bu, grup dinamikleri ve sosyal etkileşim gerektiren uygulamalarda kullanılır. Offline Pekiştirmeli Öğrenme (Offline RL), ajanlar, sadece önceden toplanmış verilerle eğitilir. Bu yöntem, gerçek zamanlı etkileşim yerine mevcut verilerle öğrenmeyi mümkün kılar, özellikle güvenli ve pahalı ortamlarda kullanılır. Etik ve Güvenlik, Pekiştirmeli öğrenme modellerinin güvenli ve adil çalışması için yapılan çalışmalarda, zararlı davranışları önlemeyi ve adaletli kararlar almayı amaçlar. Bu, yapay zekanın toplumsal yarar sağlaması için önemlidir.



# Pekiştirmeli Öğrenmenin Avantajları

**Özelleştirilmiş Öğrenme:** Pekiştirmeli öğrenme, ajanların çevreleriyle etkileşime girerek deneyimlerinden öğrenmesini sağlar. Bu, çevresel koşullara göre özelleştirilmiş öğrenme süreçlerini oluşturur.

**Daha Az İnsan Etkileşimi Gerektirme:** Geleneksel makine öğrenimi algoritmalarında, veri etiketleme ve yönlendirme için insan müdahalesi gereklidir. Ancak, pekiştirmeli öğrenme, kendi başına öğrenme yeteneği sunar ve insan müdahalesine duyduğundan daha az ihtiyaç duyar. Yine de, insan geri bildirimleri ve düzeltmeleriyle de uyum sağlayabilir.

**Bağımsız Karar Verme:** Pekiştirmeli öğrenme, akıllı sistemlerin insan müdahalesi olmadan kendi başlarına kararlar almasını sağlar. Ajanlar, çevreden aldıkları ödüllere göre kendi davranışlarını ve stratejilerini uyarlayarak belirli hedeflere ulaşmaya çalışır.

**Uyarlanabilirlik:** Pekiştirmeli öğrenme, sonuçların öngörülemediği, karmaşık ve değişken ortamlarda etkili bir şekilde çalışabilir. Bu özellik, gerçek dünya uygulamaları için oldukça yararlıdır, çünkü ortamın zaman içinde değişebileceği veya belirsiz olabileceği durumlarda bile uygun çözümler geliştirebilir.

**Karmaşıklık ve İnsan Performansını Aşma:** Pekiştirmeli öğrenme, yüksek belirsizlik içeren karmaşık problemlere çözüm getirebilir ve bazen insan performansını aşarak daha etkili sonuçlar elde edebilir. Örneğin, AlphaGo, Go oyununda dünya şampiyonlarını yenerek insan düzeyini aşan bir başarıya imza atmıştır.

**Dinamik Ortamlarda Uyum Sağlama:** Pekiştirmeli öğrenme, dinamik ve değişken ortamlarda uyum sağlama yeteneği sunar. Bu da, ajanların zamanla çevreleriyle etkileşimde bulunarak daha etkili stratejiler geliştirmesini sağlar.

**Gelişmiş Performans:** Ajanlar zamanla öğrenir ve deneyim kazandıkça, daha önce görülmemiş durumlarla karşılaştığında etkili stratejiler geliştirebilir. Bu, daha karmaşık ve belirsiz ortamlarda yüksek performans göstermeyi mümkün kılar.



# Pekiştirmeli Öğrenmenin Dezavantajları

**Zaman Alıcı:** Pekiştirmeli öğrenme, çok sayıda deneme ve veri gerektirir, bu da eğitim sürecini zaman alıcı ve kaynak tüketici yapar.

**Karmaşıklık:** Karmaşık problemleri çözmek için tasarlanmış olsa da, pekiştirmeli öğrenme basit sorunlar için uygun değildir ve yönetimi zordur.

**Deneyim Eksikliği:** Pekiştirmeli öğrenme, genellikle simülasyonlarda eğitildiği için gerçek dünyadaki belirsizliklere ve yeni durumlara uyum sağlamakta zorluk yaşayabilir.

**Anlaşılması Zor:** Karmaşık sinir ağları ve algoritmalar kullanıldığından, modelin nasıl çalıştığını anlamak zor olabilir.

**Potansiyel Riskler:** Ajanlar, kendi başlarına zarar verici kararlar alabilir, bu da etik sorunlar doğurabilir.

**Kolayca Etkilenebilir:** Gürültülü veriler, dinamik ortamlar ve insan etkileşimleri ajanların performansını olumsuz etkileyebilir.

**Veri ve Hesaplama Maliyetleri:** Yüksek miktarda veri ve güçlü hesaplama kaynakları gerektirir, bu da maliyetleri artırabilir.

# Denetimli Öğrenme (Supervised Learning)

Denetimli Öğrenme (Supervised Learning), giriş verileri (input) ile bu verilere karşılık gelen doğru çıkışların (output) bir model tarafından öğrenildiği makine öğrenimi türüdür. Amaç, modelin daha sonra karşılaşacağı yeni verilere dayanarak doğru tahminler yapabilmesidir.

Nasıl Çalışır: Model, giriş (X) ve doğru etiket (Y) çiftlerinden oluşan bir veri setiyle eğitilir. Eğitim sürecinde, model doğru çıktıyı öğrenmeye çalışır ve bu öğrenme, girişlere karşılık gelen çıktı fonksiyonunu ( $f: X \rightarrow Y$ ) öğrenmesini sağlar. Eğitim tamamlandıktan sonra, model, daha önce görmediği yeni verilere dayalı tahminler yapabilir.

Kullanım Alanları:

Sınıflandırma: Verilerin belirli sınıflara atanması. Örneğin, e-postanın spam olup olmadığının belirlenmesi.

Regresyon: Sürekli değerlerin tahmin edilmesi. Örneğin, bir evin fiyatının tahmin edilmesi.

Avantajı, yüksek doğruluk sağlar ve eğitim verisindeki örneklerin etiketli olması öğrenmeyi kolaylaştırır.

Dezavantajı, çok fazla etiketli veri gerektirir. Etiketleme maliyetli ve zaman alıcı olabilir.

# Denetimli Öğrenme vs Pekiştirmeli Öğrenme

Denetimli Öğrenme, etiketli verilerle yapılan bir öğrenme türüdür. Bu yöntemde, model giriş verileri (X) ile doğru çıkış etiketlerini (Y) öğrenir. Model, etiketli veri kümesi ile eğitilir ve doğru çıkışı tahmin etmeyi öğrenir. Örneğin, bir model bir araba fotoğrafı ile ona karşılık gelen "araba" etiketini öğrenir ve daha sonra yeni fotoğrafları sınıflandırmak için bu öğrendiklerini kullanır.

Pekiştirmeli Öğrenme ise etiketli veri gerektirmez ve bir ajanın çevreyle etkileşimi yoluyla öğrenmesini sağlar. Ajan, çevresine bir dizi eylemde bulunarak etkileşimde bulunur ve her eylemden sonra ödül veya ceza alır. Bu geri bildirimlere göre ajan, gelecekte hangi eylemleri yapması gerektiğini öğrenir. Pekiştirmeli öğrenmede doğru cevapları önceden söylemek yerine, ajan deneme-yanılma yoluyla öğrenir. Örneğin, bir robot, belirli bir görevi tamamlamak için doğru hareketi yapmayı öğrenirken, başarılı eylemleri ödüllendirir ve başarısız eylemleri cezalandırır.

# Denetimsiz Öğrenme (Unsupervised Learning)

Denetimsiz Öğrenme, yalnızca giriş verilerinin (input) olduğu ve bu verilere herhangi bir etiket veya doğru çıktının verilmediği bir makine öğrenimi türüdür. Amacı, verinin altında yatan yapıyı veya kalıpları keşfetmektir.

Nasıl Çalışır: Model, yalnızca giriş verilerini (X) kullanarak veriler arasındaki gizli yapıları, kümeleri veya ilişkileri keşfeder. Kümelenmiş gruplar veya verilerin daha düşük boyutlu bir gösterimi elde edilmeye çalışılır.

Kullanım Alanları:

Kümeleme (Clustering): Benzer özelliklere sahip verilerin gruplandırılması. Örneğin, müşteri segmentasyonu.

Boyut Azaltma (Dimensionality Reduction): Yüksek boyutlu verilerin daha düşük boyutlara indirgenmesi. Örneğin, görüntü sıkıştırma.

Avantajı, etiketli veri gerektirmez, bu da veri toplama sürecini kolaylaştırır. Bilinmeyen veri kalıplarını ortaya çıkarabilir.

Dezavantajı, sonuçların yorumlanması zordur. Elde edilen sonuçlar her zaman anlamlı veya yararlı olmayabilir.

# Denetimsiz Öğrenme vs Pekiştirmeli Öğrenme

Denetimli öğrenmede veri kümesi düzgün bir şekilde etiketlenir, yani algoritmayı eğitmek için bir dizi veri sağlanır. Denetimli ve denetimsiz öğrenme arasındaki en büyük fark, denetimsiz öğrenmede tam ve temiz etiketli veri kümesinin olmamasıdır.

Denetimsiz Öğrenme, etiketlenmiş veriler olmadan verinin yapısını keşfetmeye odaklanır. Bu öğrenme türü, yalnızca giriş verileriyle çalışır ve veriler arasındaki gizli ilişkileri veya kümeleri bulmayı amaçlar. Pekiştirmeli Öğrenme ise, bir çevreyle etkileşim yoluyla öğrenme sağlar. Bu yöntemde, ajan belirli eylemleri gerçekleştirir ve her eylemin sonucunda ödül veya ceza alarak bu geri bildirimi kullanarak öğrenir. Pekiştirmeli öğrenme, çevreden aldığı geri bildirimlere dayanarak en iyi eylemi öğrenmeyi hedefler.

# Pekiştirmeli Öğrenmenin Unsurları

Agent (Ajan): Ajan, çevresinde eylemler yapan bir sistemdir. Örneğin, bir teslimat yapan drone veya bir video oyunundaki karakter, tıpkı bizlerin de günlük yaşamda eylemde bulunan birer "ajan" olmamız gibi.

Action (Eylem - A): Eylem, ajanın çevresiyle etkileşime girebileceği tüm olasılıkları ifade eder. Bir video oyununda bu, sağa/sola koşmak, zıplamak ya da durmak gibi hareketleri kapsayabilir.

Discount Factor ( $\gamma$  - İndirim Oranı): İndirim oranı, ajanın gelecekte alacağı ödüllerin önemini azaltan bir faktördür. Bu, ajanın kısa vadede daha fazla ödül almayı tercih etmesini sağlar. Eğer  $\gamma = 0,8$  ise ve 3 adımdan sonra 10 puanlık ödül varsa, bu ödülün bugünkü değeri  $0,8^3 \times 10$  olur.

Environment (Çevre): Çevre, ajanın etkileşimde bulunduğu dünya veya ortamdır. Ajanın mevcut durumu ve yaptığı eylemlerden sonra çevre, ajana yeni bir durum ve ödül sunar.

# Pekiştirmeli Öğrenmenin Unsurları

State (Durum - S): Durum, ajanın bulunduğu somut ve o anki şartlara göre belirlenen ortamdır. Ajan, bir yerdeki engeller, ödüller veya diğer unsurlar ile karşılaşabilir. Örneğin, video oyununda Mario'nun bir engel karşısındaki durumu bir "durum"dur.

Reward (Ödül - R): Ödül, ajanın eylemlerinin ne kadar başarılı olduğunu belirleyen geri bildirimdir. Örneğin, Mario'nun bir madeni parayı toplaması ödül kazanmasını sağlar.

Policy (Politika -  $\pi$ ): Politika, ajanın her durumda hangi eylemi yapacağına karar veren stratejidir. Ajan, ödül almayı en çok sağlayacak eylemi seçmeye çalışır.

Value (Değer - V): Değer, ajanın gelecekteki ödülleri beklerken, mevcut durumda elde edebileceği uzun vadeli getiri miktarını temsil eder. Bu, kısa vadeli ödüllerin ötesindeki toplam getiriyi ölçer.

Q-Value (Q-Değeri - Aksiyon Değeri): Q-değeri, mevcut durum ve eylem çiftinin uzun vadeli değerini hesaplar. Yani, ajanın mevcut durumda hangi eylemi seçmesinin, gelecekteki ödüller açısından ne kadar değerli olduğunu gösterir.



# Markov Decision Process / Markov Karar Süreci (MDP)

Markov Karar Süreci (MDP), belirsizlik altında karar verme problemlerini modelleyen matematiksel bir çerçevedir. Bu süreç, Reinforcement Learning (Pekiştirmeli Öğrenme) algoritmalarının temel yapı taşıdır. MDP, genellikle bir ajanın (agent) bir ortamda (environment) maksimum ödülü (reward) elde etmek için eylemler (actions) yaparak ilerlediği sistemleri temsil eder.

## MDP'nin Temel Bileşenleri

MDP, 4 temel bileşenden oluşur:

1. Durumlar (States) -  $S$ : Ajanın içinde bulunduğu çevrenin anlık durumunu tanımlar. Örneğin, bir robot için oda içindeki konumu bir durum olabilir.
2. Eylemler (Actions) -  $A$ : Ajanın mevcut durumdayken gerçekleştirebileceği eylemler. Örneğin, robot ileri gitmek, dönmek veya durmak gibi aksiyonlar alabilir.
3. Geçiş Olasılıkları (Transition Probabilities) -  $P$ : Bir durum ve eylem çifti sonucunda başka bir duruma geçiş olasılığını ifade eder.  $P(s' | s, a)$ : Ajanın  $s$  durumunda  $a$  eylemini yaptıktan sonra  $s'$  durumuna geçiş olasılığıdır. Markov özelliğine göre geçiş, yalnızca mevcut duruma bağlıdır, önceki durumlardan bağımsızdır.
4. Ödül (Reward) -  $R$ : Ajanın belirli bir durum veya eylem sonucunda kazandığı değerdir.  $R(s, a)$ : Ajanın  $s$  durumunda  $a$  eylemini gerçekleştirdikten sonra aldığı ödüldür.

# Markov Decision Process / Markov Karar Süreci (MDP)

MDP'nin Matematiksel Tanımı

Markov Karar Süreci, genellikle şu beşliyle tanımlanır:

$MDP=(S,A,P,R,\gamma)$

S: Durumlar kümesi.

A: Eylemler kümesi.

P: Durumdan duruma geçiş olasılıkları.

R: Ödül fonksiyonu.

$\gamma$  (Gamma): İskonto faktörü ( $0 \leq \gamma \leq 1$ ), gelecekteki ödüllerin bugünkü değerini belirler.

MDP, Markov Özelliği taşır, yani bir sonraki durum sadece mevcut duruma ve alınan eyleme bağlıdır. Bu özellik, geçmişteki durumların ajanın gelecekteki eylem seçimlerini etkilemediğini ifade eder.

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_1, s_2, \dots, s_t, a_t)$$

MDP'nin amacı, ajanın ödülleri en üst düzeye çıkarması için optimal politikayı bulmaktır. Politika, ajanın herhangi bir durumda hangi eylemi seçmesi gerektiğini belirleyen bir stratejidir.

Politika ( $\pi$ ): Durumdan eyleme geçiş fonksiyonu.

$\pi(s)$ : Durum  $s$ 'teyken ajanın alacağı eylemi belirtir.

# Markov Decision Process / Markov Karar Süreci (MDP)

## Değer Fonksiyonları

MDP'deki amaç, her durumu değerlendirerek ajanın hangi eylemleri gerçekleştireceğini belirlemektir. Bunun için değer fonksiyonları kullanılır:

Durum Değeri Fonksiyonu (State Value Function) -  $V\pi(s)$ :

Durum  $s$ 'teyken, politika  $\pi$ 'yi takip ederek ajanın elde edeceği toplam ödülün beklenen değeridir.

$$V_{\pi}(s) = \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s \right]$$

Durum-Eylem Değeri Fonksiyonu (State-Action Value Function) -  $Q\pi(s, a)$ :

Durum  $s$ 'teyken  $a$  eylemini yaptıktan sonra, politikayı izleyerek ajanın elde edeceği ödülün beklenen değeridir.

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a \right]$$

# Markov Decision Process / Markov Karar Süreci (MDP)

## Optimal Politika ve Bellman Denklemleri

MDP'deki en önemli hedef, optimal politikayı bulmaktır. Optimal politika, ajanın her durumda en iyi eylemi seçmesini sağlar ve toplam ödülü maksimize eder. Bu amaçla Bellman Denklemleri kullanılır. Bellman Denklemleri, optimum ödül ve politika için hesaplama yapar

Bellman Optimality Denklemi (Durum Değeri): Bir durumdan başlanarak gelecekte alınabilecek toplam ödülü temsil eder.

$$V^*(s) = \max_{a \in A} \left[ R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s') \right]$$

Bellman Optimality Denklemi (Durum-Eylem Değeri): Bir durum-eylem çiftinin ödül potansiyelini hesaplar.

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a')$$

# Markov Decision Process / Markov Karar Süreci (MDP)

MDP'yi çözmek için birkaç yöntem kullanılır:

Dinamik Programlama (Dynamic Programming):

Bellman denklemlerini kullanarak optimal politikayı bulmaya yönelik yöntemlerdir.

Örnek: Değer İterasyonu (Value Iteration), Politika İterasyonu (Policy Iteration)

Monte Carlo Yöntemleri:

Çevre modeline ihtiyaç duymadan, örneklemeler yoluyla öğrenme yapılır.

Temporal Difference (TD) Öğrenme:

Q-Öğrenme (Q-Learning) gibi yöntemler, ajanın çevreyle etkileşimde bulunarak öğrenmesini sağlar.

# Pekiştirmeli Öğrenme Nasıl Çalışır?

---

Pekiştirmeli öğrenmenin temelinde, bir ödül sistemi aracılığıyla en uygun davranışı veya eylemi pekiştirme kavramı yatar. Ajan, çevresindeki durumlardan aldığı ödüller ve cezalar doğrultusunda gelecekteki eylemlerini optimize eder.

---

Mühendisler, istenen davranışları ödüllendirmek, istenmeyen davranışları cezalandırmak için uygun ödül fonksiyonları tanımlar. Ayrıca, kısa vadeli ödüllerin ajanı yanlış yönlendirmesini ve genel hedefe ulaşmasını geciktirmesini önlemek için uzun vadeli hedeflerle uyumlu ödüller belirler. Bu sayede ajan, sadece anlık ödülleri değil, uzun vadede daha yüksek toplam ödülü getirecek eylemleri öğrenir.

# Pekiştirmeli Öğrenme Nasıl Çalışır?

Pekiştirmeli öğrenme; keşif, geri bildirim ve iyileştirme döngüsü ile sürekli olarak kendini geliştiren bir süreçtir.

## Pekiştirmeli Öğrenme İş Akışı

1. Problemi Tanımlayın:  
Hangi problemi çözmek istediğinizi ve ajanın bu problemdeki hedefini belirleyin.  
Örnek: Bir robotun labirentten çıkması veya otonom bir aracın trafik içinde ilerlemesi.
2. Ortamı Ayarlayın:  
Ajanın etkileşimde bulunacağı çevreyi ve bu çevredeki durum, eylem ve ödül yapılarını tasarlayın.  
Örnek: Simülasyon ortamı veya gerçek dünya ortamı.
3. Bir Ajan Oluşturun:  
Ajanın öğrenme algoritmasını, politika ve değer fonksiyonlarını tanımlayın.  
Örnek: Q-Learning, Deep Q-Network (DQN) veya Politika Gradyanı yöntemleri.
4. Öğrenmeye Başlayın:  
Ajan, belirlenen ortamda eylemler gerçekleştirir ve ödüller alarak öğrenmeye başlar.  
Örnek: Eylemler yaparak ve sonuçlarını gözlemleyerek.
5. Geri Bildirim Alın:  
Ajan, her eylemden sonra çevreden bir ödül alır ve bu ödüle göre öğrenme sürecini şekillendirir.
6. Politikayı Güncelleyin:  
Ajanın mevcut politikasını, aldığı geri bildirimlere göre güncelleyerek daha iyi kararlar almasını sağlayın.  
Örnek: Değer fonksiyonunu veya doğrudan politikayı optimize etmek.
7. İyileştirme Yapın:  
Öğrenme süreci boyunca, ödül fonksiyonunu ve öğrenme parametrelerini optimize edin.  
Örnek: Hiperparametre ayarlamaları veya ödül yapısının güncellenmesi.
8. Dağıtım Sağlayın:  
Eğitim sürecini tamamlayan ajanı, gerçek dünya uygulamasına veya hedef sistemine entegre edin.  
Örnek: Otonom bir aracın yolda test edilmesi veya bir tavsiye sisteminin kullanıcılara sunulması.



# Pekiştirmeli Öğrenme Ne Tür Problemleri Çözebilir?

Pekiştirmeli öğrenme, beklenen ve olasılıklı ortamlardaki problemlerin çözülmesine yardımcı olur. Beklenen ortamlarda, bir ödül elde etmek için eylemlerin belirli bir sırayla gerçekleştirilmesi gerekir. Diğer sıralamalar, ajanın cezalandırılmasına neden olur. Olasılıklı ortamlarda ödüller, eylemlerin olasılığını içerdiğinden daha zor bir şekilde belirlenir. Bu ortamda, ajanın eylemleri bir politika aracılığıyla belirlenir. Politika, ortamdaki koşulları göz önünde bulundurarak, ajanın hangi eylemi gerçekleştirmesi gerektiğini belirler.

## **Pekiştirmeli Öğrenme Algoritmalarının Türleri Nelerdir?**

Pekiştirilmeli öğrenmede (RL) kullanılan çeşitli algoritmalar (ör. Q-öğrenme, politika gradyan yöntemleri, Monte Carlo yöntemleri ve zamansal fark öğrenmesi) vardır. Derin RL, derin sinir ağlarının pekiştirmeli öğrenmeye uygulanmasıdır. Derin RL algoritmasına örnek olarak Güven Bölgesi Politikası Optimizasyonu (TRPO) verilebilir.

Tüm bu algoritmalar iki geniş kategoride gruplandırılabilir.

# Model Tabanlı RL

Model tabanlı RL genel olarak ortamlar iyi tanımlanmış ve değişmez olduğunda ve gerçek dünya ortam testinin zor olduğu durumlarda kullanılır. Temsilci ilk olarak ortamın dahili bir temsilini (modelini) oluşturur. Bu modeli oluşturmak için şu süreci kullanır:

Ortam içinde eylemler gerçekleştirir ve yeni durum ile ödül değerini not eder.

Eylem-durum geçişini ödül değeri ile ilişkilendirir.

Model tamamlandığında temsilci, optimum kümülatif ödüllerin olasılığına dayalı olarak eylem dizilerini simüle eder. Daha sonra eylem dizilerinin kendilerine de değer atar. Böylece temsilci, istenen hedefe ulaşmak için ortam içinde farklı stratejiler geliştirir. Bu tip RL algoritmaları bulunduğu ortamın geçiş olasılıklarına hakim olmaya çalışır. Geçiş olasılıklarını kısaca herhangi bir durumda başka bir durumun karşımıza gelme olasılığı olarak görebiliriz. Yani bu tip algoritmalar ortam dinamiklerini öğrenmeye çalışır ve bu ortam dinamikleri üzerinden aslında planlama yaparak en iyiye ulaşmaya çalışır. Model Tabanlı algoritmalar ortam dinamiklerini modellemeye çalıştığı için bu tip algoritmalar çoğunlukla planlama algoritmaları olarak geçer. Bu gibi algoritmalarda ise sorun ortamın kompleksliği ile oluşabilecek planlama ve öngörülememe hatalarıdır.

## Örnek

Belirli bir odaya ulaşmak için yeni bir binada gezinmeyi öğrenen bir robot düşünün. Başlangıçta robot serbestçe keşif yapar ve binanın dahili bir modelini (veya haritasını) oluşturur. Ana girişten 10 metre ilerledikten sonra bir asansörle karşılaştığını öğrenebilir. Haritayı oluşturduktan sonra, bina içinde sık sık ziyaret ettiği farklı konumlar arasında bir en kısa yol dizisi oluşturabilir.

# Modelsiz RL

Ortam büyük, karmaşık ve kolayca tanımlanamadığında kullanılabilecek en iyi yöntem modelsiz RL'dir. Ayrıca ortamın bilinmediği ve değiştiği durumlarda da idealdir ve ortam temelli testlerin önemli dezavantajları yoktur. Temsilci, ortamın ve dinamiklerinin dahili bir modelini oluşturmaz. Bunun yerine, ortam içinde bir deneme yanılma yaklaşımı kullanır. Bir politika geliştirmek için durum-eylem çiftlerini ve durum-eylem çift dizilerini puanlar ve not eder.

Bu tip algoritmalar modeli yani çevreyi öğrenmeye gerek duymaksızın çevrede hareket alır. Model tabanlı algoritmalar arama ve planlamaya doğru kayarken modelden bağımsız algoritmalar direkt aksiyon alımında bulunur. Alınan aksiyonlar ve bu aksiyonlara karşılık alınan ödüller üzerinden ajan eğitilir. Bu tip algoritmalar açıkça planlama yapamaz. Q Learning gibi sıklıkla kullanılan algoritmalar temporal fark kullandıkları ve çevreyi tahmin etmeye çalışmadıkları için bu kategoride yer alır.

## Örnek

Şehir trafiğinde gezinmesi gereken sürücüsüz bir araç düşünün. Yollar, trafik modelleri, yaya davranışları ve sayısız diğer faktör ortamı oldukça dinamik ve karmaşık hale getirebilir. Yapay zeka ekipleri ilk aşamalarda aracı simüle edilmiş bir ortamda eğitir. Araç, mevcut durumuna göre eylemler gerçekleştirir ve ödüller veya cezalar alır.

Araç, zaman içinde farklı sanal senaryolarda milyonlarca mil sürüş yaparak tüm trafik dinamiklerini açıkça modellemekten her durum için hangi eylemlerin en iyi olduğunu öğrenir. Araç, gerçek dünyaya girdiğinde öğrenilen politikayı kullanır ancak yeni verilerle onu iyileştirmeye devam eder.

# **Dinamik Programlama (Dynamic Programming, DP)**

Dinamik programlama yöntemleri, ortamın dinamiklerinin (geçiş olasılıkları ve ödüller) tam olarak bilindiği durumlarda kullanılır.

Nasıl Çalışır?

Dinamik programlama, Markov Karar Süreci (Markov Decision Process, MDP) çerçevesinde, gelecekteki ödülleri dikkate alarak mevcut durumu optimize eder.

Değer İterasyonu: Her bir durum için en yüksek ödül bulana kadar değer fonksiyonunu günceller.

Politika İterasyonu: Mevcut bir politika üzerinden değerleri hesaplar ve daha iyi bir politika önerir.

Avantajı, matematiksel olarak güçlüdür ve optimum çözümler sunar.

Dezavantajı, ortamın tam modeline ihtiyaç duyar ve büyük durum uzaylarında ölçeklenebilirliği düşüktür.

## Monte Carlo Yöntemleri

Monte Carlo yöntemleri, bölümlere ayrılmış (epizodik) öğrenmeye dayanır ve çevrenin modelini bilmeye gerek yoktur.

Nasıl Çalışır?

Bir bölümün tamamlanmasından sonra, ajan ödülleri toplar ve bu verilere dayanarak politikayı günceller.

First-Visit Monte Carlo: Bir durum ilk kez ziyaret edildiğinde, toplam ödül ortalamasını alır.

Every-Visit Monte Carlo: Her ziyarette o durumu tekrar ziyaret ettiğinde ödüllerin ortalamasını hesaplar.

Avantajı, basit ve anlaşılır bir yöntemdir, ayrıca model gerektirmez.

Dezavantajı, bölümün tamamlanmasını beklemek gerektiğinden, öğrenme süreci zaman alıcı olabilir.

## Zamansal Fark (Temporal Difference, TD) Algoritmaları

TD yöntemleri, Monte Carlo ve Dinamik Programlama yöntemlerinin birleştirilmiş halidir.

Nasıl Çalışır?

Bölümün tamamlanmasını beklemeden, her bir adımda durum-aksiyon değerlerini günceller. Bu, daha hızlı ve etkili öğrenme sağlar.

Q-Learning: Politika dışı bir öğrenme algoritmasıdır; en iyi eylemi öğrenmeyi hedefler.

SARSA: Politika içi bir algoritmadır; mevcut politikayı takip ederken öğrenir.

TD( $\lambda$ ): Monte Carlo ve TD yöntemlerini birleştirir; kısa ve uzun vadeli ödülleri dengeler.

Avantajı, sürekli öğrenme yapabilir ve model gerektirmez.

Dezavantajı, karmaşık ortamlar için büyük veri ve işlem gücü gerekebilir.
















## Q-Learning

Q-Learning, ünlü Bellman Denklemine dayanan, off-policy (politika dışı) ve modelsiz bir pekiştirmeli öğrenme algoritmasıdır. Pekiştirmeli öğrenme alanındaki en bilinen algoritmalardan biridir. Bu algoritmanın temel amacı, gelecekteki hareketlerini inceleyip bu hareketlere göre alacağı ödülleri görmek ve bu ödülleri maksimize etmek için hareket etmektir. Aracın ödül haritasında, gitmesini istediğimiz veya istemediğimiz yerler önceden belirlenir ve bu ödüller ödül tablosuna (Reward Table) yazılır. Aracın tecrübeleri bu ödül tablosuna göre şekillenir. Araba, ödüllere ulaşırken her iterasyonda edindiği tecrübeleri, gidebileceği yerleri seçerken ödülleri maksimize etmek için kullanır. Bu tecrübeler, Q-Tablosu adı verilen bir tabloda saklanır.

# Q-Learning

Başlangıçta aracımızın hiçbir tecrübesi olmadığı için Q-Tablosu sıfırlarla doludur. Bu nedenle aracımız ilk seçimlerinde, Q-Tablosundaki sıfırları maksimize etmeye çalışacak ve dolayısıyla rastgele hareket edecektir. Bu rastgele hareket, aracın ilk ödülünü bulmasına kadar devam edecektir. Araba ödülün bulunduğu bir yere geldiğinde, ödüle ulaşmadan önceki yerini hatırlar ve bu yerin değerini, kendi tecrübelerini biriktirdiği Q-Tablosuna yazar. Bu yazma işlemi belirli bir algoritmaya dayanır ve ajan, her adımda daha verimli bir şekilde hareket etmek için bu bilgileri kullanarak öğrenmeye devam eder.

					
					
					
					
				End	

Actions :    

Start	0	0	0	0
Nothing / Blank	0	0	0	0
Power	0	0	0	0
Mines	0	0	0	0
END	0	0	0	0

## Q-Learning

$$Q(s,a) = Q(s,a) + lr*(r(s,a) + Y*\max(Q(s',a') ) - Q(s,a) )$$

$Q(s,a)$ ” (durum, eylem) belirli bir durum (s) ve eylem (a) çifti için o anki Q-değerini ifade eder.

lr (learning rate), öğrenme katsayısıdır ve yeni öğrenilen bilgilerin eski bilgileri ne kadar güncelleyeceğini belirler.

$r(s, a)$ , mevcut durum (s) ve eylem (a) çifti için ödül değerini temsil eder.

$\gamma$  (gamma), gelecekteki ödüllerin ne kadar önemli olduğunu belirleyen indirgeme faktörüdür (discount factor).

$\max(Q(s', a'))$ , bir sonraki durumdaki (s') tüm olası eylemler için en yüksek Q-değerini ifade eder.

Ajan, bu algoritmaya göre her adımda bir sonraki durumu tahmin eder ve bu tahmini ile ödüle ulaşır. Ödüle ulaştıktan sonra, ajan rastgele bir yerden harekete başlar ve yine ödülü bulmaya çalışır. Bu işlem, ajan ödülü sürekli olarak buldukça ve Q-değerlerini güncelledikçe, hangi durumda hangi eylemi yapması gerektiğini öğrenir.

## State-Action- Reward-State- Action (SARSA)

SARSA (State-Action-Reward-State-Action) algoritması, on-policy bir algoritmadır ve ajan, mevcut politikasını kullanarak hem veri toplar hem de öğrenir. Yani, ajan hangi eylemi seçerse, bu eylem üzerinden öğrenmesini gerçekleştirir. Bu, SARSA'nın Q-Learning algoritmasından farklı olduğu bir noktadır çünkü Q-Learning off-policy bir algoritmadır ve ajan, politikadan bağımsız bir şekilde en iyi eylemi öğrenmeye çalışır.

SARSA, öğrenme döngüsündeki şu beş temel bileşenden ismini alır:

State (Durum): Ajanın bulunduğu anlık durum.

Action (Eylem): Ajanın mevcut durumdan hareketle seçtiği eylem.

Reward (Ödül): Ajanın eylemi gerçekleştirdikten sonra aldığı ödül.

Next State (Sonraki Durum): Eylemin ardından geçilen yeni durum.

Next Action (Sonraki Eylem): Yeni durumda ajan tarafından seçilen eylem.

Güncelleme Kuralı:  $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$

# Politika Gradyan Yöntemleri

Politika Gradyan Algoritmaları, doğrudan bir politika fonksiyonunu optimize etmeye yönelik çalışır. Bu tür algoritmalar, özellikle sürekli aksiyon alanlarında etkili bir öğrenme sağlar, çünkü eylemleri belirli bir olasılık dağılımına göre seçerler.

## Nasıl Çalışır?

Politika fonksiyonunu temsil eden bir olasılık fonksiyonu üzerinden çalışır ve toplam ödülü maksimize edecek şekilde gradyan hesaplar. Bu gradyan, politikanın parametrelerini güncellemeye yönelik kullanılır.

REINFORCE: Bu algoritma, her bir eylemin sonucunda elde edilen ödülü baz alarak politikanın gradyanını günceller. Yani, eylemin ödülüne bağlı olarak, politikanın parametrelerini günceller.

Actor-Critic: İki bileşen içerir: Actor ve Critic. Actor, politikayı öğrenirken Critic, değer fonksiyonunu öğrenir. Actor politikayı optimize ederken, Critic değer fonksiyonu ile her durumda eylemin ne kadar "iyi" olduğunu değerlendirir. Bu iki bileşen birlikte çalışarak daha stabil bir öğrenme sağlar.

Proximal Policy Optimization (PPO): Politika gradyan algoritmalarının kararlılığını artırmaya yönelik geliştirilmiştir. Politikanın büyük adımlarla değişmesini engelleyerek daha stabil bir öğrenme süreci sağlar. Politika güncellemeleri, daha küçük adımlarla yapılır.

Avantajı, sürekli aksiyon alanlarında ve büyük durum uzaylarında güçlüdür, çünkü doğrudan bir politika fonksiyonu üzerinde çalışır. REINFORCE, Actor-Critic ve PPO gibi algoritmalar, karmaşık ortamlarda etkili bir şekilde öğrenme sağlayabilir.

Dezavantajı, yüksek varyans nedeniyle öğrenme süreci yavaş olabilir. Politika güncellemeleri, ödülün değişkenliğine bağlı olarak büyük dalgalanmalar gösterebilir ve bu da öğrenmeyi zorlaştırabilir.

# On-Policy

On-Policy Öğrenme, ajanın mevcut politikayı ( $\pi$ ) takip ederek veri topladığı ve bu aynı politikayı optimize ettiği bir öğrenme yöntemidir. Bu tür öğrenmede, ajan yalnızca şu anda kullandığı politikalardan öğrenir ve politikasını sürekli olarak geliştirir. Ajan veri toplarken ve öğrenme yaparken aynı politikanın çıktılarından faydalanır.

## Nasıl Çalışır?

Ajan, mevcut politikasını kullanarak bir ortamda eylemler gerçekleştirir. Topladığı verilerle, bu politikanın nasıl daha iyi hale getirilebileceğini öğrenir ve yeni, iyileştirilmiş bir politika oluşturur. Önceki politika üzerinde yapılan iyileştirmelerle yeni bir politika elde edilir.

Avantajı, ajan doğrudan mevcut politikasını geliştirdiği için öğrenme süreci genellikle daha güvenli ve istikrarlıdır. Mevcut politikasını sürekli olarak optimize ederek daha iyi sonuçlar elde edebilir.

Dezavantajı, daha fazla veri gerektirebilir, çünkü yalnızca mevcut politikanın davranışlarına dayalı veri kullanır. Çeşitlilik açısından sınırlı olabilir. Ajan, yalnızca mevcut politikasına dayalı veri topladığı için, eylem seçimlerinde çeşitlilik sınırlı olabilir. Bu durum, daha geniş bir arama yapma potansiyelini kısıtlar.

## On-Policy Öğrenme Yöntemleri

SARSA, politika içi bir algoritmadır. Mevcut politikayı takip ederek hem veri toplar hem de bu politikanın nasıl iyileştirilebileceğini öğrenir.

Actor-Critic, bir yandan politika fonksiyonunu (actor) öğrenirken diğer yandan değer fonksiyonunu (critic) öğrenir. Genellikle on-policy olarak kullanılır. Mevcut politikayı hem değerlendirir hem de optimize eder.

REINFORCE, politika gradyan yöntemlerinden biridir ve on-policy öğrenme gerçekleştirir. Burada, toplam ödüle dayalı olarak politika güncellenir.



# Off-Policy

Off-policy öğrenme, ajanın bir politika ( $\pi$ ) ile veri topladığı, ancak farklı bir politika ( $\pi'$ ) optimize ettiği bir yöntemdir. Veri toplama politikası ile öğrenme politikası bağımsızdır. Ajan, genellikle keşif için bir politika kullanır, ancak öğrenme ve optimizasyon amacıyla farklı bir politika (optimizasyon politikası) kullanır. Bu durum, özellikle karmaşık ortamlarda önemli avantajlar sağlar.

## Nasıl Çalışır?

Off-policy öğrenmede, ajan ilk başta bir keşif politikasını kullanarak etkileşimde bulunur ve bu süreçte topladığı verilerle optimize etmek istediği politikayı öğrenir. Ajan veri toplama ve öğrenme aşamalarında farklı politikalar kullanır. Bu, ajanın daha geniş bir keşif yapmasını ve daha verimli öğrenmesini sağlar.

## Avantajları,

Daha verimli öğrenme: Farklı politikaların verilerini kullanarak öğrenir.

Çeşitlilik: Karmaşık ortamlarda daha fazla keşif ve çeşitlilik sunar.

Politika bağımsız öğrenme: Keşif ve optimizasyon farklı politikalarda yapılabilir.

Dezavantajları, öğrenme süreci daha karmaşık olabilir. Politika optimizasyonu için daha dikkatli tasarım gerektirir.

## Off-Policy Öğrenme Yöntemleri:

Q-Learning: Politika dışı bir algoritmadır; ödülleri maksimize eden en iyi politikayı öğrenir.

Deep Q-Networks (DQN): Derin öğrenme ile Q-Learning'in birleşimi.

Importance Sampling: Off-policy öğrenmede yaygın olarak kullanılan bir yöntemdir; farklı politikaların verilerini karşılaştırır.



# Derin Pekiştirmeli Öğrenme Nedir?

Derin pekiştirmeli öğrenme, derin öğrenme ve pekiştirmeli öğrenmenin birleşimidir. Derin öğrenme, insan beyninin yapısını taklit eden yapay sinir ağlarını kullanan bir teknikler bütünüdür. Derin öğrenme ile bilgisayarlar, açıkça programlanmadan büyük miktarda verideki karmaşık örüntüleri tanıyabilir, içgörüler çıkarabilir veya tahminlerde bulunabilir. Eğitim, denetimli öğrenme, denetimsiz öğrenme veya pekiştirmeli öğrenmeden oluşabilir. Pekiştirmeli öğrenme, bir bilgisayarın bir ortamla etkileşime girdiği, geri bildirim aldığı ve buna bağlı olarak karar verme stratejisini ayarladığı bir öğrenme modudur. Derin pekiştirmeli öğrenme ise, daha karmaşık problemleri çözmek için derin sinir ağlarını kullanan özel bir RL biçimidir. Derin pekiştirmeli öğrenmede, derin öğrenmenin ve sinir ağlarının örüntü tanıma gücü, RL'i geri bildirim tabanlı öğrenme ile birleştirir. Görsel veriler gibi yüksek boyutlu girdiler üzerinde etkili çalışır. Yüksek işlem gücü gerektirir ve tasarımı karmaşıktır.

## Deep Q Network (DQN)

Q-Learning çok güçlü bir algoritma olmasına rağmen, temel zayıflığı genel olmamasıdır. Q-learning'i iki boyutlu bir diziyi (Eylem \* Durum Alanı) güncelleyen bir sistem olarak kabul edersek, aslında dinamik programlamaya benzer. Q-öğrenme ajanının daha önce görmediği durumlarda, hangi eylemde bulunacağına dair hiçbir fikri olmadığını belirtir. Başka bir deyişle, Q-learning aracı görülmemiş durumlar için değer tahmin etme yeteneğine sahip değildir. Bu problemle başa çıkmak için DQN, Yapay Sinir Ağını kullanarak iki boyutlu diziden kurtulur.

Deep Q Network (DQN), Q-Learning algoritmasını derin öğrenme ile birleştirerek Q-değerlerini sinir ağı kullanarak tahmin eder. Bu sayede görülmemiş durumlar için değer tahmininde bulunabilir. Q-Learning'in temel zayıflığı, daha önce karşılaşılmamış durumlarda hangi eylemin yapılacağına dair bilgi eksikliğidir, ancak DQN bunu derin sinir ağları ile aşar. Q-değerlerini tahmin etmek için derin sinir ağı kullanır. Geçmiş deneyimlerden faydalananarak öğrenme süreci stabil hale gelir. Öğrenme sürecinde stabiliteyi sağlamak için hedef bir ağ kullanılır. Daha önce görülmemiş durumlar için doğru tahminler yapabilir. Karmaşık ortamlarda iyi genelleme yapar. Q-Learning algoritmasındaki zayıflıkları aşarak daha verimli ve genellenebilir bir öğrenme süreci sunar.

# Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG), off-policy ve model-free bir Reinforcement Learning algoritmasıdır. Sürekli eylem uzayında (continuous action space) çalışan bir algoritmadır ve actor-critic yapısını kullanır.

Off-Policy Öğrenme: Ajan, öğrenme sürecinde farklı bir politika kullanabilir.

Deterministik Politika: Ajan, her durumda tek bir eylem seçer (stochastic değil).

Actor-Critic Yapısı:

Actor: Eylem seçen bileşen (politika).

Critic: Politikanın kalitesini değerlendiren bileşen (değer fonksiyonu).

DDPG Nasıl Çalışır?

Actor (Politika) Ağı: Ajanın hangi eylemi seçeceğini belirler.

Critic (Değer) Ağı: Aktif politikanın ne kadar iyi olduğunu belirlemek için Q-değerini tahmin eder.

Deneyim Yeniden Örneklemeye (Experience Replay): Ajanın önceki deneyimlerinden faydalanarak öğrenme sürecini daha verimli hale getirir.

Hedef Ağı (Target Networks): Stabil öğrenme için kritik ve actor ağlarının kopyaları kullanılır.

Avantajları: Sürekli eylem uzayında verimli çalışır. Düşük boyutlu girişlerle büyük ve karmaşık ortamları öğrenebilir. İyi bir keşif ve sömürü dengesi sağlar.

Dezavantajları: Büyük ağlar ve çok sayıda parametre gerektirir. Kararlı öğrenme için dikkatli hiperparametre ayarları gerekir.

# Ters Pekiştirmeli Öğrenme (Inverse Reinforcement Learning - IRL) Nedir?

Ters Pekiştirmeli Öğrenme (IRL), bir ajanın, çevresiyle etkileşim yoluyla ödülleri öğrenmeye çalıştığı Pekiştirmeli Öğrenme algoritmalarının tersine, uzman bir ajanın davranışlarını gözlemleyerek bu ajanın izlediği politikayı ve ödül fonksiyonunu öğrenmeye odaklanır. IRL'in amacı, bir uzmanın davranışlarını anlamak ve bu davranışları modellemek için öğrenme süreçlerini tersine çevirmektir.

# Ters Pekiştirmeli Öğrenme (IRL) Nedir?

Ters Pekiştirmeli Öğrenme, ajanın örneklerle (uzman ajanın davranışlarıyla) ödül fonksiyonunu keşfetmesini sağlayan bir yöntemdir. Geleneksel pekiştirmeli öğrenme, çevreyle etkileşim yoluyla ödülleri öğrenmeye çalışırken, ters pekiştirmeli öğrenme çevreyle doğrudan etkileşime girmeden, yalnızca gözlemler yoluyla öğrenir.

## Temel Fark

Pekiştirmeli Öğrenme (RL): Ajan, ödülleri maksimize etmeye çalışırken ödül fonksiyonu bilinir.

Ters Pekiştirmeli Öğrenme (IRL): Ajan, uzman ajanın davranışlarını gözlemleyerek ödül fonksiyonunu öğrenmeye çalışır.

# IRL'in Temel Adımları

Ters pekiştirmeli öğrenme genellikle şu adımları içerir:

**Uzman Davranışlarının Gözlemi:** Ajan, bir uzman ajanın çevrede gerçekleştirdiği eylemleri gözlemler. Örneğin, bir insan oyuncusunun veya bir robotun çevreyle etkileşimi.

**Politika Tahmini:** Uzman ajanın gözlemlerini kullanarak, ajanın izlediği politikayı tahmin ederiz. Bu, ajanın hangi durumlarda hangi eylemleri tercih ettiğini anlamayı içerir.

**Ödül Fonksiyonunun Öğrenilmesi:** Uzman davranışını taklit edebilmek için, ajanın ödül fonksiyonunu öğrenmesi gerekir. Bu ödül fonksiyonu, ajanın çevreye karşı nasıl hareket etmesi gerektiğini belirler.

**Politika İyileştirmesi:** Öğrenilen ödül fonksiyonu ile, ajan, daha fazla gözlem yaparak ya da simülasyon yaparak politikasını iyileştirir ve hedefe daha uygun davranışlar sergiler.

# Ters Pekiştirmeli Öğrenme (IRL) Matematiksel Modeli

IRL'de amaç uzman ajanın çevresindeki her durumu nasıl değerlendirdiğini anlamaktır. İdeal olarak, ajan, uzman ajanın eylemlerini gözlemleyerek ödül fonksiyonunu öğrenir. Bu süreç genellikle şu adımlarla ifade edilir:

**Uzman Davranışları:** Uzman ajanın, belirli bir ortamda aldığı eylemler dizisini gözlemliyoruz.

**Ödül Fonksiyonu:** Ajanın öğrendiği ödül fonksiyonu, uzman ajanın eylemlerinin neden belirli şekilde yapıldığını açıklar.

**Politika ( $\pi$ ):** Uzman ajanın izlediği politika, en iyi ödülü elde etmek için yaptığı eylemlerdir. IRL, bu politikayı modelleyerek, ajanın nasıl davranması gerektiğini öğrenmeye çalışır.

**Değerleme (Value Estimation):** Ajan, ödül fonksiyonunu öğrenirken her durumun değerini tahmin eder.

# IRL'in Zorlukları

Ters Pekiştirmeli Öğrenme, güçlü bir yaklaşım olmakla birlikte bazı zorluklar da taşır:

Gözlem Verisi: Yeterli ve kaliteli gözlem verisi elde etmek zor olabilir. Uzman davranışlarının zengin bir şekilde gözlemlenmesi gerekir.

Çoklu Ödül Fonksiyonları: Aynı gözlemlerle farklı ödül fonksiyonları öğrenilebilir. Bu da, ajan için daha karmaşık bir öğrenme süreci yaratabilir.

Ödül Fonksiyonunun Tam Olarak Bilinmemesi: Eğer uzman ajanın ödül fonksiyonu tam olarak bilinmiyorsa, öğrenme süreci daha zorlu hale gelebilir.



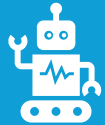
# Pekiştirmeli Öğrenme Uygulama Alanları



**Sağlık Hizmetleri:** Hasta verilerini ve geçmiş ziyaret bilgilerini inceleyerek her hastaya özel tedavi planları oluşturabilir. Bu, tıbbi teşhisleri hızlandırarak, hastaların daha hızlı ve kişiselleştirilmiş tedaviler almasını sağlar. Ayrıca, iyileşme süreçlerinin zaman çizelgelerini de hesaba katarak tedavi planlarını optimize eder.



**Enerji:** Sensörlerden toplanan verileri analiz ederek enerji tüketimini tahmin edebilir. Ekipler, veri merkezlerinde soğutma işlemlerini gerçekleştirirken, enerji ve maliyetleri en aza indiren ideal koşulları belirleyebilir.



**Üretim:** Fabrikalarda ve depolarda, robotların bilgisayar görüş sistemlerine güç verir. Mobil robotlar, depo koridorlarında gezinmeyi ve kazalardan kaçınırken envanteri taşımayı öğrenebilir.

# Pekiştirmeli Öğrenme Uygulama Alanları



**Pazarlama:** Kişiselleştirilmiş önerilerle müşterileri hedeflemeyi kolaylaştırır. Müşterilerin hangi ürünleri ve web sayfalarını görüntülediklerini analiz ederek, ilgilerini çekebilecek diğer ürünler belirlenebilir.



**Oyun:** Video oyunlarında oyuncu olmayan karakterlerin yapay zekalarını geliştirir. Bu yapay zekalar, farklı saldırı ve savunma taktikleri benimseyebilir ve oyunda yeni yollar keşfedebilir.



**Robotik:** Derin öğrenme ve pekiştirmeli öğrenme, robotları daha önce karşılaşmadıkları nesneleri kavrayacak şekilde eğitebilir. Özellikle montaj hattı gibi ortamlarda kullanılabilir.

# Pekiştirmeli Öğrenme Uygulama Alanları



**Otomotiv:** Sürücüsüz araçların güvenli bir şekilde çalışmasını sağlamak için gerçekçi ortamlarda eğitim verir. Algoritmalar, şeritte kalma, hız sınırına uyma ve diğer sürücülerle yayaları dikkate alma gibi faktörleri öğrenir.



**Ulaşım:** Şehirler, trafik sıkışıklığıyla mücadele etmek için pekiştirmeli öğrenmeye yönelmektedir. Trafik sinyalleri, araç sayısı ve saat gibi değişkenleri dikkate alarak, trafik ışıklarını en verimli şekilde kontrol etmenin yolları bulunur.



**Müşteri Hizmetleri (NLP):** Doğal dil işlemeyi geliştiren önemli bir tekniktir. Bu teknoloji, sohbet robotları ve sanal asistanlar gibi müşteri hizmetleri sistemlerini mümkün kılar. Metin özetleme, soru yanıtlama ve makine çevirisi gibi uygulamalar bu yöntemle çalışır.

# Pekiştirmeli Öğrenme Uygulama Alanları



**Eğitim:** Öğrenciler için kişiselleştirilmiş öğrenme deneyimleri oluşturulabilir. Öğrenci ihtiyaçlarına uyum sağlayan, bilgi boşluklarını belirleyen ve eğitim sonuçlarını geliştiren özel ders sistemlerine imkan tanır.



**Ticaret ve Finans:** Karar verme süreçlerinde bilgisayarları eğitmek için kullanılır. Bu teknoloji, finansal piyasalarda, hisse senedi alım satımı ve fiyat tahminleri gibi işlemler için kullanılır. Otomatik ajanlar, hisse senedini satmaya, almaya veya tutmaya karar verebilir, böylece gerçek zamanlı kararlar alınabilir.

# GELECEĞİ

## Otonom Sistemler ve Robotik

Robotlar, insansız hava araçları (dronlar) ve otonom araçlar gibi sistemlerin, çevreleriyle daha etkili ve güvenli bir şekilde etkileşime girebilmelerini sağlayacak. Gelecekte, bu sistemlerin daha hızlı öğrenmesi ve daha doğru kararlar vermesi bekleniyor.

**Otonom Araçlar:** Otonom araçlar, trafik, yol koşulları ve diğer araçlarla etkileşimde bulunarak daha güvenli bir şekilde seyahat edebilir. Bu araçlar, sürücü müdahalesi olmadan güvenli ve verimli bir şekilde yol alabilecekler.

**Robotik Manipülasyon:** Robotlar, insan benzeri beceriler geliştirerek daha karmaşık görevleri öğrenebilir ve gelişmiş işlerde kullanılabilir.

## Sağlık Alanında Uygulamalar

**Pekiştirmeli öğrenme,** sağlık alanında hastalık teşhisi, tedavi önerileri, ilaç keşfi ve kişiselleştirilmiş tedavi planları gibi birçok önemli alanda devrim yaratabilir.

**Kişiselleştirilmiş Tedavi:** Pekiştirmeli öğrenme ajanları, bireylerin sağlık durumlarına göre kişiselleştirilmiş tedavi planları önererek tedavi süreçlerini daha etkili hale getirebilir.

**İlaç Keşfi:** Yeni ilaçların geliştirilmesinde kullanılabilir. Bu yöntem, ilaçların etkilerini daha hızlı ve verimli bir şekilde analiz etmemize olanak tanıyabilir.

# GELECEĞİ

## Eğitim ve Öğrenme Sistemleri

Öğrencilere kişisel hızlarına ve tarzlarına göre özelleştirilmiş öğrenme deneyimleri sunar. Öğrenme platformları, öğrencilerin gelişimine göre dinamik içerikler sağlayarak daha verimli eğitim süreçleri oluşturur.

Öğrenci Performansı: Öğrencilerin öğrenme süreçleri kişisel seviyelere göre optimize edilir.

Dinamik Eğitim Sistemleri: Öğrencilere özel eğitim yöntemleri ve içerikler sunulur.

## Yapay Zeka ve İnsan Etkileşimleri

Pekiştirmeli öğrenme, yapay zekanın insanlarla daha doğal etkileşim kurmasını sağlar. Gelecekte, yapay zeka kullanıcıların niyetlerini anlayarak daha uygun tepkiler verebilir.

İnsan-Bilgisayar Etkileşimi: Yapay zeka, insan davranışlarını analiz ederek kişiselleştirilmiş tepkiler verir.

Akıllı Asistanlar: Pekiştirmeli öğrenme kullanan akıllı asistanlar daha verimli çalışabilir.



**BİZİ DİNLEDİĞİNİZ İÇİN TEŞEKKÜR EDERİZ.**