Q-learning ile Labirentte Yol Bulma

1 Giriş

Pekiştirmeli öğrenme (reinforcement learning), öznelerin (agent) bir görevi en yüksek kazançla tamamlayabilmek için hangi eylemleri gerçekleştirmeleri gerektiği ile ilgilenen bir makine öğrenmesi tekniğidir. Bu tür öğreme algoritmaların girdisi öznelerin görev yapacakları farklı durumlardan oluşan bir ortam S, yapabilecekleri eylemler A, ortamdaki durumuna göre yapabilecekleri eylemleri belirleyen prensipler, bir durumdan diğer duruma geçtiklerinde elde edecekleri kazançtır.

Q-learning pekiştirmeli bir öğrenme algoritmasıdır. Ortam hakkında hiçbir şeyin bilinmediği durumlarda, Q-learning algoritması ortamı brute-force şeklinde, her ortam için olası tüm aksiyonları takip ederek, problem çözümü için en karlı yolu bulmaya çalışır. Q-learning algoritmasının girdileri kazanç matrisi olarak adlanıdırılan R matrisidir. Bu matrisin satır ve sütünları ortamları temsil etmekte, *R[i][j]* değeri ise i durumundan j durumuna geçildiğinde elde edilen anlık kazanç değeridir. Eğer i durumunda j durumuna bir geçiş yoksa *R[i][j]* değeri -1, geçiş var ancak j durumu hedef durum değilse değeri 0, j hedef durum ise değeri kullanıcı tarafından belirlenen bir kazanç değeridir.

Q-learning algoritmasının çıktısı ise öğrenmenin kalitesini gösteren Q matrisidir. Q-learning iteratif bir algoritmadır ve tüm değerleri başlangıçta 0 olan Q matrisi optimal değerlere yakınsadığında sona erer. Algoritma her iterasyonda rastgele bir durumdan öğrenmeye başlar, A'ya göre durum değiştirir ve Q matrisini günceller. A'ya göre hedef duruma ulaşıldığında itersyon sona erer. A'ya göre bir durumdan birden fazla duruma geçiş olabilir. Böyle bir durumda, olası geçişlerden biri rastgele seçilir. Eğer seçilen durum hedef duruma ulaştırmıyorsa, durum rastgele olacak durum olarak belirlenir. Hedef duruma ulaşılana

kadar iterasyon devam eder. Q matrisi aşağıdaki formule* göre güncellenir:

$Q(durum, aksiyon) = R(durum, aksiyon) + \gamma \times Max\{Q(sonrakidurumlar, tumaksiyonlar)\}$

γ öğrenme katsayısıdır ve 0 ile 1 arasında bir değer alır.

Aşağıdaki örnek Q-learning algoritmasını çalışmasını kısaca açıklamaktadır. Figür 1'de 6 durumdan oluşan bir ortam verilmiştir. Durumlar arası geçişler de oklar ile göstermektedir. Okların üzerindeki değerler anlık kazançları göstermektedir. Buna göre

- Durumlar $S = \{A, B, C, D, E, F\}$
- Eylemler ve kazaç değerleri $A = \{A \xrightarrow{0} B, A \xrightarrow{0} D, B \xrightarrow{0} A, B \xrightarrow{100} C, B \xrightarrow{0} E, D \xrightarrow{0} A, D \xrightarrow{0} E, E \xrightarrow{0} D, E \xrightarrow{0} B, E \xrightarrow{0} F, F \xrightarrow{100} C\}$

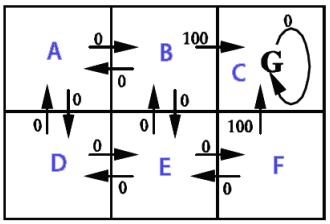


Figure 1: Ortam

Buna göre R matrisi ve Q matrisinin ilk hali aşağıdaki gibi olur.

$$R = \begin{pmatrix} -1 & 0 & -1 & 0 & -1 & -1 \\ 0 & -1 & 100 & -1 & 0 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 \\ -1 & -1 & 100 & -1 & 0 & -1 \end{pmatrix} \qquad Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Başlangıç durumu A olarak seçilsin ve $\gamma = 0.8$ olsun. A durumundan olasi aksiyonlar B durumuna geçiş ve D durumuna geçiştir. Rastgele olarak B durumuna geçiş seçilsin. Buna göre

$$Q(0,1) = R(0,1) + 0.8 \times Max\{Q(1,4), Q(1,2)\}$$

Buna göre Q(0,1) = 0 olarak güncellenir. B durumu hedef durum olmadığı için, durum B olarak güncellenir ve iterasyon devam eder. B durumundan olası aksiyonlar C durumuna geçiş ve E durumuna geçiştir. Rastgele olarak C durumuna geçiş seçilsin. Buna göre

$$Q(1,2) = R(1,2) + 0.8 \times Max\{Q(2,2)\}$$

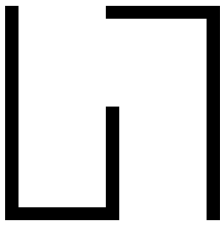
*Bu formülün farklı türevleri olmakla birlikte ödevde bu formulu kullanınız.

Buna göre Q(1,2) = 100 olarak güncellenir ve C hedef durum olduğu için iterasyon sonlandırılır. Optimuma yakınsama oluşana kadar yeni itersyonlar işletilir.

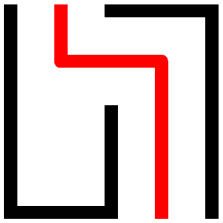
2 Ödev

Bu ödevde Q-learning algoritmasını kullanarak labirentte yol bulma problemi çözülecektir. Labirent bir çizge olarak düsünülebilir: labirentteki odacıklar düğüm olarak temsil edilirse, aralarında geçiş olan odalar bu çizgede bağlı düğümler olarak düşünülebilir.

Örnek olarak aşağıda verilen 2×2 labirentin odaciklari soldan sağa ve yukarıdan aşağı sırası gözetlenerek 0'dan itibaren adlandırılırsa, yanındaki gibi olur.



Düğüm	**Komşu**
0	1,2
1	0,3
2	0
3	1



- Bu ödevde **input.txt** isimli dosyadan komşuluk listesi verilen **kare** labirent okunacaktır. Bu dosyada başka bilgi olmayacaktır.
- Kullanıcıdan hedef odacık, başlangıç odacık numaraları ve iterasyon sayısı istenecektir. Eğer geliştireceğiniz uygulama konsoldan çalışıyorsa bunlar konsol parametreleri olara, grafiksel arayüz destekleyen bir uygulama geliştirecekseniz, arayüzden bu değerleri alabilirsiniz.
- Çıktı olarak ise R matrisi, son iterasyon sonunda oluşan Q matrisi ve başlangıç odacığından hedef odacığa giden yol verilecektir[†]. R matrisi **outR.txt** dosyasına, Q matrisi **outQ.txt** dosyasına, yol ise **outPath.txt** dosyasına yazılacaktır.
- Ayrıca labirentin ve yolun çizili olduğu bir grafik oluşturulmalıdır. Yukarıdaki figürde kırmızı ile belirtilen yoldur. Grafiğin çok ayrıntılı olması gerekmez, ancak odacık gecişleri ve yol anlaşılır olmallıdır.

Ödevde öğrenme katsayısı 0.8 olarak kullanılacaktır. Hedef düğüme bağlamayan geçişler için anlık kazanç 0, hedef durumuna bağlayan geçişlerin kazancı 100 olacaktır. Ayrıca her hedef durumdan kendine kazancı 100 olan geçiş tanımlanacaktır.

Girdi ve çıktı isterleri belirtildiği şekilde karşılanmalıdır. Labirent boyutu konusunda herhangi bir kısıtlama yoktur, ancak değerlendirmede kullanılacak labirentler kesinlikle *kare* labirent olacaktır.

[†]Örnek olarak yukarıdaki labirent için yol: 0,1,3 şeklindedir.