Pekiştirmeli Yapay Zeka Öğrenmesi

Pekiştirmeli Yapay Zeka Öğrenmesi (Reinforcement Learning – RL), yapay zeka alanında kendi kendine öğrenen sistemler geliştirmek için kullanılan güncel ve oldukça başarılı sonuçlar veren bir teknolojidir. Öğrenme sürecini deneme yanılma yoluyla gerçekleştirir.

Bu makaleyi Spotify'da sesli olarak dinlemek için podcast'ine bu linkten ulaşabilirsiniz.

1.1 Pekiştirmeli Yapay Zeka Öğrenmesinin Temel Tanımı ve Mekanizması

RL, diğer yapay zeka algoritmalarından daha farklı bir çalışma mekanizmasına sahiptir. **Ödül-ceza sistemi** (rewards and punishments) üzerine kuruludur. Bu sistemde bir **ajan** (agent), belirli bir **ortam** (environment) içerisinde **eylemler** (actions) gerçekleştirir. Ajanın yaptığı eylemler sonucunda, ortamdan pozitif veya negatif geri bildirimler alır. Pozitif geri bildirimler "ödül" (reward), negatif geri bildirimler ise "ceza" (punishment) olarak adlandırılır. Aslında ceza, ödülün negatif değer alması durumudur.

RL'in temel hedefi, ajanın ortam üzerindeki toplam **kümülatif ödülünü maksimize etmektir**. Ajan ne kadar fazla pozitif ödül alırsa, hedefe o kadar optimum ve hızlı bir şekilde ulaşır.

Bu makaleyi Youtube'da görüntülü olarak izlemek için videosuna bu linkten ulaşabilirsiniz.

1.2 Temel Kavramlar

- **Ajan (Agent)**: Öğrenen veya öğrenmesi gereken varlıktır. Bir yazılım, simülasyondaki bir oyun karakteri, fiziksel dünyadaki bir otomobil veya drone olabilir.
- **Ortam (Environment)**: Ajanın içinde bulunduğu ve etkileşimde bulunduğu fiziksel veya sanal dünyadır. Ajan, ortamdan geri bildirimler alır. Endüstriyel bir simülasyon, bir oyun veya siber güvenlik senaryosundaki ağ topolojisi bir ortam olabilir.
- **Eylem (Action)**: Ajanın ortamda gerçekleştirdiği hareketlerdir. Pacman oyununda "sola dön", "sağa dön" gibi eylemler, sürücüsüz araçta "direksiyonu çevir", "hızlan" gibi eylemler olabilir.
- **Durum (State)**: Ajanın ortam üzerindeki mevcut pozisyonu veya durumudur. Görsel olabileceği gibi (bir karedeki konum), siber güvenlik senaryosunda bir saldırı aşamasındaki nokta da olabilir.
- **Ödül (Reward)**: Ajanın yaptığı bir eylem sonucunda ortamdan aldığı pozitif veya negatif puandır. Hedefe ulaşma pozitif ödül, duvara çarpma negatif ödül olabilir. Reward değeri ajan tarafından belirlenir, hangi eylemin ne kadar ödül veya ceza alacağı konfigürasyon dosyalarında ayarlanır.
- **Politika (Policy)**: Ajanın durumunu eylemlerle eşleme yöntemidir. Ajanın belirli bir

durumda hangi eylemi seçeceğine karar veren algoritmadır.
- **Değer (Value)**: Bir ajanın belirli bir durumda eylemde bulunarak alacağı gelecekteki ödülün tahminidir. Q-value gibi kavramlarla ilişkilidir.

1.3 Çalışma Döngüsü

Bir RL sürecinde ajan, ortamda bir **durum (S)** içindedir ve bir **eylem (A)** gerçekleştirir. Ortam bu eylemin sonucunda ajana bir **ödül (R)** verir ve ajanı yeni bir **duruma (S')** geçirir. Bu S-A-R-S' döngüsü, ajanın deneyerek ve yanılarak öğrenmesini sağlar.

1.4 RL'in Uygulama Alanları (Use Case'ler)

RL, birçok farklı sektör ve alanda başarılı bir şekilde kullanılmaktadır:

- **İnsansız Otomobil Çalışmaları**: Otonom sürüş sistemlerinin temelini oluşturur. Sanal simülasyon ortamlarında (örn. GTA 5 gibi oyunlar environment olarak kullanılarak) milyonlarca kilometre sürüş bilgisi öğrenilir.
- **Robotik**: Zıplayan, takla atan, koşan robotların temel AI algoritması RL üzerine kuruludur. Kollaboratif çalışmalarda da kullanılabilir.
- **Finansal Analizler**: Borsa, stok ve kripto para analizlerinde kullanılır.
- **Doğal Dil İşleme (NLP)**: Metin özetleme (text summarization), soru cevaplama (question answering) ve makine çevirisi (machine translation) gibi alanlarda başarılı sonuçlar verir. ChatGPT gibi büyük dil modellerinde "Reinforcement Learning from Human Feedback (RLHF)" yaklaşımı kullanılır.
- **Endüstri**: Geniş bir kullanım alanına sahiptir. Fabrika otomasyonu ve optimizasyonunda simülasyon ortamları oluşturularak uygulanır.
- **Oyunlar**: AlphaGo Zero gibi Go şampiyonu yenen yapay zekalar RL kullanır. Bilgisayar oyunlarında AI botları geliştirilerek insan oyunculara karşı mücadele eder. Pacman gibi oyunların kendi kendine öğrenen versiyonları yapılabilir.
- **Simülasyon**: Havacılıkta insansız drone gibi sistemlerde kullanılır. SpaceX roketlerinin iniş algoritmalarında dengeleme için RL kullanılır.
- **Veri Merkezleri Soğutması (Data Centers Cooling)**: Google'ın veri merkezlerinde enerji tüketimini %40-50'ye kadar azaltmada kullanılmıştır.
- **Trafik Işık Kontrolleri**: Şehir içi trafik akışını optimize etmek için trafik lambalarının yönetiminde kullanılır.
- **Sağlık (Healthcare)**: Medikal görüntü işleme (CT taramaları gibi) ve dinamik tedavi rejimleri (DTR) alanlarında potansiyeli vardır.
- **Siber Güvenlik**: Kendi kendine öğrenen saldırı (hacking) veya savunma algoritmaları geliştirmede kullanılır. Sonsuz olasılıklı ağ topolojileri üzerinde güvenlik zafiyetlerini manipüle edebilir.
- **Derleyici Optimizasyonu (Compiler Optimization)**: LLVM gibi derleyicilerde kod optimizasyonu görevleri için RL ortamları oluşturulabilir.
- **Reklam Sunumu (Ad Serving)**: Kullanıcının davranışlarına göre hangi reklamların gösterileceğine karar veren algoritmalarda kullanılır.

1.5 Önemli Kavramlar ve Yaklaşımlar

1. **Exploration (Keşfetme) ve Exploitation (Değerlendirme) İkilemi**:
   - **Exploitation**: Bilinen en iyi eylemi yapmaya odaklanmaktır. Ajan, daha önce öğrendiklerini en iyi şekilde kullanarak mevcut ödülü maksimize etmeye çalışır.
   - **Exploration**: Yeni eylemler denemek ve bilinmeyen durumları keşfetmektir. Ajan risk alarak daha yüksek ödüllerin potansiyelini araştırır.
   - Bu ikilem arasında bir denge kurulmalıdır. Çok fazla exploitation, ajanın yeni ve daha iyi çözümleri bulmasını engellerken, çok fazla exploration zaman ve kaynak israfına yol açabilir. Bu dengeyi sağlamak için **epsilon-greedy** gibi yaklaşımlar kullanılır. Epsilon-greedy, belirli bir olasılıkla (epsilon) keşfetme, kalan olasılıkla ise değerlendirme yapar.
2. **Markov Karar Süreçleri (Markov Decision Process – MDP)**:
   - Reinforcement Learning problemlerinin neredeyse tamamı MDP'ler kullanılarak formüle edilebilir. MDP, RL'in matematiksel bir çerçevesidir.
   - Bir MDP, sonlu bir durum kümesi (states) ve her durumda alınabilecek eylemler (actions) kümesinden oluşur. Ajanın geçmiş durumları umrunda değildir; bir sonraki duruma geçiş olasılığı mevcut duruma bağlıdır.
3. **Model-Free (Model Bağımsız) ve Model-Based (Model Tabanlı) Reinforcement Learning**:
   - **Model-Free RL**: Ajanın ortamın bir modeline (yani durum geçişlerini ve ödülleri tahmin eden bir fonksiyona) doğrudan erişimi yoktur. Bunun yerine, ajanın politikası veya değer fonksiyonu doğrudan deneyimlerden öğrenilir. Q-learning ve SARSA gibi algoritmalar model-free yaklaşımlardır. Genellikle daha popüler ve üzerinde daha fazla araştırma yapılmış yöntemlerdir.
   - **Model-Based RL**: Ajanın ortamın bir modeline erişimi vardır. Bu model, ajanın geleceği düşünerek plan yapmasını sağlar. Örneğin, otonom araçlarda simülasyonda öğrenilmiş ağırlıklar veya parametreler bir model olarak saklanır ve ajanın sıfırdan başlamamasını sağlar. AlphaGo Zero ünlü bir model-based RL örneğidir.
4. **Algoritmik Yaklaşımlar**:
   - **Q-Learning**: Model-free bir yaklaşımdır. Ajanın belirli bir durumda belirli bir eylemi gerçekleştirmenin değerini ifade eden "Q değerlerini" güncellenmesi kavramı etrafında döner. Q-değeri, bir Q-tablosunda tutularak sürekli güncellenir.
   - **SARSA**: Q-learning gibi yaygın kullanılan model-free bir algoritmadır. Keşif stratejileri açısından farklılık gösterir ancak kullanım stratejileri benzerdir. SARSA, "on-policy" (ilke-içi) bir yöntemken, Q-learning "off-policy" (ilke-dışı) bir yöntemdir.
   - **Derin Pekiştirmeli Öğrenme (Deep Reinforcement Learning – DRL)**: Reinforcement Learning algoritmalarının **derin öğrenme** (deep learning) mimarileriyle birleştirilmesidir. Örneğin, **DQN** (Deep Q-Network), Q-learning'i evrişimli sinir ağları (Convolutional Neural Networks – CNN) ile birleştirerek ajana görsel verileri anlama yeteneği kazandırır.

- **Politika Optimizasyon Algoritmaları**: PPO, DDPG, SAC, A2C, A3C, Policy Gradient gibi algoritmalar model-free kategorisinde politika optimizasyon yaklaşımlarıdır.

1.6 Ortam Çeşitleri

RL ortamları, çeşitli özelliklerine göre sınıflandırılabilir:

- **Eylem Alanına Göre (Action Space)**:
  - **Ayrık Eylem Alanı (Discrete Action Space)**: Ajanın alabileceği eylemlerin birbirinden bağımsız ve sınırlı olduğu durumlardır (örn. Pacman'de "yukarı", "aşağı", "sol", "sağ").
  - **Sürekli Eylem Alanı (Continuous Action Space)**: Ajanın eylemlerinin sürekli bir aralıkta olduğu durumlardır (örn. sürücüsüz arabada direksiyonun dönüş açısı, hızlanma).
- **Belirleyiciliğe Göre (Determinism)**:
  - **Deterministik Ortamlar (Deterministic Environments)**: Ortamın bir sonraki durumu, mevcut duruma ve ajanın eylemine göre her zaman kesin olarak belirlenebilir (örn. direksiyonu sola çevirince arabanın kesinlikle sola gitmesi).
  - **Stokastik Ortamlar (Stochastic Environments)**: Ortamın bir sonraki durumu, mevcut duruma ve ajanın eylemine göre kesin olarak belirlenemez, rastlantısal unsurlar içerebilir (örn. frene basıldığında frenin patlama olasılığı).
- **Ajan Sayısına Göre**:
  - **Tek Ajanlı Ortamlar (Single-Agent Environments)**: Ortamda yalnızca bir ajanın bulunduğu ve ortamla etkileşime girdiği durumlardır (örn. A noktasından B noktasına giden tek bir otonom araç).
  - **Çok Ajanlı Ortamlar (Multi-Agent Environments)**: Ortamda birden fazla ajanın bulunduğu durumlardır (örn. araba yarışları, eş zamanlı çalışan robotlar). Bu tür ortamlarda ajanlar arası iletişim önemli bir zorluktur. Hiyerarşik Reinforcement Learning (Hierarchical RL) gibi yaklaşımlar, ana görevin parçalara ayrılarak birden fazla ajan tarafından yapıldığı senaryolarda kullanılabilir.
- **Durum Alanına Göre (State Space)**:
  - **Ayrık Ortamlar (Discrete Environments)**: Durum alanının ayrık ve sınırlı olduğu ortamlardır (örn. Grid World'deki belirli kare konumları).
  - **Sürekli Ortamlar (Continuous Environments)**: Durum alanının sürekli bir aralıkta olduğu ortamlardır (örn. bir arabanın hız, ivme ve direksiyon rotasyonu gibi sürekli değişen durumları).
- **Zaman Yapısına Göre**:
  - **Epizodik Ortamlar (Episodic Environments)**: Ajanın eylemlerinin yalnızca belirli bir bölümle sınırlı olduğu ve önceki bölümlerle bağlantılı olmadığı ortamlardır (örn. bir topu üç bardak arasında bulma oyunu).
  - **Sıralı Ortamlar (Sequential Environments)**: Ajanın daha önce gerçekleştirdiği eylemlerin mevcut ve gelecekteki durumları etkilediği

ortamlardır (örn. satranç oyunu, her hamlenin birbiriyle ilişkili olduğu durumlar).
- **Gözlemlenebilirliğe Göre**:
  - **Tamamen Gözlemlenebilir Ortamlar (Fully Observable Environments)**: Ajanın herhangi bir zamanda ortamın tam durumunu her zaman bildiği durumlardır (örn. satranç oyununda ajanın tüm tahtayı ve taşların konumlarını görmesi).
  - **Kısmen Gözlemlenebilir Ortamlar (Partially Observable Environments)**: Ajanın herhangi bir zamanda ortamdaki durumların tamamını göremediği durumlardır (örn. poker oyununda rakibin elini görememek).

1.7 Araçlar ve Kütüphaneler

Reinforcement Learning uygulamaları geliştirmek için çeşitli kütüphaneler ve ortamlar mevcuttur:

- **OpenAI Gym (şimdiki adıyla Gymnasium)**: RL algoritmalarını test etmek için standart bir API ve çeşitli hazır ortamlar (environment) sunar (örn. Lunar Lander, CartPole, Taxi Driver, Mountain Car, Pendulum).
- **Stable Baselines (ve Stable Baselines3)**: PyTorch tabanlı (önceden TensorFlow) popüler RL algoritmalarının implementasyonlarını sunan bir kütüphanedir.
- **Keras/TensorFlow/PyTorch**: Derin öğrenme modelleri oluşturmak için kullanılan kütüphanelerdir, DRL uygulamalarında sinir ağları tasarlamak için entegre edilebilirler.
- **Pygame**: Python ile oyun geliştirme kütüphanesi olup, görsel RL ortamları oluşturmak için kullanılabilir.
- **Unity ML Agents**: Unity oyun motoru içinde yapay zeka geliştirme ve entegrasyonuna olanak tanır.
- **Gazebo**: Robotik simülasyon yazılımıdır.
- **Carla Simulator**: Otonom araçlar için gelişmiş bir simülasyon platformudur.

1.8 Örnekler

- **Taxi Driver Ortamı**: Ajan (taksi), bir harita üzerinde yolcu alır ve gitmek istediği yere bırakır. Ajan, her adımda ödül (yolcu hedefe ulaştığında +20), zaman kaybı için ceza (-1) veya yasa dışı eylemler için ceza (-10) alır. Bu ortam, ayrık ve deterministik bir yapıya sahiptir.
- **CartPole Ortamı**: Ajan, üzerinde bir çubuk olan bir arabayı dengelemeye çalışır. Amaç, çubuğun düşmesini engellemektir. Bu da bir dengeleme problemidir.

Reinforcement Learning, kendi kendine öğrenen sistemler tasarlamak için güçlü bir yaklaşım olup, gelecekte birçok karmaşık problemi çözme potansiyeline sahiptir.