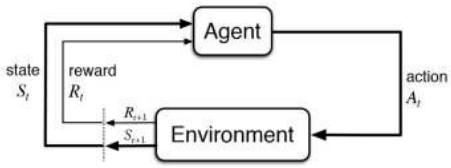
Makine Öğrenmesi-3

Akış

- Makine Öğrenmesi nedir?
- Günlük Hayatımızdaki Uygulamaları
- Verilerin Sayısallaştırılması
- Özellik Belirleme
  - Özellik Seçim Metotları
    - Bilgi Kazancı (Informaiton Gain-IG)
    - Sinyalin Gürültüye Oranı: (S2N ratio)
    - Alt küme seçiciler (Wrappers)
  - Yeni Özelliklerin Çıkarımı
    - Temel Bileşen Analizi (Principal Component Analysis)
    - Doğrusal Ayırteden Analizi (Linear Discriminant Analysis)
- Sınıflandırma Metotları
  - Doğrusal Regresyon
  - Karar Ağaçları (Decision Trees)
  - Yapay Sinir Ağları
  - En Yakın K Komşu Algoritması (k - Nearest Neighbor)
  - Öğrenmeli Vektör Kuantalama (Learning Vector Quantization)
- Kümeleme Algoritmaları:
  - Hiyerarşik Kümeleme
  - K-means
  - Kendi Kendini Düzenleyen Haritalar (Self Organizing Map -SOM)
  - DBscan
- Regresyon Algoritmaları
- Çok Boyutlu Verilerle Çalışmak
- Veri Sızıntısı
- **Pekiştirmeli Öğrenme**

Pekiştirmeli Öğrenme
(Destekleyici / Takviyeli)

- Ödül/ceza ile eğitim
- Ödül genelde uzakta
- Hangi durumda (S) hangi hareketin (A) yapılacağı öğrenilir.
- Ne yapması gerektiğini söylemeyiz. Ödül/ceza veririz sadece.
- Ajan, toplam ödülü maksimize etmeye çalışır
- Ajanın hareketleri hangi verilere erişeceğini belirler



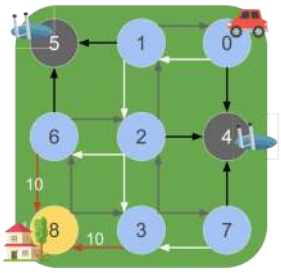
Pekiştirmeli Öğrenme
Reinforcement Learning

- Olası tüm durumların erişilebilir olduğu simülasyon dünyalarda (oyunlar) iyi çalışır.
- Oyunlarda kendi kendine eğitim de mümkün. Simülasyon dünya, öğrenen sisteme oyunu kazandın/kaybettin/ şu kadar puan aldın vb. diyebilir.

Temel Kavramlar

- **Environment / ortam**: ajanın içinde bulunduğu ortam
- **State / durum**: mevcut durum
- **Reward / ödül**: ortamdan gelen geri besleme
- **Policy / politika**: ajanın durumlarını hareketlerine eşleyen tablo
- **Value / gelecek ödül**: şu durumda şu hareketi yapınca alınacak gelecek ödül

Eve gidelim, Ama nasıl



- 0(başlangıç)-8(ev) arası toplam 9 durum
- Her durumda yapılabilecek 4 olası hareket
- Ödüller sadece 8'e giden yerlerde (10)

- Q: Satırlarında durumlar, sütunlarında hareketler olan bir matris
Başlangıçta tüm değerleri 0,
zaman içinde değişecek

[*] https://towardsdatascience.com/practical-reinforcement-learning-02-getting-started-with-q-learning-582f63e4acd9

t=0

	UP	DOWN	LEFT	RIGHT
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

t=k

	UP	DOWN	LEFT	RIGHT
0	0	0	0	0
1	0	0	0	0
2	0	2.25	2.25	0
3	0	0	5	0
4	0	0	0	0
5	0	0	0	0
6	0	5	0	0
7	0	0	2.25	0
8	0	0	0	0

t=∞

	UP	DOWN	LEFT	RIGHT
0	0	0	0.45	0
1	0	1.01	0	0
2	0	2.25	2.25	0
3	0	0	5	0
4	0	0	0	0
5	0	0	0	0
6	0	5	0	0
7	0	0	2.25	0
8	0	0	0	0

Q learning

- Episode/Bölüm: Başlangıçtan başlayan, fail ya da hedefle biten hareket dizisi
- Policy / politika: mevcut durumda (s), yapılacak hareketi (a), Q'ya göre seçer
- Ajan, iyi bir politika (Q) arar. Deneme yanılma ile. Bir politika ile başlar, onu iyileştirir. İyileştirmede ikilem: keşfetmek / kullanmak (exploration vs. exploitation)

keşfetmek / kullanmak (exploration vs. exploitation)

- Kullanmak: Şu anda en iyi görünen hareketi denemek. Hedefe daha az maliyetle / daha çok puanla ulaşmamızı sağlamaya çalışır.
- Keşfetmek: Şu anda en iyi olmayan bir hareketi denemek. Ortamı daha iyi keşfetmemizi sağlar. Yeni durumlara erişir. Uzun vadede iyileşme sağlayabilir.

Keşfetmek / Kullanmak

- Acıktınız: yeni bir yeri denemek / iyi bildiğiniz bir yere gitmek
- Petrol arıyorsunuz: daha önce petrol çıkmış bölgede çalışmak / yeni bir bölgede çalışmak
- Simulated annealing'i hatırlayın (rastgele bir hareket seç, iyiyse uygula, değilse azalan bir olasılıkla uygula)

Q-learning Algoritması

Her bölüm için tekrar et
- Başlangıç durumuna git ($s=s_0$)
- s bölüm sonu olmadığı sürece
  - politikaya göre hareket (a) seç
  - hareketi yap, ödülü (R) ve sonraki durumu (s') belirle
  - $Q[s, a] = Q[s, a] + \alpha*(R + \gamma*\max[Q(s', A)] - Q[s, a])$
  - s=s'

max [Q(s', A)] : s' durumunda yapılabilecek tüm hareketlerin ödüllerinin maksimumu

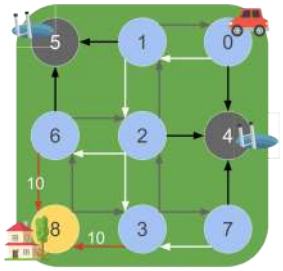
Politikaya göre hareket seçimi: keşfet / kullan ikilemi:
- Q[s,:] lerin en yükseğini seçmek→ kullan,
- rasgele birini seç → keşfet

decay ε-greedy: rastgele hareket seçme olasılığı giderek azalır, simulated annealing deki sıcaklık gibi.

α: öğrenme katsayısı

γ: (0,1) arası, 0'a yakın: yakın ödüllere, 1'e yakın: gelecek ödüllere odaklanır

Örnek Q güncellemeleri



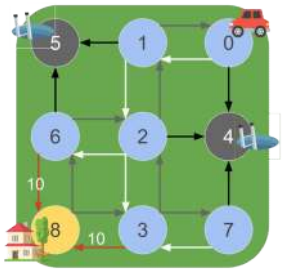
- $Q[s, a] = Q[s, a] + \alpha*(R + \gamma*\max[Q(s', A)] - Q[s, a])$
- $\alpha=0.5$ $\gamma=0.9$

İlk bölümler (durum 3 teyim, Left seçtim)

Q[3,L] = Q[3,L]+0.5(10+0.9*Max[Q(8,U),Q(8,D),Q(8,R),Q(8,L)]-Q(3,L))*
*Q[3,L] = 0 + 0.5 * (10 + 0.9 * Max [0, 0, 0, 0] -0)*
Q[3,L] = 5, benzer olarak Q[6,D] = 5

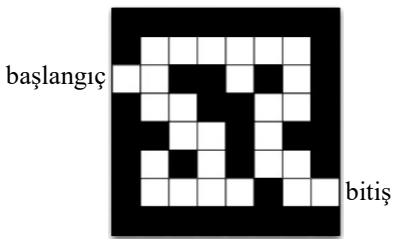
Sonraki bölümler (durum 2 deyim, Left seçtim)

Q[2,L] = Q[2,L]+0.5(0+0.9*Max[Q(6,U),Q(6,D),Q(6,R),Q(6,L)]-Q(2,L))*
*Q[2,L] = 0 + 0.5 * (0 + 0.9 * Max [0, 5, 0, 0] -0)*
Q[2,L] = 2.25, benzer olarak Q[2,D] = 2.25 ve Q[7,L] = 2.25



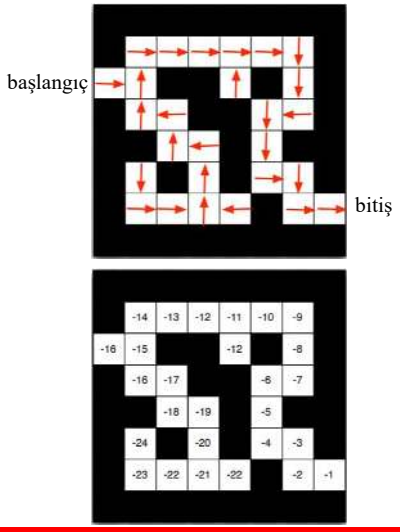
- Çukurlardan uzak durmak istersek?

Labirentten çıkış



- State / durum: ajanın konumu
- Actions / hareketler: her durumda 4 yön
- Reward / ödül: her adımda -1 (Birden fazla yol varsa ve en kısa yolu bulmak istersek)
- Policy / politika: her durumda en fazla ödülü kazandıran hareket
- Value / gelecek ödül

Öğrenmemiz gereken



Oklar: s durumda en iyi value (gelecek ödül) ya sahip olan hareket

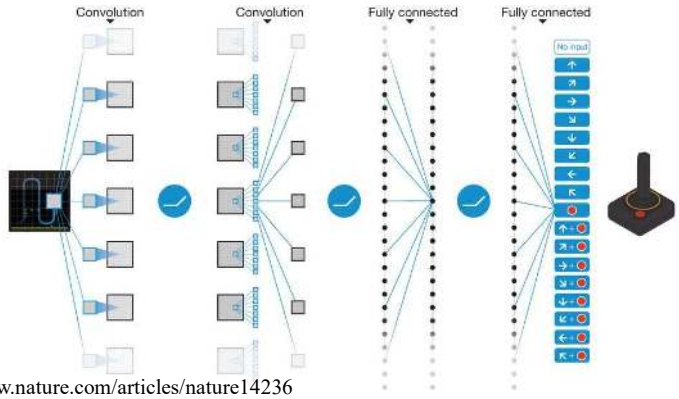
Gelecek ödüller

Yeni Durumlar

- Durum sayısı en başta bilinebilir (sabit boyutlu Q matrisi).
- Zaman içinde yeni durumlar (Q 'ya yeni satırlar) eklenebilir.
- Q-learning daha önce görmediği bir durumun değerini (value) bilemez, dolayısıyla yeni durumlara adapte olamaz
- Eğitim sürecinde görülmeyen (yeni) durum sayısı oyunlarda çok fazladır.
- Olası çözüm: yeni durumu eldekilerden birine benzet

Atari oyunları / deep Q-network

- Renkli bir ekranda olası durum sayısı ???
- Resmi (durumu) CNN'lerle işle, Q(s,a) ları / hareketi belirle
- Tek bir an yeter mi?

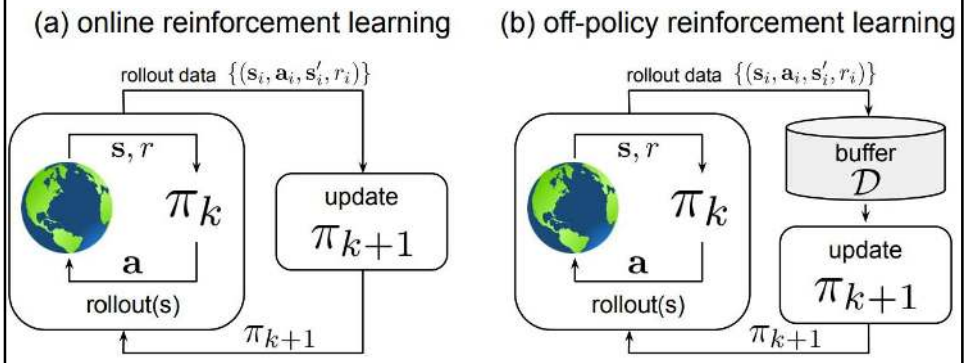


[*] https://www.nature.com/articles/nature14236

On / off policy / offline RL*

- Ajan, durumu (s_t) gözler, buna göre policy $\boldsymbol{\pi_\theta(a|s)}$ i kullanıp hareket (a) seçer. Bunun sonucunda ödül (r) alır ve sonraki duruma (s_{t+1}) geçer.
- Policy: durumu harekete dönüştüren bir fonksiyon (amaç bunu optimize etmek)
- Toplanan deneyimlerin formatı : **<s,a,s',r>**
- Bu deneyimlerle policy eğitilir.
- Yöntemlerin farkı, deneyimlerin üretim süreçleri

[*] https://arxiv.org/pdf/2005.01643.pdf



- A* ortamdaki harici reward la birleştirilebilir mi?
- Discrite / stochastic action/state
- Multi agent collobration / competetive
- Human imitation (sparse reward games)
- Inverse RL (psedou rewards)
- Policy gradient methods (PPO vb.)
- Bir oyun olarak diyalog: RLHF
- ???

Kaynaklar ve ek okumalar

- https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/DP.pdf
- http://mnemstudio.org/path-finding-q-learning-tutorial.htm
- https://medium.freecodecamp.org/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc
- https://medium.freecodecamp.org/a-brief-introduction-to-reinforcement-learning-7799af5840db
- https://towardsdatascience.com/dqn-part-1-vanilla-deep-q-networks-6eb4a00febfb