Pekiştirmeli Öğrenme Algoritmaları kullanılarak Ters Sarkacın Denge Kontrolü

Balancing Inverted Pendulum using Reinforcement Algorithms

Rüstem Özakar[1,2], Gülşah Tümüklü Özyer[2], Barış Özyer[2]
[1] Bilgisayar Mühendisliği Bölümü, Erzurum Teknik Üniversitesi, Erzurum, Türkiye
[2] Bilgisayar Mühendisliği Bölümü , Atatürk Üniversitesi, Erzurum, Türkiye
rustem.ozakar@erzurum.edu.tr, gulsah.ozyer@atauni.edu.tr, baris.ozyer@atauni.edu.tr

***Özetçe*— Teknolojik gelişmelerle birlikte robotlar karmaşık ve zor davranışları çeşitli makine öğrenme algoritmaları ile öğrenip gerçek hayatta uygulayabilir sistemler haline geldiler. Bu algoritmalar arasında pekiştirmeli öğrenme algoritmaları robotik sistemlere deneme ve yanılma yöntemiyle yeni görevler ve işlevler öğretmekte yaygın olarak kullanılan algoritmalardır. Bu çalışmada, amacımız iki farklı pekiştirmeli öğrenme algoritmasını kullanarak ters sarkaç sistemine dengede kalmasını öğretmek ve her iki algoritmanın performanslarını incelemektir. Bu amaçla Adaptif Sezgisel Kritik (ASK) ve Q-öğrenmesi pekiştirmeli öğrenme algoritmaları kullanılmıştır. Ters sarkaç sistemini ve ortamı simüle etmek için iki boyutlu Box2d simulatörü kullanılmıştır. Deneysel sonuçlar incelendiğinde ASK algoritmasının Q-öğrenmesine göre daha fazla adım sayısıyla sistemi dengede tuttuğu gözlenmektedir.**

Anahtar Kelimeler — pekiştirmeli öğrenme, ters sarkaç

***Abstract*— With the advancements in technology, robots has become systems that can learn and achieve complex behaviors in real life with the help of machine learning algorithms. Among those algorithms, reinforcement learning algorithms are widely used in robotics to teach the systems by trials and errors. In this work, our goal is to use the two different reinforcement algorithms, Q-learning and Adaptive Heuristic Critic (AHC) algorithm, on well-known cart-pole balancing problem and examine the performance results. We used Box2d physics engine simulator to simulate the cart-pole model and the environment. Observing the experimental results, AHC algorithm was able to balance the system for more step counts than Q-learning algorithm.**

Keywords — reinforcement learning, inverted pendulum

[1] GİRİŞ

Robotların son yıllarda mekanik tasarımlarında ve bu tasarımlarda kullanılan malzemelerdeki iyileştirmeler, elektronik parçalarındaki teknolojinin de ilerlemesi ile işlem yükü fazla olan işlemci ve hafıza elemanlarının kullanılması ve bu teknolojik gelişmelere bağlı olarak makine öğrenme algoritmalarının robotlar üzerinde uygulanabilir hale getirilebilmesi ile robotlar karmaşık görevleri yerine getirebilen, etrafı algılayabilen ve öğrenebilen makineler haline dönüşmeye başlamıştır. Bu sayede robotlar sadece endüstride kullanılan makineler olmaktan çıkıp, günlük hayatımızda da kullanabileceğimiz; mutfakta bulaşık makinesini boşaltan, etrafı süpürebilen ve futbol maçı yapabilen akıllı sistemler haline gelmiştir. Temelde makine öğrenme algoritmaları olarak literatürde çalışılan öğretici ile öğrenme, gözetimsiz öğrenme ve pekiştirmeli öğrenme algoritmaları robotik araştırmacılar tarafından robotların yeni görevleri öğrenmesinde ve bu zor görevleri yerine getirmesinde yaygın olarak kullanmaktadırlar [1].

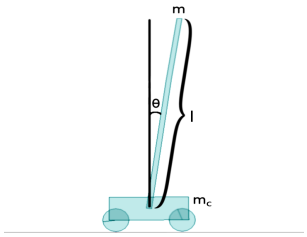
Biz bu çalışmamızda, pekiştirmeli öğrenme algoritması kullanarak ters sarkaç (inverted pendulum) sisteminin dengede kalmasını kontrol edebilen ve bu sistemin farklı başlangıç koşullarına göre kendi kendine dengede kalmasını öğrenen yöntemler önerdik ve denetim ortamında simülasyonunu gerçekleştirdik. Ters sarkaç sistemi üzerinde, iki farklı pekiştimeli öğrenme algoritmasının performanslarını inceledik ve karşılaştırdık. Pekiştirmeli öğrenme algoritması, bir ajana (robot) çevreyle etkileşim tecrübesiyle, durum ve hareket tahminlerini bir ödül veya ceza fonksiyonu kullanarak öğretme yöntemidir. Bu öğrenme deneme ve yanılma ile gerçekleşir. Ajanın çevreyi keşfedip istenen davranışı kendi kendine öğrenmesi beklenir. Pekiştirmeli öğrenme algoritması özellikle insansı robotlarda masa tenisi veya beyzbol oynayan, dengede kalma veya yürüme gibi zor görevleri öğretmek amacı ile literatürde uygulanmıştır. [2]-[4].

Ters sarkaç, bir araç üzerine dik olarak yerleştirilmiş bir çubuğu dengede tutmaya yarayan bir sistem geliştirilmesini içeren klasik bir kontrol teori problemidir. Genel olarak ters sarkaç üç alt sistemden meydana gelir. Bunlar mekanik sistem, geri besleme sistemi ve kontrol mekanizmasıdır. Problemin en sık kullanılan hali, lineer hareket eden bir araç üzerine yerleştirilmiş bir çubuk şeklindedir. Araç (cart) hareketi sağlanarak, sarkaç yukarı doğru dik bir biçimde dengelenir [5]. Ters sarkaç probleminin standartlaşmış bir uygulaması yoktur. Farklı türdeki pendulum platformları benchmark sonuçlarını karşılaştırmayı zor hale getirmiştir [6]. İki bacaklı robotların insan yürümesi modeliyle dengeli bir biçimde hareket ettirilmesi ve taşıyıcı algılayıcılar ile iki tekerlekli bir sistemin dengede tutabilen araçlarda ters sarkaç modeli kullanılmaktadır [7], [8].

Bildirinin geri kalan bölümleri şu şekilde düzenlenmiştir; 2. bölümde önerilen pekiştirmeli öğrenme algoritmalarının, yöntemlerin ve kullanılan ters sarkaç

978-1-5090-1679-2/16/$31.00 ©2016 IEEE

sisteminin matematiksel modelleri hakkında bilgi verilmektedir. 3. bölümde pekiştirmeli öğrenme algoritmalarının Box2d denetim ortamındaki ters sarkaç sistemiyle simülasyonu ile ilgili deneysel sonuçlar anlatılmıştır. 4. bölümde sonuçlar tartışılarak özetlenmiş ve gelecekteki yapmayı planladığımız çalışmalar belirtilmiştir.



Şekil-1: Araç-çubuk (cart-pole) sistemi

[2] METODOLOJİ

Bu çalışmada Adaptif Sezgisel Kritik (Adaptive Heuristic Critic) ve Q-öğrenmesi (Q-learning) pekiştirmeli algoritmalar kullanılmıştır. Temel amacımız bu iki farklı algoritma ile ters sarkaç sisteminin kendi kendine dengede kalmasını kontrol eden ve öğrenen bir sistem haline getirmek, bu iki algoritmanın performanslarını incelemek ve karşılaştırmaktır.

Pekiştirmeli öğrenme temelde, çevre durumları, aksiyon setleri, durumları arası geçiş kuralları ve ödüllerden oluşur. Pekiştirmeli öğrenme algoritmaları, ajanın hareketini belirlemek için poliçeler kullanır. Bu poliçe bir arama tablosu olabileceği gibi bir fonksiyon ya da arama işlemi de olabilir. Biz ajan olarak Şekil-1'de görülen araç-çubuk (cart-pole) sistemini kullandık. Durum ve aksiyon setlerini Boxes algoritması (Algoritma-1) kullanarak ayrıntılı bir şekilde 2.3 alt bölümünde açıkladık.

Pekiştirmeli öğrenme, Markov karar verme sürecine benzer olarak, verilen bir çevre modeline yönelik optimal poliçeleri hesaplayan dinamik programlama algoritmalarını içerir. Dinamik programlama ve pekiştirmeli öğrenmenin amacı genel olarak değer fonksiyonlarını kullanarak en iyi poliçeyi bulmaktır.

2.1. Q-Öğrenmesi

Q-öğrenmesi (Q-learning), Q(s,a) durum (state) ve aksiyon (action) değerlerinin en uygununu tahmin etmek için geçici farklılık (temporal difference) yöntemi üzerine inşa edilmiş poliçe bağımsız bir pekiştirmeli öğrenme algoritmasıdır. Aşağıdaki formül ile ifade edilir;

$$Q(s_{t,}a_t) = Q(s_{t,}a_t)+\alpha[r_{t+1}+\gamma \max Q(s_{t+1},a) - Q(s_t,a_t)] \quad (1)$$

s durumları, a aksiyonları, r ödülü ve t ise şu andaki durum ve aksiyonu göstermektedir.

α öğrenme oranı γ ise discount faktörüdür. Bu değerler 0-1 arası seçilir.

Q-öğrenmesi veri depolamak için tablolar kullanır. Genel olarak Q-öğrenmesi algoritması şu adımlarla özetlenebilir; başlangıçta algortima bir aksiyon seçer ve onu uygular, bunun sonucunda anlık bir ödül elde eder. Daha sonra yeni durumu gözler ve tablo girdilerini günceller.

2.2. Adaptif Sezgisel Kritik Algoritması

Adaptif Sezgisel Kritik (ASK) algoritması değer fonksiyonunun geçici farklılıklar (temporal difference) TD(λ) ile hesaplandığı bir pekiştirmeli öğrenme algoritmasıdır. Algoritma iki elementten oluşur, bunlar bir kritik ve pekiştirmeli öğrenme bileşenidir [9]. Kritik tarafından hesaplanan sezgisel (heuristic) değerini maksimize etmek için çalışır. ASK algoritması, sistemi sonunda bir bütün olarak analiz etmek yerine her adımda değerlendirmeye tutar. Aşağıda verilen formül ile ifade edilir;

$$V(u)=V(u)+\alpha(r + \gamma V(s') - V(s))e(u) \quad (2)$$

v değer fonksiyonu, e(u) üstünlük izleri (eligibility trace) ve r ödülü temsil etmektedir. Geçici farklılıklar, iki başarılı tahmin arasındaki fark olarak tanımlanabilir. TD(λ) algoritmaları bütün tahminlerin verilerini kaydetmeden öğrenme etkinliğini artırmak için üstünlük izleri kullanır [9]. Üstünlük izleri, her bir durum ziyaret edildikten sonra oluşturulan kısa dönemli bir hafıza görevi görür.

2.3. Inverted Pendulum Matematiksel Modeli ve Boxes Algoritması

Şekil-1'de görülen ters sarkaç sisteminin matematiksel modeli aşağıdaki şekilde ifade edilmektedir [11];

$$\ddot{\theta}=\frac{g\sin\theta+\cos\theta\left[\frac{-F-ml\dot{\theta}^2\sin\theta}{m_c+m}\right]}{l\left[\frac{4}{3}-\frac{m\cos^2\theta}{m_c+m}\right]} \quad (3)$$

$$\ddot{x}=\frac{F+ml[\dot{\theta}^2\sin\theta-\ddot{\theta}\cos\theta]}{m_c+m} \quad (4)$$

F araca uygulanan kuvvet, θ çubuğun açısı, l çubuk boyunun uzunluğunun yarısı, m_c araç ağırlığı, m çubuk ağırlığıdır. Amacımız, araç-çubuk hareket halindeyken çubuk açısını nomalle sıfır derece yapacak şekilde dengede tutmaktır.

Durum-uzayı temsil etmek için Algoritma-1'de görülen Boxes algoritması çalışmamızda kullanılmıştır. Algoritma ile durum-uzayı (state-space) ayrık bir şekilde modelleyebilmekteyiz. Her bir ayrık durum uzay

modeline kutu adı verilir. Bu çalışmada, durum-uzayını dört boyutlu vektor olarak olarak aşağıdaki gibi tanımlanmıştır;

$$s=[x, \dot{x}, \theta, \dot{\theta}] \quad (5)$$

x araç pozisyonu, $\dot{x}$ araç hızı, θ çubuğun normalle yaptığı açı, $\dot{\theta}$ çubuğun açısal hızıdır. Her bir kutu, sistem o kutucuğa ulaştığında bulunulan duruma göre çalıştırılacak olan aksiyonu içerir [10]. Algoritmada, durumlar fonksiyona parametre olarak anlık bir biçimde gönderilmektedir. Algoritma da bu durumların değerlerine göre, sistemin konumunu kutu değişkenini değiştirerek belirlemektedir.

[3] DENEYSEL SONUÇLAR

İki farklı pekiştirmeli öğrenme algoritmasının ters sarkaç sistemi üzerinde simüle edilip performanslarını incelenmesi için C++ programlama dili üzerine yazılmış iki boyutlu açık kaynak kodlu karmaşık fiziksel işlemleri hesaplamak ve ekrana gerçeğe uygun bir biçimde iki boyutlu olarak yansıtmak için kullanılan Box2D simülatörü kullanılmıştır. Şekil-2'de, robotun öğrenme işlemini gerçekleştirirken Box2D çalışma ekranından örnekler gösterilmiştir. Bu çalışmamızda kullandığımız algoritmalar ve başlangıç parametreleri Sutton ve Anderson[1,2] tarafından verilen değerleri ile simülasyonda kullanılmıştır. Başlangıç parametreleri olarak uygulanan değerler Tablo-1 deki gibidir;

Çubuk ağırlığı *(m)*	0.1
Araç (cart) ağırlığı *(m_c)*	1
Çubuk uzunluğu *(l)*	1
Alpha (Q-öğrenmesi) *(α)*	0.30
Alpha (ASK) *(α)*	1000
Beta (ASK) *(β)*	0.5
Gamma (Q-öğrenmesi) *(γ)*	1
Gamma (ASK) *(γ)*	0.95
Reward *(r)*	0
LambdaV	0.8
LambdaW	0.9
Force	10
Seed (Q-öğrenmesi)	24

Tablo-1: Simülasyonda kullanılan değerler

Simülasyonun genel çalışma prensibi şu şekildedir; algoritmaya göre sağa ya da sola uygulanacak kuvvet deneysel olarak seçilmektedir. (3) ve (4) denklemler kullanılarak uygulanacak ivme kuvveti hesaplanmaktadır. $\dot{\theta}$ ve $\dot{x}$ durum parametreleri çubuğun dengede tutulması için (3) ve (4) denklemlerinde hesaplamada kullanılmaktadır. Denklem (5)'teki durumlar Boxes algoritmasına simülasyondan okunup parametre olarak gönderilmektedir. Algoritmanın ürettiği çıktılara göre pekiştirmeli öğrenme gerçekleştirilmektedir.

Q-öğrenmesi algoritmasında, başlangıçta bir Q dizisi, rastgele değerlerle doldurulmaktadır. Algoritma bu dizideki rastgele değerlere göre bir aksiyon seçip çalışmaya başlamaktadır. Daha sonra öğrenme gerçekleştikçe bu dizideki değerleri güncellemektedir. Bu rastgele diziyi oluşturmadan önce, rastgele bir tohum (seed) değeri belirlenmektedir.

```
procedure BOXES(x, xdot, theta, thetadot)
    box ← 0
    if x < −2.4 ∥ x > 2.4 ∥ theta < −twelvedegrees ∥ theta > twelvedegrees then
        return −1
    if x < −0.8 then
        box ← 0
    else if x < 0.8 then
        box ← 1
    else
        box ← 2
    if xdot < −0.5 then
        ;
    else if xdot < 0.5 then
        box ←box +3
    else
        box ←box +6
    if theta < −sixdegrees then
        ;
    else if theta < −onedegree then
        box ←box +9
    else if theta < 0 then
        box ←box +18
    else if theta < onedegree then
        box ←box +27
    else if theta < sixdegrees then
        box ←box +36
    else
        box ←box +45
    if thetadot < −fiftydegrees then
        ;
    else if thetadot < fiftydegrees then
        box ←box +54
    else
        box ←box +108
    return box
```

Algoritma-1: Boxes algoritması

ASK algoritmasında, başlangıçta dört adet vektör bulunmaktadır. Bu vektörler, aksiyon ağırlıkları, kritik ağırlıkları, aksiyon ağırlıkları üstünlükleri ve kritik ağırlıkları üstünlükleri (eligibility) şeklindedir. Q-öğrenmesi'nin aksine bu vektörler başlangıçta sıfır değeriyle doludur. Algoritma çalıştıkça, vektörleri uygun değerlerle doldurarak, çubuğu dengede tutacak vektörleri oluşturur. Yine Q-öğrenmesi'nden farklı olarak, LambdaV ve LambdaW isimli iki değişken, üstünlük izleri için bozulma oranı (decay rate) olarak kullanılmaktadır.

Araca uygulanacak kuvvet 10 olarak belirlenmiştir. Bu kuvvet veya negatifi, algoritmada seçilerek, denklemde hesaplamalardan sonra ivmeye çevrilmekte, bu çevrilen değer simülasyonda tekerleklere dönme kuvveti olarak uygulanmaktadır.

Şekil-2: Box2d simülasyon ekran görüntüleri

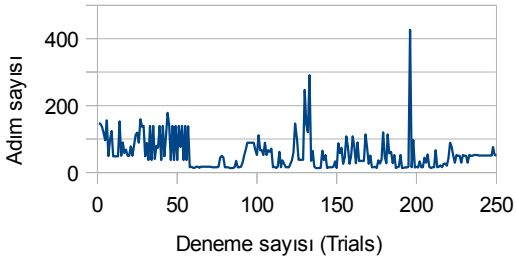
1: http://webdocs.cs.ualberta.ca/~sutton/book/code/pole.c
2: http://www.cs.cmu.edu/afs/cs/project/lri/members/lalit/Phd/course/nn/asg4/qpole.c

Her bir algoritma için 250 deneme yapılmıştır. Her bir deneme için, çubuğun düşmeden dengede durduğu süre adım olarak hesaplanmaktadır.

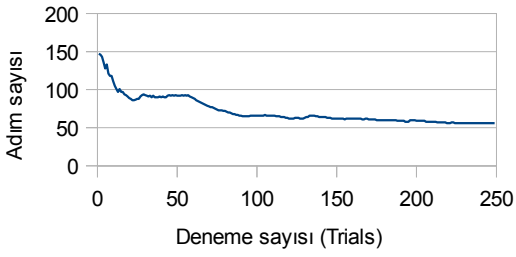
Denemelerin sonunda, her bir deneme için elde edilen adım değeri ve ortalama adım değerleri kaydedilmektedir. Simülasyon çalışırken, her adımdaki adım sayısı bütün olarak toplanmakta, bu toplam şu andaki deneme sayısına böldürülüp ortalama adım sayısı hesaplatılmaktadır.

Şekil-3'te, Q-öğrenmesi algoritmasının her bir denemede elde ettiği adım sayısı gösterilmiştir. Yine Şekil-4'te, Q-öğrenmesi için her bir deneme için hesaplanan ortalama adım sayısı gösterilmektedir. Şekil-3'te ortalama adım sayısı 150'den 50'ler civarına kadar düşmektedir.

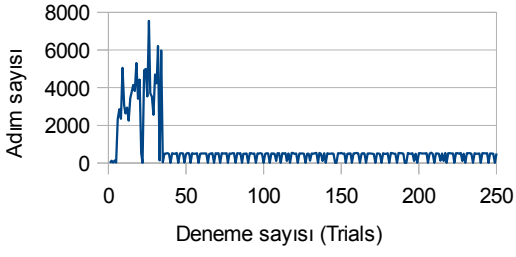
Şekil-5'te, ASK algoritmasının her bir denemede elde ettiği adım (step) sayısı gösterilmiştir. Şekil-6'da, ASK için her bir deneme için hesaplanan ortalama adım sayısı gösterilmektedir. Şekil-5'te görüldüğü üzere, algoritma yaklaşık 40 adım sonunda, adım sayısı olarak 500 civarına inmektedir. Şekil-6'da ortalama adım sayısı 3000'den 700'ler civarına kadar düşmektedir.



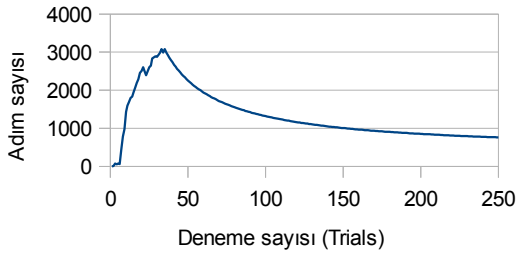
Şekil-3: Q-öğrenmesi algoritması adım grafiği



Şekil-4: Q-öğrenmesi algoritması ortalama adım grafiği



Şekil-5: ASK algoritması adım grafiği



Şekil-6: ASK algoritması ortalama adım grafiği

[4] SONUÇ

Çalışmamızda ters sarkaç sisteminin kendi kendine dengede kalmasını öğrenebilmesi için kullanılan iki farklı pekiştirmeli öğrenme algoritmasının performans sonuçları incelenmiştir. Her iki algoritma da daha önceden yaygın olarak kullanılan parametreler ile çalıştırılmış ve birbirleri ile farklı sonuçlar elde edilmiştir. Q-öğrenmesi algoritması, başlangıçta elde edilen rastgele diziye bağlı olduğundan, algoritmanın başarısı başlangıçta seçilen seed değerine ve rastgele dizinin oluşturulma şekline oldukça bağlıdır. ASK algoritmasında ise, sonuçlar seçilecek parametre değerlerine daha fazla bağlıdır. Gelecek olarak, önerilen pekiştirmeli öğrenme algoritmalarını çok parmaklı robot el ile nesne manipülasyonu yapan bir sisteme adapte etmek için çalışmalarımız devam etmektedir.

KAYNAKÇA

[1] J. Kober, J. A. Bagnell and J. Peters, "Reinforcement learning in robotics: A survey", International Journal of Robotic Research, 32(11), 2013.

[2] S. Schaal, "Is imitation learning the route to humanoid robots?", Trends in cognitive sciences, 3(6), pp. 233-242, 1999.

[3] Y. Ng Andrew, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning", Experimental Robotics IX, pp.363-372, Springer Berlin Heidelberg, 2006.

[4] M. Riedmiller, T. Gabel, R. Hafner and S. Lange, "Reinforcement learning for robot soccer", Autonomous Robots, 27(1), pp. 55-73, 2009.

[5] R. Kumar, R. B. Singh and J. Das, "Modeling and Simulation of Inverted Pendulum System Using Matlab: Overview" International Journal of Mechanical and Production Engineering, 1(4), pp. 52-55, Oct. 2013.

[6] J. Brownlee, "The Pole Balancing Problem: A Benchmark Control Theory Problem", Technical Report 7-01, July 2005

[7] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi and H. Hirukawa, "The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation", IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, pp. 239-246, 2001.

[8] H. G. Nguyen, J. Morrell, K. Mullens, A. Burmeister, S. Miles, N. Farrington, K. Thomas and D. W. Gage, "Segway Robotic Mobility Platform", The International Society for Optical Engineering, pp. 207-220, 2004.

[9] X. Xu, H. He and D. Hu, "Efficient Reinforcement Learning Using Recursive Least-Squares Methods", Journal of Artificial Intelligence Research 16, pp. 259-292, May 2002.

[10] C. Sammut, "Recent progress with BOXES", Machine intelligence 13, pp. 363-383, 1994.

[11] P. F. P. Faustino, "Dynamic Equilibrium through Reinforcement learning", M.S. thesis, Computer and Information Engineering, ISEL, Lisbon, Portugal, 2011.