Nüfus ve Sosyal Araştırmalar Enstitüsü - Yapay Zeka Modülü : Makine Öğrenmesi - 2

Bana herkes bu gördüklerimiz ne işime yarıyor diye sordu. Ben ise "sen ne hayal edebiliyorsun ?" diye cevapladım. *Dr. Oktay Altun*

0 - Neden Öğreniyoruz ?

0.1 Temel Geometriden Balistik Analize: İki Nokta Prensibi

Geometrinin en temel aksiyomlarından biri olan **"İki noktadan yalnızca tek bir doğru geçer"** kuralı, kağıt üzerinde öğrenilen en basit bilgilerden biridir. Ancak bu kural, adli bilimler ve askeri stratejilerde, ateş eden birinin (örneğin bir keskin nişancının) yerini tespit etmek gibi son derece komplike ve hayati sorunları çözer.

0.1.1 Teori (Basit Aşama)

Öklid geometrisine göre, uzayda rastgele belirlenmiş A ve B noktaları birleştirildiğinde elde edilen düz çizgi eşsizdir. Matematiksel olarak bu durum, sadece bir yönü ve doğrultuyu ifade eden sıradan bir soyutlamadır.

0.1.2 Pratik (Karmaşık Senaryo)

Bir suç mahallinde veya çatışma bölgesinde kaos hakimdir. Ancak bir mermi ateşlendiğinde ve doğrusal bir hızla ilerlerken iki farklı yüzeyi (örneğin iki ayrı pencere camını, duvarı veya bir aracın ön ile arka koltuğunu) delip geçtiğinde, bu delikler fiziksel "A" ve "B" noktalarına dönüşür.

0.1.3 Çözüm (Tersine Mühendislik ve Hedef Tespiti)

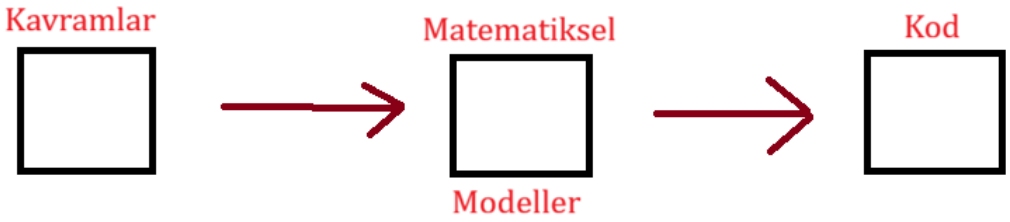
Balistik uzmanları bu kaotik ortamda basit geometri kuralını tersine bir mühendislikle uygular:

- **Hizalama:** Birbiri ardına sıralanan iki mermi deliğinden (A ve B noktaları) özel bir balistik çubuk veya lazer ışını geçirilir.
- **Yörünge Sabitlemesi:** İki nokta kuralı gereği, merminin geliş açısı başka hiçbir ihtimale yer bırakmayacak şekilde tek bir doğruya indirgenmiş olur.

- **Kökenin Bulunması:** Bu doğru çizgiyi (ışını) ateşin geldiği yöne doğru hayali olarak uzattığınızda, çizginin ucu doğrudan merminin çıktığı namluya, yani atıcının gizlendiği konuma işaret eder.

> **Özetle:** En karmaşık adli vakalar veya çatışma analizleri bile, "iki noktadan tek bir çizgi geçer" şeklindeki en temel, en basit matematiksel kuralın fiziksel dünyaya uyarlanmasıyla aydınlatılır.

Gökyüzünün mavi olmasının nedeni havadaki tozlar. Rayleigh saçılımı ile oluşur.



0.1.4 Python Neden Daha Az Karmaşık?

Alttaki "İnsan Dili - Makine Dili" yelpazesine baktığımızda, Python'ın insan diline en yakın uçta yer aldığını görüyoruz. Bu durum, onu teknik olarak "yüksek seviyeli" bir dil yapar ve karmaşıklığı şu şekillerde azaltır:

- **Günlük Dile Yakın Sözdizimi (Syntax):** Python kodları basit İngilizce kelimelerden oluşur. Karmaşık parantezler veya noktalı virgüller yerine sade, okuması tıpkı bir metin gibi akıcı bir yapı kullanır.
- **Detayları Arka Planda Halletme:** C++ veya makine dilinde sizin manuel olarak uğraşmanız gereken bellek yönetimi (hafıza tahsisi) gibi bilgisayarın donanımına ait karmaşık işleri, Python kendi kendine halleder.
- **Algoritmaya Odaklanma:** Bilgisayara "nasıl" çalışacağını adım adım anlatmakla vakit kaybetmezsiniz; doğrudan çözmek istediğiniz probleme ve işin mantığına odaklanırsınız.

Özetle: Python, bilgisayarın karmaşık donanım kurallarına göre değil; insanın doğal ve mantıksal düşünce yapısına daha uygun tasarlandığı için öğrenmesi ve yazması en kolay dillerden biridir.

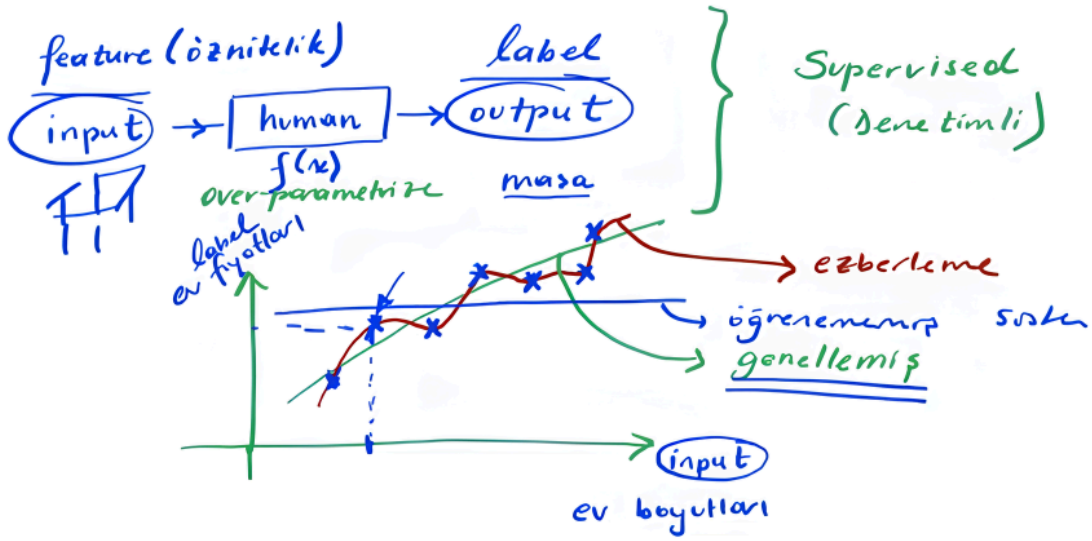
1 - Denetimli Öğrenme (Supervised Learning) ve Model Performansı

Yapay zekanın en yaygın kullanım şekillerinden biri olan **Denetimli Öğrenme**, algoritmaya hem girdilerin (verilerin) hem de doğru cevapların verilerek aradaki kuralın öğretilmesine dayanır.

1.1 Sistemin Temel Bileşenleri: Öznitelik ve Etiket

Öğrenme sürecinin gerçekleşmesi için veriler iki ana parçaya ayrılır:

- **Öznitelik (Feature / Input):** Sisteme verilen açıklayıcı bilgidir. Örneğin; bir evin metrekaresi, oda sayısı veya bulunduğu semt.
- **Etiket (Label / Output):** Bu özniteliklere karşılık gelen doğru sonuç veya hedeftir. Örneğin; o evin gerçek satış fiyatı. Sistem, özniteliklere bakarak etiketi tahmin etme örüntüsünü bulmaya çalışır.



1.2 Öğrenme Senaryoları (Ev Fiyatı Tahmini Örneği)

Yapay zeka veriler arasındaki ilişkiyi çözmeye çalışırken üç farklı tablo ortaya çıkabilir:

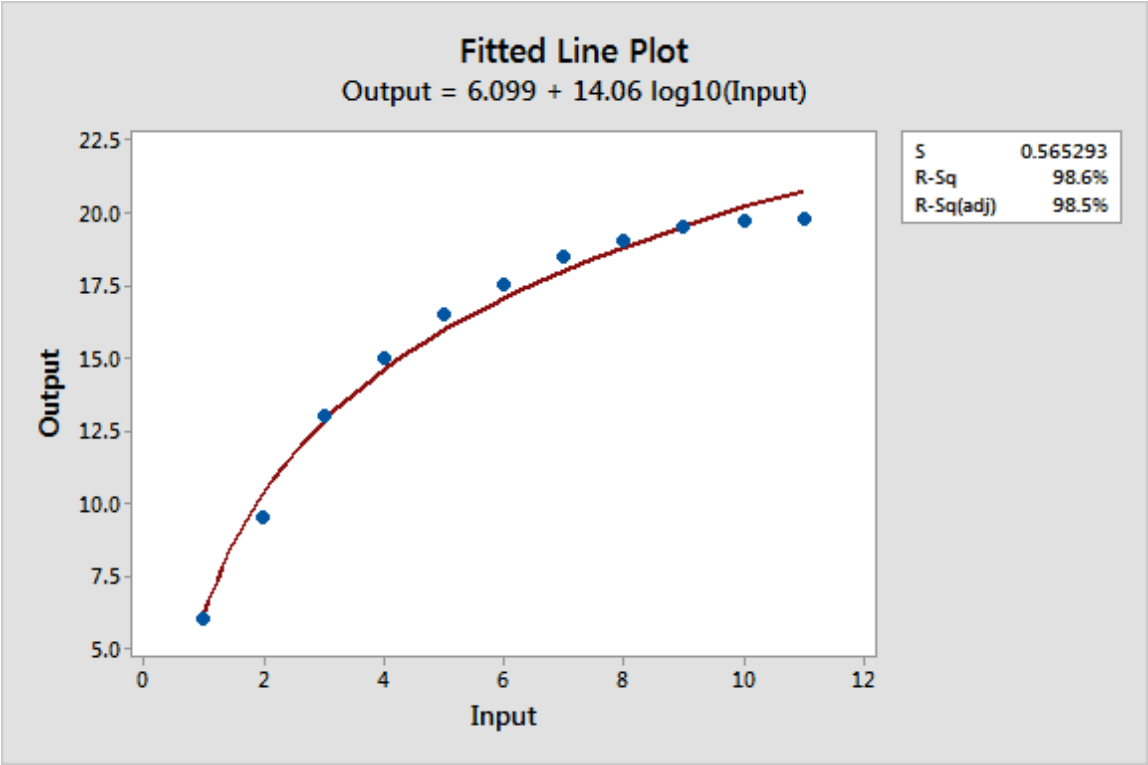
- **Öğrenememe (Underfitting):** Sistemin verilerdeki genel eğilimi veya mantığı hiç yakalayamaması durumudur. Örneğin, ev ne kadar büyürse büyüsün sistemin hep aynı sabit fiyatı tahmin etmesidir. Model, problemi çözecek kadar karmaşık değildir veya yeterince eğitilmemiştir.
- **Ezberleme (Overfitting):** Sistemin genel kuralı anlamak yerine, kendisine verilen eğitim verilerini noktası noktasına ezberlemesidir. Kendi elindeki verilerde kusursuz çalışır; ancak daha önce hiç görmediği yeni bir evin fiyatını tahmin etmesi istendiğinde tamamen mantıksız sonuçlar üretir.

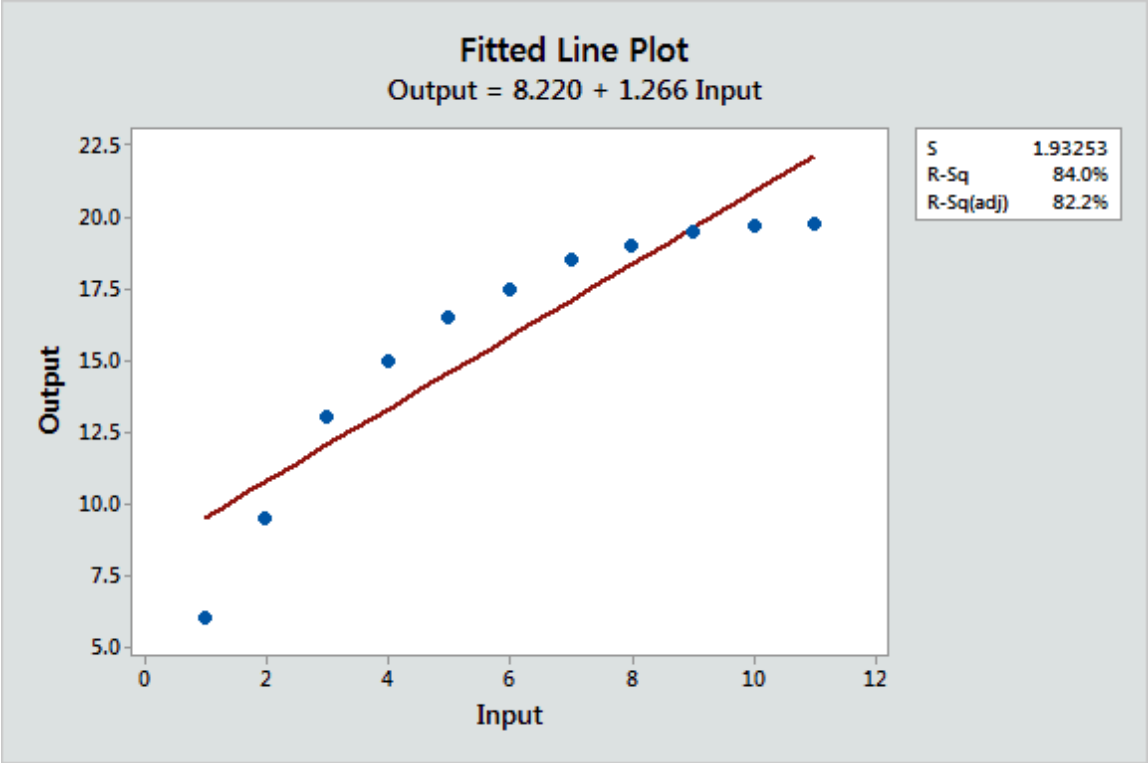
- **Genelleme (Generalization):** Hedeflenen en ideal durumdur. Model, istisnalara veya gereksiz detaylara takılmak yerine "ev büyüdükçe fiyat belli bir oranda artar" şeklindeki ana trendi (eğilimi) doğru bir şekilde kavramıştır. Bu sayede yeni ve bilinmeyen verilerle karşılaştığında en isabetli tahminleri yapar.

1.3 Eğri Uydurma (Curve Fitting) ve Gelişmiş Modeller (Transformer)

Ev fiyatlarını tahmin ederken veri noktaları arasına en uygun çizgiyi (genellemeyi) çizme işlemine matematiksel olarak **Eğri Uydurma (Curve Fitting)** denir. İlginç olan, bu mantığın günümüzdeki en gelişmiş yapay zeka sistemlerinin de temelini oluşturmasıdır.

- **Dizilim Tahmini:** Yapay zeka sadece sayısal değerleri tahmin etmekle kalmaz, aynı mantığı bir dizi veya metin üzerinde de uygular. Bir cümlede arka arkaya gelen kelimelere (örneğin `x1, x2, x3`) bakarak bir sonraki kelimeyi (`x4`) tahmin eder. Ardından bu yeni kelimeyi de dâhil ederek (`x1, x2, x3, x4`) sıradaki kelimeyi (`x5`) öngörür.
- **Transformer Mimarisi ve Denetimli Öğrenme:** Doğal dil işlemede (NLP) çığır açan **Transformer** modelleri tam olarak bu mekanizmayla çalışır. Sıradaki kelimeyi veya karakteri tahmin etme işi, temelde çok karmaşık ve devasa ölçekli bir **Denetimli Öğrenme** ve eğri uydurma sürecinden başka bir şey değildir.





2 - Gözetimsiz Öğrenme (Unsupervised Learning)

Makine öğrenmesinde **Gözetimsiz Öğrenme**, algoritmaya sadece ham verilerin (girdilerin) sağlandığı, ancak bu verilerin ne anlama geldiğini gösteren etiketlerin (çıktıların/doğru cevapların) bulunmadığı modelleme türüdür. Modelin amacı, dışarıdan bir yönlendirme

olmaksızın veri seti içindeki gizli yapıları, olasılık dağılımlarını ve doğal örüntüleri kendi başına keşfetmektir.

2.1 Çalışma Mantığı: Hedefsiz Keşif (Input $\rightarrow f(x)$)

Denetimli öğrenmede sistem girdilerden (X) belirli bir hedefe (Y) ulaşmayı öğrenirken, gözetimsiz öğrenmede tahmin edilecek bir Y (bağımlı değişken) yoktur. Sistem tamamen girdilerin (X) kendi içindeki matematiksel özelliklerine, varyanslarına ve birbirlerine olan uzaklık metriklerine odaklanır.

2.2 Temel Yaklaşımlar ve Kullanım Alanları

Etiketsiz veriyi anlamlandırmak için algoritma genellikle şu üç temel yöntemi kullanır:

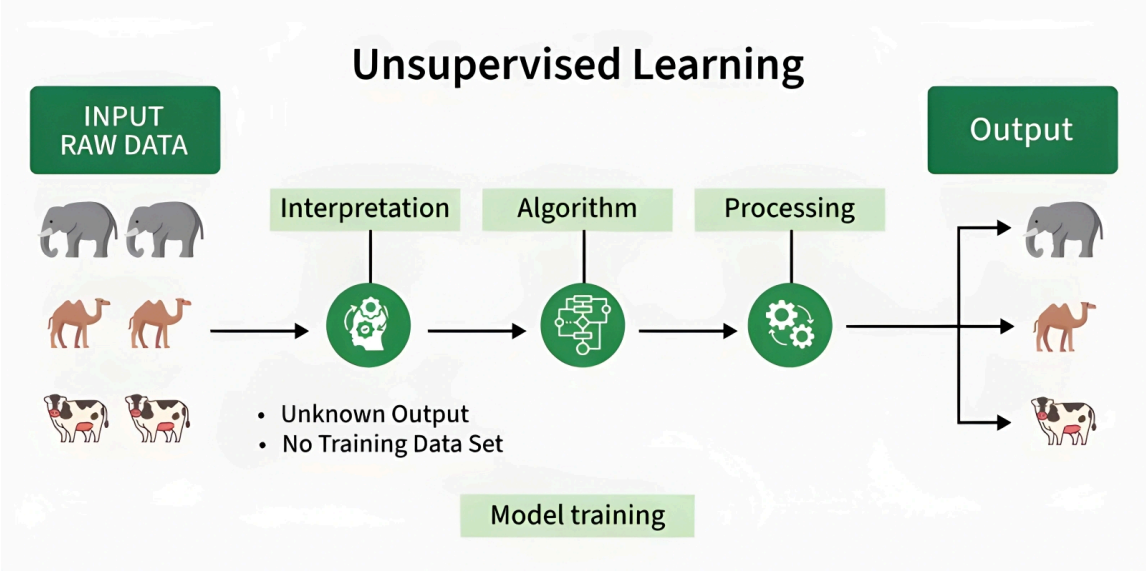
- **Kümeleme (Clustering):** Veri noktalarını birbirlerine olan benzerliklerine göre alt gruplara ayırma işlemidir. Algoritma, istatistiksel olarak birbirine yakın olan verileri aynı kümede toplarken, farklı özellikler gösterenleri ayırır. (Örn: Müşteri segmentasyonu).
- **Boyut İndirgeme (Dimensionality Reduction):** Çok fazla değişkene sahip (yüksek boyutlu) verilerdeki en önemli özellikleri, yani varyansı en çok açıklayan temel bileşenleri koruyarak veri setini sadeleştirme işlemidir. Veriyi görselleştirmeyi ve işlem yükünü hafifletmeyi sağlar.
- **Anomali Tespiti (Anomaly Detection):** Genel veri dağılımına uymayan, standart sapmanın çok dışında kalan aykırı değerleri (outliers) tespit etmektir. (Örn: Kredi kartı sahtekarlıklarının veya sistemdeki siber saldırıların tespiti).

2.3 Örnek: Etiketsiz Veriden Anlam Çıkarmak

Elimizde 1 milyon satırlık devasa bir hayvan veri seti olduğunu düşünelim. Ancak elimizdeki tek bilgi bu hayvanların **"kilo"** ve **"boy"** ölçümleri olsun. Hangi satırın kediye, hangisinin köpeğe ait olduğu bilgisi (etiket) veride **yoktur**.

- **Uzaysal Dağılım:** Model, bu binlerce boy ve kilo verisini iki boyutlu bir koordinat düzlemine (veya daha fazla öznitelik varsa çok boyutlu bir uzaya) yerleştirir.
- **Matematiksel Gruplama:** Veriler üzerindeki yoğunlukları ve noktalar arası mesafeleri (Örn: Öklid uzaklığı) analiz eden algoritma, verilerin rastgele dağılmadığını, aksine iki farklı yoğunluk merkezi etrafında kümelendiğini tespit eder.
- **Sınıflandırma Sınırı:** Sistem, bu iki yoğunluk dağılımının arasına matematiksel bir sınır çizgisi çeker. Veriyi kendi içinde "A Kümesi" ve "B Kümesi" olarak ikiye ayırır.
- **İnsan Yorumu:** Algoritma veriye bakıp "A kümesindeki noktalar genelde daha ağır ve uzun, B kümesindekiler daha hafif ve kısa" şeklinde bir ayrım yapar. Daha sonra bu kümeleri inceleyen bir kişi, A kümesinin "Köpekleri", B kümesinin ise "Kedileri" temsil ettiğini anlar ve veriyi anlamlandırmış olur.

Özetle: Gözetimsiz öğrenme; karmaşık, dağınık ve etiketsiz veri yığınlarının altındaki istatistiksel doğayı ve yapısal mimariyi ortaya çıkarmak için kullanılan en temel keşif aracıdır.



3 - Pekiştirmeli Öğrenme (Reinforcement Learning)

Makine öğrenmesinin üçüncü büyük dalı olan **Pekiştirmeli Öğrenme (RL)**, bir ajanın (yapay zeka modelinin) karmaşık bir ortamda kendi eylemleri sonucunda elde ettiği geri bildirimlerle öğrenmesi sürecidir. Bu modelde önceden verilmiş bir veri seti veya doğru cevaplar (etiketler) yoktur; sistem çevresiyle etkileşime girerek tecrübe kazanır.

3.1.1 Temel Öğrenme Mekanizmaları

Pekiştirmeli öğrenme, insan ve hayvan psikolojisindeki davranışsal öğrenmeye çok benzer bir mantıkla çalışır:

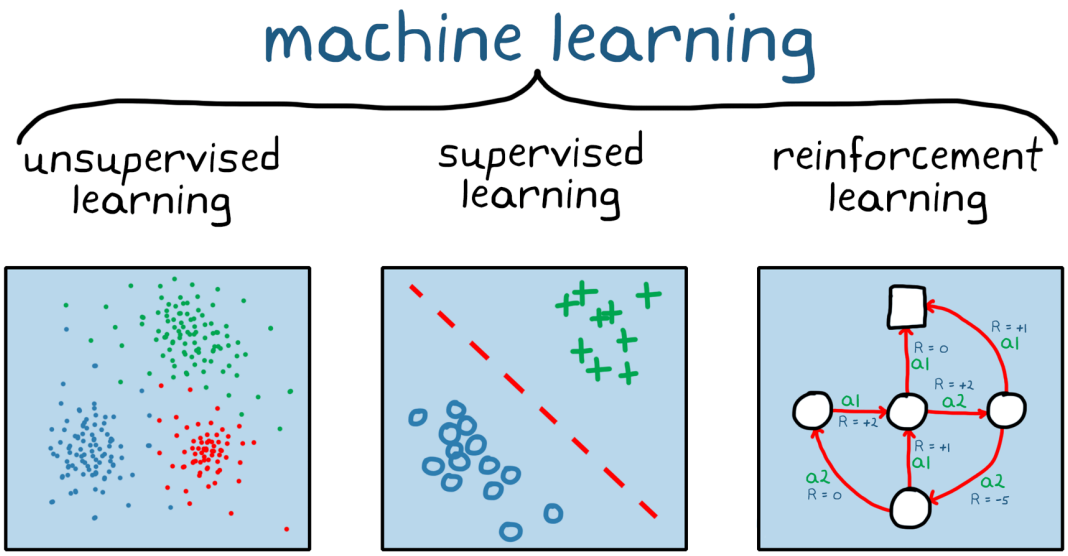
- **Deneme-Yanılma (Trial and Error):** Model, içinde bulunduğu ortamda rastgele veya hesaplanmış hamleler yapar. Her hamlesinin sonucunda ortamın nasıl değiştiğini gözlemleyerek neyin işe yarayıp neyin yaramadığını keşfeder.
- **Ödül-Ceza Sistemi (Reward-Punishment):** Modelin yaptığı hamleler amaca hizmet ediyorsa sisteme matematiksel bir "ödül" (artı puan) verilir. Eğer hamle başarısızlığa veya hataya yol açıyorsa "ceza" (eksi puan) verilir. Sistemin tek amacı, toplam ödül miktarını maksimize etmektir.

3.1.2 Gecikmeli Ödül (Delayed Reward) Kavramı

Pekiştirmeli öğrenmenin en zorlayıcı ve karakteristik özelliklerinden biri **gecikmeli ödül veya ceza** durumlarıdır.

- Modelin yaptığı bir hamlenin iyi mi yoksa kötü mü olduğu anında belli olmayabilir.

- Örneğin; bir satranç oyununda yapılan stratejik bir piyon fedası (o an için eksi puan/ceza gibi görünse de), 10 hamle sonra oyunu kazandıran (büyük ödül) asıl kilit hamle olabilir. Model, anlık kazançlardan ziyade uzun vadeli başarıyı hesaplamayı öğrenmelidir.



3.2.1 Makine Öğrenmesi Paradigmalarının Kesişimi: Yarı Denetimli Öğrenme

Makine öğrenmesi dünyası keskin sınırlarla birbirinden ayrılmaz. Denetimli (SL), Gözetimsiz (UL) ve Pekiştirmeli (RL) öğrenme kümelerinin kesiştiği özel alanlar vardır. Bunlardan en önemlisi **Yarı Denetimli Öğrenmedir (Semi-Supervised Learning)**.

- **Yarı Denetimli Öğrenme (Semi-Supervised):** Gözetimsiz öğrenme (UL) ile Denetimli öğrenmenin (SL) kesişim noktasıdır.
- **Neden İhtiyaç Duyulur?** Gerçek hayatta verileri insanlar tarafından tek tek etiketlemek (Denetimli Öğrenme için) çok maliyetli ve zaman alıcıdır. Öte yandan tamamen etiketsiz veriden (Gözetimsiz Öğrenme) anlam çıkarmak her zaman hedeflenen spesifik sonucu vermeyebilir.
- **Nasıl Çalışır?** Elimizde çok az sayıda etiketlenmiş (doğru cevabı bilinen) veri ve devasa boyutta etiketlenmemiş ham veri bulunur. Sistem, önce o küçük etiketli veriden genel bir kural öğrenir, sonra bu kuralı kullanarak devasa boyuttaki etiketsiz veriyi anlamlandırır ve kendi kendini eğitmeye devam eder.

3.2.1.1 Pekiştirmeli Öğrenmenin Etkileşim Döngüsü ve Matematiksel Temeli

Pekiştirmeli öğrenme, bir yapay zeka modelinin çevresiyle girdiği sürekli bir etkileşim döngüsü üzerinden formülize edilir. **1. Etkileşim Bileşenleri (a_t, s_t, r_t)** Sistem, bir **Ajan (Agent)** ve onun içinde bulunduğu **Çevre (Environment)** arasında geçen üç temel değişken üzerinden işler:

- a_t **(Action/Eylem):** Ajanın t anında yaptığı hamle.
- s_t **(State/Durum):** Çevrenin bu hamleye verdiği tepki ve ajanın geçtiği yeni durum.
- r_t **(Reward/Ödül):** Bu durum değişikliği sonucunda ajanın aldığı puan (ödül veya ceza).

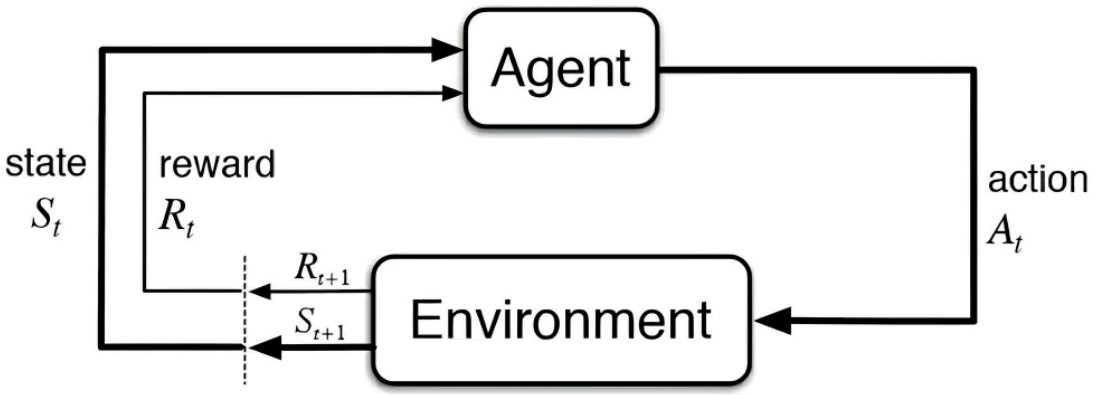
2. Çok Adımlı Öğrenme Süreci Pekiştirmeli öğrenme anlık bir tahmin değil, **"çok adımlı" (multi-step)** bir süreçtir. Ajanın zaman içindeki tecrübeleri şu şekilde kaydedilir:

- $\langle a_1, s_1, r_1 \rangle$
- $\langle a_2, s_2, r_2 \rangle$
- $\langle a_3, s_3, r_3 \rangle$ ...

Sistem bu uzun dizilere (trajectories) bakarak **gecikmeli ödül** kavramını öğrenir. Yani 1. adımdaki bir hamlenin asıl ödülünün veya cezasının 10. adımda ortaya çıkabileceğini kavrar.

3. Nihai Hedef: Optimal Politika (π^*) Ajanın tüm bu deneme-yanılma sürecinin sonunda elde etmek istediği nihai bir kılavuz veya kural seti vardır. Buna literatürde **Politika (Policy - π)** denir. Hedefi her zaman en yüksek ödülü getirecek olan **Optimal Politikayı (π^*)** bulmaktır.

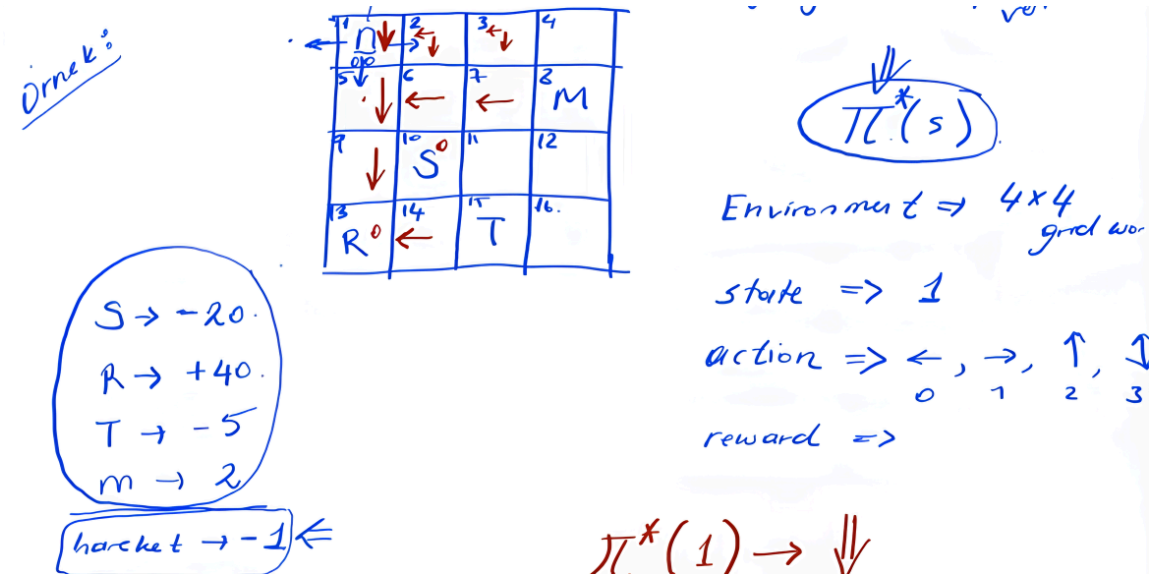
- **Örnek Kurulum:** Algoritma π^* fonksiyonunu oluşturduğunda, mevcut durum (S) bir **"kavşak"** olarak algılandığında, model bu mükemmel politikayı kullanarak otomatik olarak en yüksek ödülü getirecek olan **"düz git"** kararını (aksiyonunu) üretir.



Keyifli bir kaynak : https://au.mathworks.com/discovery/reinforcement-learning.html

3.2.2 Pekiştirmeli Öğrenme Uygulaması: Grid World (Izgara Dünyası) Örneği

Pekiştirmeli öğrenmedeki etkileşim döngüsünü somutlaştırmak için en sık kullanılan örneklerden biri "Grid World" modelidir. Bu modelde ajan, labirent benzeri bir ızgara üzerinde hareket ederek hedefe ulaşmaya çalışır.



3.2.2.1 Çevre ve Durum (Environment & State)

- **Çevre (Environment):** Örnekte 4x4 boyutlarında, toplam 16 kareden/hücreden oluşan kapalı bir dünya (grid world) tanımlanmıştır.
- **Durum (State - s):** Ajanın o an bulunduğu kareyi ifade eder. Görseldeki örnekte ajan başlangıç olarak sol üst köşede, yani **1 numaralı durumdadır (State = 1)**.

3.2.2.2 Eylemler (Actions - a)

Ajanın bu çevre içinde yapabileceği hamle kapasitesidir. Ajan 4 farklı yöne gidebilir ve algoritmada bu yönler genellikle matematiksel/sayısal olarak ifade edilir:

- `0` $\rightarrow$ Sol (Left)
- `1` $\rightarrow$ Sağ (Right)
- `2` $\rightarrow$ Yukarı (Up)
- `3` $\rightarrow$ Aşağı (Down)

3.2.2.3 Ödül Sistemi ve Hiperparametreler (Rewards & Hyperparameters)

Ajanın rastgele gezinmek yerine mantıklı bir politika ($\pi^*(s)$) geliştirebilmesi için çevredeki belirli karelere ödül ve cezalar atanmıştır. Sistemi tasarlayan kişi tarafından dışarıdan belirlenen bu başlangıç ayarlarına **Hiperparametre (Hyperparameter)** denir.

Örnekteki ödül haritası şu şekildedir:

- **R (Reward): +40 Puan.** Ajanın ulaşmaya çalıştığı nihai ve en büyük hedeftir (Örn: 13. kare).
- **M (Minor Reward): +2 Puan.** Yolda alınabilecek küçük bir bonustur (Örn: 8. kare).
- **T (Trap/Minor Penalty): -5 Puan.** Ajanın kaçınması gereken küçük bir engeldir (Örn: 15. kare).

- **S (Severe Penalty): -20 Puan.** Ajanın kesinlikle düşmemesi gereken büyük bir tuzaktır (Örn: 10. kare).

Kritik Kural: Hareket Maliyeti (Living Penalty)

- **Hareket $\rightarrow$ -1 Puan:** Bu çok önemli bir hiperparametredir. Ajanın attığı her adımda hanesinden 1 puan düşülür.
- **Amacı:** Eğer bu ceza olmasaydı, ajan +40 puanlık hedefe ulaşmak için acele etmez, haritada sonsuza kadar güvenli karelerde gezinebilirdi. Adım başı -1 puan cezası verilmesi, ajanı **"en kısa ve en güvenli yoldan"** hedefe gitmeye zorlar.

> **Özetle:** Ajanın buradaki nihai amacı (Optimal Politika - $\pi^*(s)$), 1. kareden başlayıp -20'lik tuzağa düşmeden ve olabildiğince az adım atarak (hareket cezasını minimize ederek) +40'lık hedefe ulaşacağı rotayı keşfetmektir.

3.2.3 Pekiştirmeli Öğrenmede Bölümler (Episodes), Toplam Ödül ve Karar Tablosu

Ajanın ortamdaki öğrenme süreci anlık tek bir karardan ibaret değildir; baştan sona defalarca tekrarlanan "oyunlar" dizisi şeklinde ilerler.

3.2.3.1 Bölüm (Episode) Kavramı ve Deneyim Dizisi

Ajanın başlangıç noktasından (State 1) harekete geçip, oyunu tamamen bitiren terminal bir duruma (büyük ödül `R` veya büyük tuzak `S` gibi) ulaşana kadar geçen sürecin tamamına bir **Episode (Bölüm)** denir. Ajan hedefe ulaşsa da tuzağa düşse de o oyun (bölüm) biter, sistem sıfırlanır ve yeni bir bölüm başlar.

Başarısız Bir Bölüm Örneği (Trajectory): Ajanın tuzağa düştüğü kısa bir senaryoyu adım adım ($\langle \text{Durum}, \text{Aksiyon}, \text{Ödül} \rangle$) yazarsak:

- **Adım 1:** $\langle 1, \rightarrow, -1 \rangle$ (1. kareden sağa gitti, hareket maliyeti olarak -1 aldı)
- **Adım 2:** $\langle 2, \downarrow, -1 \rangle$ (2. kareden aşağı indi, -1 aldı)
- **Adım 3:** $\langle 6, \downarrow, -1 \rangle$ (6. kareden aşağı indi, -1 aldı)
- **Adım 4:** $\langle 10, \text{Tuzak}, -20 \rangle$ (10. karedeki 'S' tuzağına düştü, -20 ceza yedi ve bölüm bitti)
-

3.2.3.2 Toplam Ödül Muhasebesi (Aggregate Reward)

Pekiştirmeli öğrenmenin matematiksel temelindeki en kritik kurallardan biri şudur: **"Ödül muhasebesi 'aggregate' (kümülatif/toplam) ödül üzerinden yapılır."** Ajan, her adımda aldığı anlık -1 cezalara takılıp kalmaz; bölümün (episode) sonunda elinde kalan toplam değere (getiriye) odaklanır. Yukarıdaki başarısız senaryoda ajanın bu bölümden aldığı toplam

getiri **-23**'tür. Ajanın nihai amacı, milyonlarca bölüm oynayarak bu toplam değeri en yüksek (pozitif) seviyeye çıkaracak formülü bulmaktır.

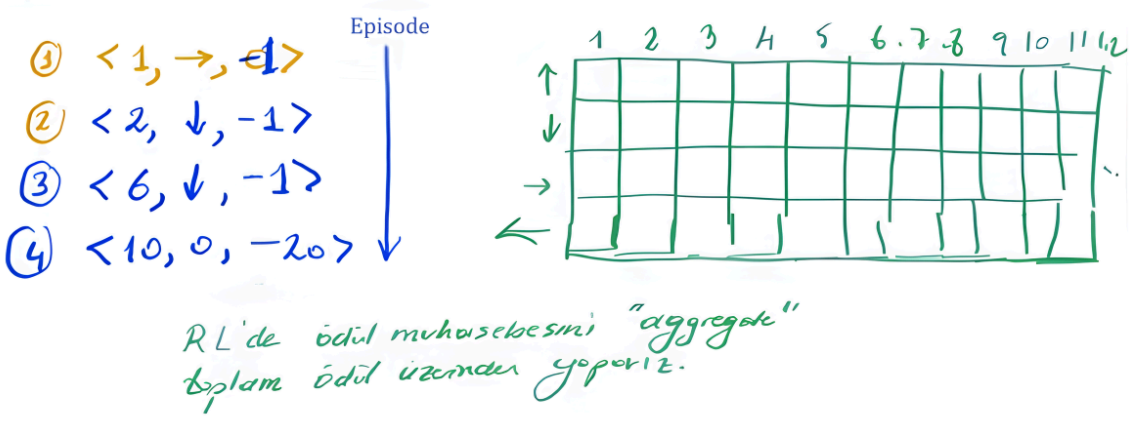
3.2.3.3 Optimal Politika ($\pi^*(s)$) ve Yönlendirme Okları

Ajan binlerce kez yanılıp farklı yolları denedikten sonra, haritadaki her bir kare için en kârlı hareketin ne olduğunu bulur. Haritadaki karelerin içine yerleşen "yön okları", ajanın bulduğu o kusursuz kılavuzu, yani **Optimal Politikayı** temsil eder. Öğrenme tamamlandıktan sonra ajan nerede olursa olsun düşünmez veya hesap yapmaz; sadece bulunduğu karedeki oku takip ederek en yüksek ödüllü rotadan hedefe ulaşır.

3.2.3.4 Öğrenme Hafızası: Durum-Eylem Matrisi (Q-Tablosu)

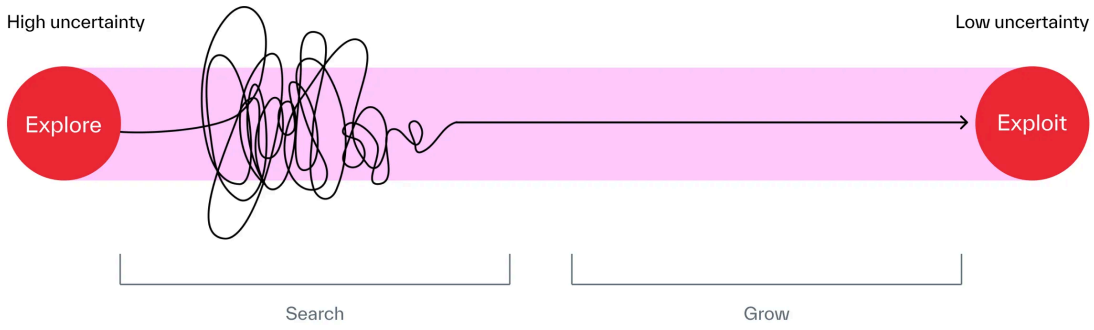
Ajanın bulduğu bu "okları" (en iyi hamleleri) aklında tutması ve öğrenme aşamasında puanları kaydetmesi gerekir. Bunun için satır ve sütunlardan oluşan boş bir matris (tablo) yaratır:

- **Sütunlar (1, 2, 3... 16):** Ajanın bulunabileceği tüm durumları (Kareleri) gösterir.
- **Satırlar ($\uparrow, \downarrow, \leftarrow, \rightarrow$):** O durumda yapılabilecek tüm eylemleri (Yönleri) gösterir.
- Ajan her denemesinde bu tablonun içindeki hücrelere puanlar (Q-değerleri) yazar ve her bölümde bu puanları günceller. Öğrenme bittiğinde, her sütundaki en yüksek puanlı satır, haritadaki o meşhur nihai yön oklarına dönüşür.



4 - Pekiştirmeli Öğrenmede Karar Verme Stratejisi: Keşif ve Sömürü İkilemi (Exploration vs. Exploitation)

Pekiştirmeli öğrenmede (Reinforcement Learning) bir ajan (agent), çevresiyle etkileşime girerken sürekli olarak şu zorlu kararı vermek zorundadır: **"Yeni ve bilinmeyen bir şey mi denemeliyim, yoksa şu ana kadar bildiğim en iyi yolu mu kullanmalıyım?"** Öğrenme süreci ve modelin ömrü, bu iki temel fazın hassas dengesine dayanır.



4.1 Keşif (Explore / Search) → "Rastgele" Hamleler

- **Durum (Yüksek Belirsizlik):** Ajanın çevreyi henüz tam olarak tanımadığı, hangi hamlenin ne kadar ödül veya ceza getireceğini bilmediği aşamadır.
- **Hareket Mantığı:** Ajan karar verirken, anlık yüksek puanları veya bildiği kısa yolları göz ardı ederek bilinçli olarak **rastgele** hamleler yapar. Görselleştirmelerde bu faz, ajanın haritada rastgele dolaştığı karmaşık ve dolambaçlı bir arayış çizgisi ile temsil edilir.
- **Amaç:** Anlık ödülleri feda ederek ortam hakkında bilgi toplamak, haritadaki farklı ihtimalleri görmek ve gizli kalmış, potansiyel olarak çok daha büyük ödülleri veya kestirme yolları bulmaktır.

4.2 Sömürü / Mevcut Bilgiyi Kullanma (Exploit / Grow) → "Geçmiş Deneyimden Yararlanma"

- **Durum (Düşük Belirsizlik):** Ajanın yeterince deneme-yanılma yaptığı, çevrenin kurallarını (örn. Q-Tablosunu) ve ödül dağılımını büyük ölçüde öğrendiği aşamadır.
- **Hareket Mantığı:** Ajan artık rastgele maceraya atılmayı bırakır. O zamana kadar öğrendiği en kârlı, en güvenli ve puanı en yüksek hamleyi seçerek geçmiş deneyimlerinden yararlanır. Bu aşama, dümdüz ve emin adımlarla ilerleyen bir çizgi ile temsil edilir.
- **Amaç:** Elde edilen mevcut bilgiyi doğrudan kullanarak toplam ödülü maksimize etmek (büyüme/grow) ve doğrudan hedefe gitmektir.

> **Kritik Denge Neden Önemlidir?** Eğer ajan **sadece keşfederse** (sürekli Explore), hiçbir zaman hedefe istikrarlı şekilde ulaşıp yüksek puan toplayamaz; haritada sürekli rastgele dolanır durur. ZZ Ancak ajan **hemen sömürüye geçerse** (erken Exploit), bulduğu ilk ortalama yola saplanıp kalır (**Local Optima**) ve belki de haritanın diğer ucundaki asıl büyük ödülü (**Global Optima**) sonsuza dek kaçırır.
>
> **Başarılı bir modelin stratejisi:** Eğitime yüksek oranda rastgele keşifle başlar ve ortamı tanıdıkça zamanla geçmiş deneyimleri kullanmaya (sömürüye) doğru kayarak bu dengeyi kusursuzca kurar.

4.3 Bu Denge Teknik Olarak Nasıl Kurulur? (Epsilon-Greedy Stratejisi)

Pratikte modellerin bu geçişi pürüzsüz yapabilmesi için çoğunlukla **Epsilon-Greedy (ϵ-greedy)** adı verilen bir algoritma yaklaşımı kullanılır.

- **Epsilon (ϵ) Değeri:** Ajanın karar anında "rastgele" hareket etme (keşif) olasılığını temsil eder.
  - Örneğin $\epsilon = 0.9$ ise; ajan %90 ihtimalle rastgele bir hamle yapar (Explore), %10 ihtimalle ise Q-Tablosundaki en yüksek puanlı hamleyi seçer (Exploit).
- **Epsilon Decay (Epsilon'un Azaltılması):** Modelin eğitiminin en başında ϵ değeri kasıtlı olarak çok yüksek tutulur (örn. 1.0). Ajan çevreyi deneyimledikçe ve eğitim adımları (episodes) ilerledikçe, bu değer algoritmik olarak yavaş yavaş düşürülür (örn. minimum 0.01'e kadar).
- **Sonuç:** Bu sayede ajan hayata "maceraperest" olarak başlar, ancak yaşlandıkça ve öğrendikçe "garantici" bir yapıya bürünerek mevcut bilgisini sömürür.

-Memreu