ورقة بحثية مجلة مفتوحة الوصول

المفاهيم الأساسية للتعلم المعزز

د. سامر سليمان*، مها ديوب**
*(قسم الميكاترونيكس، جامعة المنارة
البريد الإلكتروني: samer.sulaiman@manara.edu.sy)
**(قسم الحاسبات والتحكم الآلي، جامعة تشرين
البريد الإلكتروني: mahadayoub@gmail.com)

الملخص
تقدم هذه المقالة شرحاً عن مفهوم التعلم المعزز وعناصره الأساسية كأحد مجالات تعلم الآلة، مع توضيح الاختلاف بينه وبين المجالات الأخرى (التعلم مع إشراف، التعلم بدون إشراف)، كما سنوضح بشكل مختصر عملية القرار باستخدام ماركوف، إضافة إلى تطبيق بعض الأمثلة البرمجية عن التعليم المعزز باستخدام إطار العمل OpenAI gym ولغة بايثون.

كلمات مفتاحية – تعلم الآلة، التعلم مع إشراف، التعلم بدون إشراف، التعلم المعزز

I. مقدمة

التعلم المعزز هو أحد مجالات تعلم الآلة الذي ظهر في فترة الخمسينيات، وبدأ بالانتشار عندما قام فريق DeepMind بإنشاء خوارزمية AlphaGo وتمكنت من التغلب على بطل العالم في لعبة Go. اليوم، لم يعد استخدام التعلم المعزز مقتصرا على الألعاب، بل توسع ليشمل مجالات الصحة والتجارة والروبوتات إضافة إلى القيادة الذاتية. **Error! Reference source not found.**

II. تعلم الآلة

يمكن تعريف تعلم الآلة على أنه العلم الذي يهتم ببرمجة الحواسيب بحيث تصبح قادرة على التعلم من البيانات، أو كما يعرفه Tom Mitchell: يقال عن برنامج حاسوبي أنه يتعلم من تجربة E تجاه مهمة ما T ومقياس أداء ما P، إذا كان أداؤه على المهمة T مقاساً حسب P يتحسن من خلال التجربة E. **Error! Reference source not found.**

يمكن تصنيف تعلم الآلة حسب الإشراف إلى تعلم مع إشراف، تعلم بدون إشراف، والتعلم المعزز.

III. التعلم المعزز وعناصره

A. تعريف التعلم المعزز

تتمحور الفكرة الأساسية للتعلم المعزز بوجود عميل يحاول اكتشاف حالات البيئة المحيطة به والتفاعل معها من خلال أفعال بهدف تعزيز مقدار يدعى الجائزة، وربط كل حالة بالفعل المناسب لها، وهنا يكمن الاختلاف بينه وبين الأصناف الأخرى. ففي التعلم مع إشراف، يكون لدينا بيانات معنونة أي أننا نحدد لخوارزمية التعلم الخرج الموافق للدخل الحالي، بينما في التعلم بدون إشراف، تكون البيانات غير معنونة، وعلى الخوارزمية أن تكتشف التشابه بين المدخلات والفصل بينها اعتمادا عليه. **Error! Reference source not found.**

B. عناصر التعليم المعزز

- الاستراتيجية Policy: تحدد كيفية تفاعل العميل مع بيئته، يمكن أن تكون تابع أو جدول أو شبكة عصبونية.

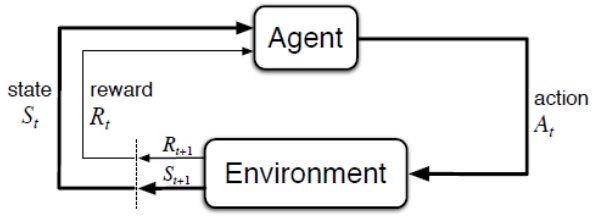
- الجائزة Reward: الجائزة هي الإشارة الفورية التي تعيدها البيئة للعميل بعد اتخاذ فعل معين. هدف العميل الأساسي هو الحصول على أكبر جائزة على المدى الزمني الطويل، وإذا كانت قيمتها سالبة فهذا يعني أنه على العميل تغيير الفعل المتخذ عند الوقوع مجدداً في الحالة التي تسببت بالإشارة السالبة.
- تابع القيمة Value Function: هو مجموع الجوائز التي يحصل عليها العميل انطلاقاً من حالة معينة حتى الوصول إلى الهدف باستخدام استراتيجية معينة.
- نموذج البيئة Environment Model: هو تمثيل لسلوك البيئة يمكنه توقع الحالة الجديدة للبيئة بعد اتخاذ العميل لفعل معين انطلاقاً من حالة ما، وإلا سيقوم العميل بعملية التجريب والخطأ حتى يصل إلى الهدف. **Error! Reference source not found.**

IV. عملية القرار باستخدام ماركوف

تعد هذه الطريقة مثالية لتمثيل مشكلة التعلم المعزز بشكل رياضي، المتعلم والذي يتخذ القرار يدعى العميل، أما الوسط المحيط به ويتفاعل معه يدعى البيئة. **Error! Reference source not found.**

A. العميل والبيئة

في كل لحظة زمنية t، يستقبل العميل من البيئة حالة S_t ويقوم باتخاذ فعل A_t، فتعيد له البيئة إشارة الجائزة في اللحظة التالية R_{t+1} وحالة جديدة S_{t+1} ليقوم باتخاذ فعل جديد A_{t+1} وهكذا... **Error! Reference source not found.**



الشكل 1. تفاعل العميل مع البيئة في عملية القرار باستخدام ماركوف

B. القيم المرجعة والحلقات

ذكرنا سابقاً أن هدف العميل هو الحصول على أكبر جائزة خلال المدى الزمني الطويل، أي مجموع الجوائز لكل فعل متخذ حتى الوصول إلى الهدف، وهذا يدعى بالعائد Return ويرمز له ب G_t ويعرف تابعها بالعلاقة التالية:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T$$

حيث T هي الخطوة الزمنية الأخيرة. وهذا يقودنا إلى تمييز نوعين من مسائل التعلم المعزز: المنتهية والمستمرة.

في المسائل المنتهية تُجزَّأ أفعال العميل بشكل طبيعي إلى مراحل جزئية وتدعى بالحلقات تنتهي كل منها بحالة طرفية بعدها نعود إلى الحالة الابتدائية مجدداً بعد إعادة تهيئتها أو إلى حالة جديدة، مثلا في الألعاب تكون اللعبة مؤلفة من عدة مراحل كل منها يمكن أن تنتهي بالفوز أو الربح لتبدأ المرحلة التالية بشكل مستقل عن المرحلة السابقة.

أما في المسائل المستمرة يقوم العميل باتخاذ أفعال بشكل مستمر مثل عملية تحكم نشطة. المشكلة في هذه المسائل هي كيفية تعريف تابع العائد باعتبار أن T=∞، وبالتالي قد لا نحصل على قيمة محددة للتابع.

لحل المشكلة السابقة نلجأ إلى مفهوم التخفيض بحيث يصبح هدف العميل الحصول على أعظم عائد مخفض. سنعرف أولا عامل التخفيض g ليكون متغير تتراوح قيمته بين 0 و1. هذا العامل يحدد معدل تخفيض قيم الجوائز المكتسبة في المستقبل ويحدد القيمة الحالية لها، وعليه يصبح تعريف تابع العائد المخفض:

$$G_t = R_{t+1} + g\ R_{t+2} + g^{\,2} R_{t+3} + \cdots = \sum_{k=0}^{\infty} g^{\,k} R_{t+k+1}$$

إذا كانت قيمة g=0 يصبح هدف العميل الاهتمام بزيادة قيمة الجائزة الفورية فقط، أما إذا كانت g=1 يهتم العميل بالجوائز المستقبلية بنفس مقدار الجائزة الفورية وهذا يزيد من صعوبة اتخاذ القرار، لذا تكون قيمته عادة متراوحة بين 0 و1، وفي المسائل المستمرة يجب أن تكون أصغر من 1 حتى نتمكن من

الحصول على قيمة منتهية لتابع العائد. **Error! Reference source not found.**

مثلا، إذا كانت الجائزة في كل خطوة زمنية ثابتة وتساوي الواحد يكون تابع العائد:

$$G_t = \sum_{k=0}^{\infty} \mathrm{g}^k = \frac{1}{1-\mathrm{g}}$$

C. الاستراتيجيات وتوابع القيمة

تحدد الاستراتيجية احتمال أن يأخذ العميل فعل معين في حالة معينة ويرمز لها ب π. إذا كان العميل يتبع سياسة π في زمن t، حينها $\pi(a|s)$ هي احتمال أن $A_t = a$ إذا كانت $S_t = s$.

ومن أجل كل **s ϵ S** يكون توزع الاحتمال ضمن **a ϵ A(s)**.

تابع القيمة إما أن يخمن القيمة المتوقعة انطلاقاً من الحالة المتواجد فيها العميل واتباع استراتيجية π ويدعى هنا بتابع القيمة–الحالة ويرمز له ب v_π، ويعطى بالعلاقة:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$
$$= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \mathrm{g}^{\,k} R_{t+k+1} \quad |S_t = s], \boldsymbol{s \in S}$$

أو أن يخمن القيمة المتوقعة انطلاقاً من الحالة المتواجد فيها العميل واتخاذ فعل باتباع استراتيجية π ويدعى هنا بتابع القيمة–الفعل ويرمز له ب $q_\pi(s,a)$ ويعرف بالعلاقة:

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$
$$= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \mathrm{g}^{\,k} R_{t+k+1} \,|S_t = s, A_t = a]$$

هذا التابع يدعى أيضا ب Q–function، وخرجه يدعى Q–value، والحرف `Q` يستخدم للتعبير عن جودة quality اتخاذ فعل ما في حالة ما. **Error! Reference source not found.**

D. المثالية Optimality

يجب على العميل أن يجد أفضل استراتيجية تحقق له أكبر قيمة من الجوائز على المدى الزمني الطويل، وتعتبر الاستراتيجية π مثالية إذا كانت تعطي قيمة أكبر لتابع القيمة–الحالة من استراتيجية أخرى π'.

$$\pi \geq \pi' \; if \; and \; only \; if \; v_\pi(s) \geq v_{\pi'} \; for \; all \; s \in S$$

أما تابع القيمة–الحالة المثالي يعطي أكبر قيمة عائد متوقعة لكل حالة باتباع عند اتباع استراتيجية π.

$$v_*(s) = \max_\pi v_\pi(s)$$

وذلك من أجل كل **s** $\in \boldsymbol{S}$

تابع القيمة–الفعل المثالي أو Q–function المثالي والي يرمز له ب q_* يعطي أكبر عائد يمكن الحصول عليه باتباع استراتيجية π من أجل كل زوج حالة–فعل ممكن.

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

1) معادلة مثالية بيلمان من أجل التابع $\boldsymbol{Q_*}$:

معادلة مثالية بيلمان من أجل التابع Q_* تحدد أنه من أجل أي زوج حالة–فعل (s,a) عند زمن t، العائد المتوقع عند البدء من حالة s واختيار فعل a واتباع استراتيجية مثالية بعدها سيكون الجائزة المتوقعة من اتخاذ الفعل a في الحالة s، وهي R_{t+1} مضافاً إليها العائد المخفض المتوقع الأكبر والذي يمكن الحصول عليه من أي زوج حالة–فعل تالٍ متوقع(s′,a′)، وتمثل المعادلة كالآتي:

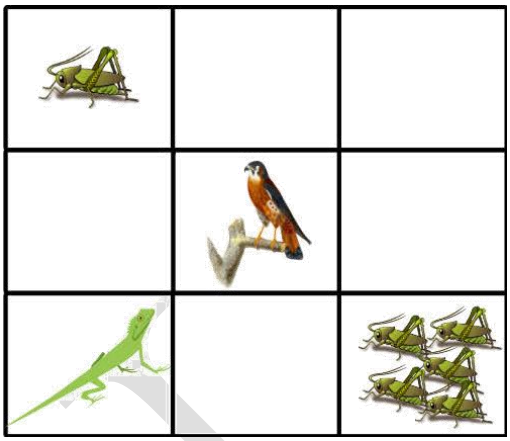
$$q_*(s,a) = \mathbb{E}[R_{t+1} + \mathrm{g} \max_{a'} q_*(s',a')]$$

E. Q–Learning و Q–Tables

يهدف Q–Learning إلى إيجاد الاستراتيجية المثالية من خلال تعلم Q–values المثالية من أجل كل زوج حالة–فعل.

خوارزمية Q–Learning تقوم بتحديث Q–values من أجل كل زوج حالة–فعل باستخدام معادلة بيلمان بشكل متكرر حتى الحصول على تابع Q–value المثالي، وهذه الطريقة تدعى تكرار القيمة value iteration.

مثال لعبة السحلية. البيئة موضحة في الشكل2، العميل هو السحلية. ستحاول السحلية أكل crickets قدر المستطاع خلال أقصر مدة زمنية مع تجنب الطائر وإلا سيأكلها.



الشكل 2. بيئة لعبة السحلية

يمكن أن تتحرك السحلية نحو اليسار، اليمين، الأعلى، أو الأسفل. أما الحالات تحدد بموقع السحلية على الرقعة عند لحظة زمنية ما. إذا وصلت السحلية إلى موقع يحوي cricket واحد، الجائزة نقطة واحدة، وإذا وصلت إلى موقع يحوي 5 crickets الجائزة 10 نقاط وتنتهي الحلقة، أما إذا وصلت إلى الموقع الذي يحوي الطائر فالجائزة −10 وأيضاً تنتهي الحلقة.

الحالة	الجائزة
Cricket واحد	+1
فارغ	−1
خمس crickets	+10 وتنتهي اللعبة
طائر	−10 وتنتهي اللعبة

الجدول 1. الحالات والجوائز في لعبة السحلية

في البداية، السحلية لا تعلم أي شيء عن البيئة وبالتالي لا تعلم ما هو الفعل الذي يجب أن تتخذه، لهذا تكون Q−values لكل زوج حالة−فعل صفرية في البداية.

يتم استخدام جدول يدعى Q−table لتخزين هذه القيم، سطور الجدول تمثل الحالات، بينما تمثل الأعمدة الأفعال.

	يسار	يمين	أعلى	أسفل
Cricket واحد	0	0	0	0
فارغ 1	0	0	0	0
فارغ 2	0	0	0	0
فارغ 3	0	0	0	0
طائر	0	0	0	0
فارغ 4	0	0	0	0
فارغ 5	0	0	0	0
فارغ 6	0	0	0	0
خمس crickets	0	0	0	0

الجدول 2. Q−table الابتدائي

في البداية ستختار السحلية أفعال عشوائية حتى تكتشف حالات البيئة ومعرفة Q−value لكل زوج حالة−فعل ثم تقوم بتحديث الجدول..**Error! Reference source not found**

F. الاكتشاف Exploration والاستغلال Exploitation

الاكتشاف هو التعرف على البيئة لجمع معلومات عنها، أما الاستغلال فهو استغلال معلومات معروفة مسبقا عن البيئة للتفاعل معها.

يجب تحقيق الموازنة بين الاكتشاف والاستغلال في مسائل التعلم المعزز، وبالعودة إلى مثالنا، في البداية سيكون من المنطقي أن تقوم السحلية باختيار أفعال عشوائية حتى تتمكن من اكتشاف البيئة ثم بعد جمع المعلومات تقوم باستغلالها لاتخاذ الفعل الأفضل الموافق لكل حالة، ولكن لا يمكن القيام بالاكتشاف دائماً لأن هذا يسبب تأخير كبير في إنجاز المهمة، ولا يمكن القيام بالاستغلال دائماً لأن العميل ممكن ألا يكتشف بعض الأفعال التي تعطي عائد أكبر.

عادة يتم اكتشاف البيئة بنسبة 30% والاستغلال بنسبة 70%، وإحدى الخوارزميات المتبعة لتحقيق هذا التوازن هي خوارزمية إبسيلون الجشعة epsilon greedy والتي تعمل كالآتي:

في البداية نحدد $\varepsilon = 1$ أي أن العميل حتماً سيقوم بالاكتشاف، ثم نولد رقماً عشوائياً r كل خطوة زمنية، إذا كان $r \leq \varepsilon$ يقوم العميل باكتشاف البيئة وإلا يقوم بالاستغلال أي سيختار من Q− table الفعل الذي يعطي أكبر عائد للحالة المتواجد بها.

بعد أن تقوم السحلية بتنفيذ الفعل، تنتقل إلى حالة جديدة وتكتسب جائزة من فعلها، ثم تحدث قيمة Q–value للفعل الذي اتخذته في الحالة السابقة.

كما ذكرنا سابقا يتم تحديث Q–value باستخدام معادلة بيلمان، ونريد جعل الفرق بين q(s,a) و $q_*(s,a)$ أصغر ما يمكن ويمثل هذا الفرق الخطأ.

يتم حساب Q–value الجديدة باستخدام الصيغة:

$$q^{new} = (1-\alpha)q(s,a) + \alpha(R_{t+1} + \operatorname{gmax}_{a'} q(s',a')]$$

q(s,a) تمثل القية القديمة، بينما الحد الثاني يمثل القيمة المتعلمة.

α هو معدل التعلم تتراوح قيمته بين 0 و1، يحدد كمية المعلومات التي سيحتفظ بها العميل عن Q–value السابقة لزوج حالة–فعل مقارنة بقيمة Q–value الجديدة لنفس الزوج، ويتم فرضه 0.7 في لعبة السحلية. من الصيغة السابقة نلاحظ أنه إذا كانت α=1 لن يحتفظ العميل بأي معلومة عن Q–value السابقة.

بفرض أن السحلية في الحالة الابتدائية اختارت أن تتحرك نحو اليمين، ستصل إلى الموقع الفارغ 6 وتأخذ جائزة 1–. **Error! Reference source not found.**

بفرض عامل التخفيض يساوي 0.99 وبالتعويض في الصيغة السابقة نحصل على:

$$q^{new} = (1-0.7)(0) + 0.7(-1 + 0.99\left(\max_{a'} q(s',a')\right))$$

للتذكير الجدول مهيأ حاليا بقيم ابتدائية صفرية.

$$\max_{a'} q(s',a') = \max\left(q(empty6,left), q(empty6,right), q(empty6,up), q(empty6,down)\right)$$

=max(0,0,0,0)

=0

نعوض هذه القيمة في الصيغة السابقة:

$$q^{new} = (1-0.7)(0) + 0.7(-1 + 0.99(0))$$

=−0.7

V. تطبيق بعض الأمثلة باستخدام OpenAI gym

OpenAI gym هو إطار عمل يقدم عدد كبير من البيئات التي يمكن تطبيق خوارزميات التعلم المعزز فيها وتطويرها. **Error! Reference source not found.**

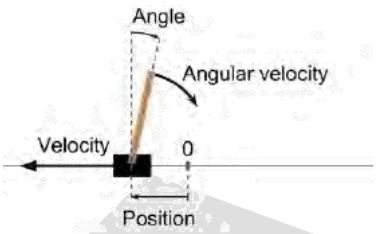
يمكن تحميلها بكتابة السطر البرمجي

pip install gym

سنبدأ بمثال بسيط لمعرفة أهم توابع هذا الإطار ثم سننتقل إلى مثال يشمل الأفكار المطروحة في هذا المقال، وسنستخدم لغة البرمجة بايثون.

A. بيئة CartPole–v0

في هذه البيئة لدينا عصا مركبة على عربة والهدف هو الحفاظ على توازن العصا. **Error! Reference source not found.Error! Reference source not found.**



الشكل 3. بيئة CartPole

أولا سنقوم باستدعاء الإطار

```
import gym
```

ثانيا سنقوم بإنشاء البيئة باستخدام التابع make

```
env=gym.make("CartPole-v0")
obs=env.reset
print(obs)
env.render()
```

التابع reset يعيد الحالة الأولى للبيئة. في بيئة CartPole كل حالة هي مصفوفة سطرية تحوي أربع قيم تمثل موقع العربة، سرعتها، زاوية العصا، وسرعتها الزاوية.

التابع render يقوم بعرض البيئة.

لمعرفة الأفعال الممكنة في البيئة:

```
env.action_space.n
```

سيكون الخرج في هذه البيئة 2. أي هناك فعلين 0 و 1.

0 يمثل تسارع العربة نحو اليسار، بينما 1 يمثل تسارع العربة نحو اليمين.

ثالثا اتخاذ فعل ضمن البيئة

```
action=1
obs,reward,done,info=env.step(action)
```

تابع step يقوم بتنفيذ الفعل الممر له ويعد أربع قيم:

obs الحالة الجديدة للبيئة

reward في هذه البيئة نحصل على جائزة +1 كل خطوة، أي الهدف الحفاظ على توازن العصا لأطول فترة ممكنة.

done تصبح هذه القيمة True عندما تنتهي الحلقة، وهذا يحدث عندما تميل العصا كثيراً.

info قاموس يحتوي على معلومات تصحيح إضافية متعلقة بحالة البيئة السابقة لذا لا يجب استخدامه في التدريب (يعتبر غش).

أخيرا سنقوم بتعريف تابع الاستراتيجية

```
def basic_policy(obs):
    angle=obs[2]
    return 0 if angle < 0 else 1
totals=[ ]
for episode in range(500):
    episode_rewards=0
    obs=env.reset()
     for step in range(1000):
          action=basic_policy(obs)
           obs,reward,done,info=env.step(action)
           episode_rewards+=reward
           if done:
                break
      totals.append(episode_rewards)
```

يتم اختيار الفعل باستخدام التابع basic_policy، نمرر للتابع حالة البيئة obs واعتمادا على زاوية العصا obs[2] نسرع السيارة باتجاه اليسار إذا كانت الزاوية سالبة أي أن العصا تميل نحو اليسار وإلا نسرع السيارة باتجاه اليمين.

totals هي القائمة التي سنخزن فيها الجوائز خلال فترة التعلم، في البداية تكون فارغة.

سوف نجري 500 حلقة في بداية كل حلقة سنعيد تهيئة جوائز الحلقة بقيمة صفرية وإعادة البيئة إلى حالتها الابتدائية.

في كل حلقة سنقوم باتخاذ 999 خطوة، وكما ذكرنا نختار الفعل اعتمادا على تابع الاستراتيجية ثم نمرره للتابع step لتنفيذه ثم نقوم بزيادة جوائز الحلقة ونختبر إذا انتهت الحلقة أو لا، إذا انتهت نبدأ حلقة جديدة، أخيرا نضيف قيمة جوائز كل حلقة إلى القائمة totals.

والآن لمعرفة كم استطعنا الحفاظ على توازن العصا سنستدعي التابع max من مكتبة numpy الخاصة بالمصفوفات.

```
import numpy as np
print(np.max(totals))
```

يمكن أن تختلف هذه القيمة في كل مرة ننفذ فيها المقطع البرمجي السابق، أكبر قيمة حصلت عليها هي 74 أي في إحدى الحلقات بقيت العصا متوازنة في 74 خطوة من أصل 999 وهذه النتيجة ليست جيدة بالتالي يجب تغيير الاستراتيجية المتبعة..**Error! Reference source not found**

B. بيئة FrozenLake-v0:

الشكل 4. بيئة FrozenLake-v0

يمثل سطح البيئة بالشبكة التالية:

```
SFFF
FHFH
FFFH
HFFG
```

S هي نقطة بداية العميل وهي آمنة.

F يمثل السطح المتجمد وهو أيضاً آمن.

H تمثل حفرة وإذا سقط فيها العميل تنتهي اللعبة.

G مكان وجود الطبق الطائر وهو الهدف.

يمكن للعميل أن يتحرك نحو الأعلى، الأسفل، اليمين، أو اليسار، وتنتهي اللعبة عند الوقوع في حفرة أو عند الوصول إلى الهدف..**Error! Reference source not found.**

الحالة	الوصف	الجائزة
S	نقطة البداية– آمنة	0
F	سطح متجمد– آمن	0
H	حفرة– تنتهي اللعبة	0
G	الهدف– تنتهي اللعبة	1

الجدول 3. توصيف بيئة FrozenLake

أولا سنبدأ باستدعاء gym والمكتبات اللازمة

```
import numpy as np
import gym
import random
import time
```

ثانيا إنشاء البيئة

```
env = gym.make("FrozenLake-v0")
```

ثالثا إنشاء Q-table

```
action_space_size = env.action_space.n
state_space_size = env.observation_space.n
```

env.observation_space تعيد الحالات الممكنة في البيئة وهنا توجد 16 حالة.

```
q_table = np.zeros((state_space_size, action_space_size))
```

هنا أنشأنا مصفوفة صفرية (الحالة الابتدائية للجدول) مؤلفة من 16 سطر (عدد الحالات) و4 أعمدة (عدد الأفعال).

رابعاً تهيئة بارامترات Q-learning

```
num_episodes = 10000
max_steps_per_episode = 100
learning_rate = 0.1
discount_rate = 0.99
exploration_rate = 1
max_exploration_rate = 1
min_exploration_rate = 0.01
exploration_decay_rate = 0.001
```

exploration_rate يمثل ε في البداية قيمته 1، القيمة العظمى max له هي 1 والقيمة الدنيا min هي 0.01، أما exploration_decay_rate يحدد مقدار تناقص exploration_rate

أخيراً إنشاء حلقة التدريب

```
rewards_all_episodes = []

for episode in range(num_episodes):
    state = env.reset()
    done = False
    rewards_current_episode = 0
    for step in range(max_steps_per_episode):

        exploration_rate_threshold = random.uniform(0, 1)
        if exploration_rate_threshold > exploration_rate:
            action = np.argmax(q_table[state,:])
        else:
            action = env.action_space.sample()
        new_state, reward, done, info = env.step(action)

        q_table[state, action] = q_table[state, action] *\
                                 (1 - learning_rate) + \
                                 learning_rate *\
                                 (reward + discount_rate *\
                                 np.max(q_table[new_state, :]))
```

نكمل التعليمات الآتية بنفس إزاحة حلقة for الثانية

```
state = new_state
rewards_current_episode += reward
if done == True:
    break
```

أما التعليمتين التاليتين نكتبهما بنفس إزاحة حلقة for الأولى

```
exploration_rate = min_exploration_rate + \
                   (max_exploration_rate -\
                    min_exploration_rate) *\
                    np.exp(-exploration_decay_rate * episode)
rewards_all_episodes.append(rewards_current_episode)
```

(تم استخدام المعامل \ لتجزئة التعليمة الواحدة على أكثر من سطر)

قبل بداية حلقة التدريب، قمنا بإنشاء قائمة فارغة لتخزين مجموع جوائز (قيمة العائد) لكل حلقة كما في المثال السابق، في بداية كل حلقة سنقوم بتهيئة البيئة والعائد من جديد.

عند كل خطوة في الحلقة سنقوم بتوليد رقم عشوائي rate_threshold لتتم مقارنته مع exploration_rate لتحديد فيما إذا كان العميل سيقوم بالاكتشاف أو لا، إذا كان الرقم العشوائي هو الأكبر سيختار العميل الفعل الذي يعطي أكبر قيمة Q-value للحالة المتواجد بها في تلك الخطوة، وإلا سيختار العميل فعلا عشوائياً من الأفعال المتاحة في البيئة، يلي ذلك تنفيذ الفعل وتحديث Q_table باستخدام الصيغة المطروحة سابقاً ثم يتم الانتقال إلى الحالة الجديدة وجمع الجائزة المكتسبة في هذه الخطوة مع الجوائز المكتسبة الأخرى في الخطوات السابقة في نفس الحلقة.

بعدها نختبر شرط انتهاء الحلقة إذا كان محققاً نبدأ حلقة جديدة.

بعد الانتهاء من كل حلقة يتم إنقاص exploration_rate وإضافة جائزة الحلقة إلى قائمة جوائز باقي الحلقات.

يمكن طباعة Q_table لملاحظة تحديث القيم. **Error! Reference source not found.**

VI. الاستنتاجات

التعلم المعزز بات يشغل اليوم مساحة هامة في مجال تعلم الآلة لاسيما في المسائل التي تتطلب التفاعل مع البيئة واتخاذ القرار، لكن في مشاكل العالم الحقيقي لا يكون فضاء الحالات والأفعال صغيراً كما في المثال المطروح، مما يجعل استخدام خوارزمية Q-learning غير مجدية وذلك لأن Q-table سيشغل حجماً كبيراً من الذاكرة بالإضافة إلى استغراق البحث عن أفضل قيمة Q-value لفعل ما في حالة ما زمناً طويلاً جداً، لذا يتم دمج مفاهيم التعلم المعزز مع مفاهيم التعلم العميق للتمكن من إيجاد حلول فعالة لهذه المشاكل.

المراجع:

[1].
[2].
[3]. A. Géron, Hands-On Machine Learning with Scikit-Learn & Tensorflow, United States of America: O'Reilly, 2017.
[4]. R. Sutton and A. Barto, Reinforcement Learning: An Introduction, United States of America: The MIT Press, 2018.
[5]. OpenAI, "Environments," OpenAI, [Online]. Available: https://gym.openai.com/docs/. [Accessed 20 2 2021].
[6]. DeepLizard, "Reinforcement Learning- Goal Oriented Intelligence," [Online]. Available: https://deeplizard.com/. [Accessed 20 2 2021].

منشورات المؤلف(د. سامر سليمان):

[1]. Sulaiman, S., Lehnert, R., Dai, Q. (2009). Improved QoS-Aware Awitching Mechanism for PIM-SM Protocol. 3th IEEE IMSAA.
[2]. Türk, S., Sulaiman, S., Haidine, A., Michaelis, Th., Lehnert, R. (2009). Approaches for the migration of optical backbone networks towards Carrier Ethernet: GLOBECOM 2009/EFSOI09.
[3]. Sulaiman, S., Haidine, A., Lehnert, R. (2009). Performance Evaluation of Center Search Algorithms Used for Dynamic Rendevzous Point Relocation
[4]. Sulaiman, S., Haidine, A., Lehnert, R., Türk, S. (2009). Comparative Study of Multicast Protection Algorithms Using Shared Links in 100GET Transport Network.
[5]. Dai, Q., Lehnert, R. Sulaiman, S. (2008). An Adaptive Packet Dropping Algorithm for Improved VoIP Quality at ADSL.
[6]. Sulaiman, S. (2008). Optimization of Multicast Distribution Trees. Workshop der Fachgruppe 5.2.1, TU Dortmund.
[7]. Sulaiman, S. (2006). Investigation of End-2-End delay in IP-Multicast Networks: *Workshop der TU Dresden, der Universität Twente*.

[8]. Sulaiman, S. (2003). IP Multicast Routing Protocols: MMB/Dagstuhl Seminar Performance of MobileSystem, SchlossDagstuhl, 9.-12.
[9]. Baumann, M., Marandin, D., Sulaiman, S. (2002). Combined Modelling of TCP and MultiRED in DiffServ Networks: Proc. 2nd Polish-German Teletraffic Symposium PGTS 2002; Gdansk; 23.-24.9.
[10]. Baumann, M., Marandin, D., Sulaiman, S. (2005). Combined modelling of TCP and multiRED in DiffServ networks: European Transactions on Telecommunications, Vol. 16, No. 3, pp. 217-224.
[11]. Alkheir, J., Sulaiman, S., Mualla, R. (2020). Performance Evaluation on the Effect of Different Text Representation Models on the Image Captioning Systems. *Tishreen University Journal*, Vol. 42, No. 4, print ISSN: 2097-3081.
[12]. Alkubeily, M., Sulaiman, S., Esber, GM. (2021). Designing a Virtual Platform for Modeling Nodes in Wireless Sensor Networks at the Central Processing Unit Level. *Journal of Engineering Sciences and Information Technology*, Vol. 5, No. 5.
[13]. Alkubeily, M., Sulaiman, S., Esber, GM. (2021). Performance Evaluation of the Kernel Based Wireless Sensor Network Simulator Using an Authentication Algorithm. *Tishreen University Journal*, Vol. 43, No. 4.
[14]. Alkheir, J., Sulaiman, S., Mualla, R. (2021). Using Image Pre-classification to improve the accuracy of the image captioning systems. *Journal for the Engineering sciences*, Vol.37 No.2
[15]. Alkheir, J., Sulaiman, S., Mualla, R. (2020). Performance Evaluation on the Effect of Different Text Representation Models on the Image Captioning Systems. *Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series*, Vol. 42 No.4.
[16]. Alkubeily, M., Sulaiman, S., Esber, GM. (2020). WINDOWS FORM APPLICATION FOR VIRTUAL MINIMIZED PLATFORM KERNEL FOR WIRELESS SENSOR NETWORK SIMULATOR. *Far East Journal of Electronics and Communications*, Vol. 42 No.4.