# به نام خداوند رنگین کمان

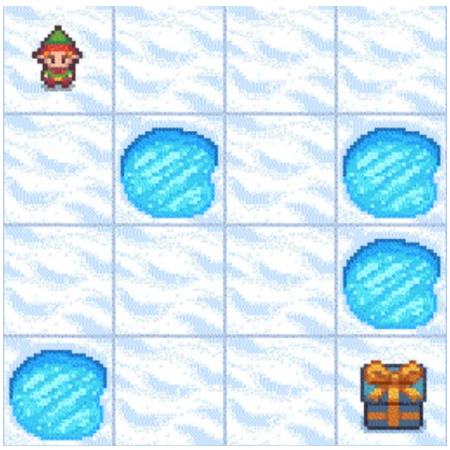
پروژه یادگیری تقویتی – مبانی هوش کلاسیک بهار ۱۴۰۲

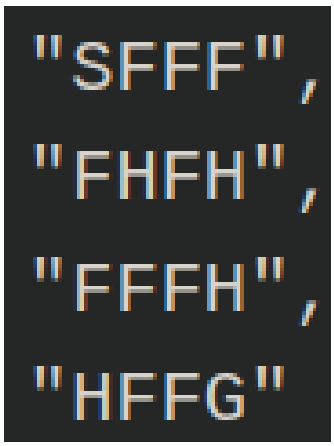
---

در این پروژه قصد داریم با پیاده‌سازی سه  الگوریتم:

- **Policy Iteration**
- **Monte Carlo (prediction – first visit)**
- **Monte Carlo (prediction – every visit)**

با استفاده از کتابخانه openAI  gym در محیط Frozen  Lake به بررسی و تاثیر پاداش (Reward)، ضریب تنزیل(Discount  factor  or  γ) و قطعیت(Deterministic) در چند محیط مختلف بپردازید. برای درک بهتر تعدادی متود جهت نمایش مقادیر سیاست و ارزش حالت‌ها از قبل پیاده سازی شده و می‌توانید از آن ها استفاده کنید.






**محیط Frozen Lake:**

شامل عبور از یک دریاچه یخ زده از Start(S) تا Goal(G) بدون افتادن در هیچ(Hole(H با قدم زدن بر روی دریاچه Frozen(F) است. در صورت افتادن در یکه سوراخ (H) عامل باید مجددا از خانه شروع (S) پیمایش را آغاز کند. به دلیل لغزنده بودن دریاچه یخ زده، عامل ممکن است همیشه در جهت مورد نظر حرکت نکند. عامل می‌تواند در چهار جهت چپ-پایین-راست-بالا حرکت کند. در صورتی که عامل حرکتی کند که سبب عبور از مرز محیط شود، موقعیت عامل تغییر نمی‌کند. با توجه به مسئله مورد بررسی، رسیدن به هر کدام از حالت‌های S و G و F دارای پاداش مشخص می‌باشد.(برای توضیحات بیشتر می‌توانید صفحه اصلی آن را مطالعه کنید)

## بخش اول – پیاده سازی الگوریتم‌ها

برای انجام پروژه ابتدا کتابخانه gym را نصب کنید. ورژن gym کدهای پیاده سازی شده 0.26.2 بوده اما در صورتی که قادر نبودید از این ورژن استفاده کنید، می‌توانید از ورژن‌های دیگر کتابخانه gym یا gymnasium نیز استفاده کنید. (دقت کنید که در صورتی که از ورژن دیگری استفاده کردید، احتمالا نیاز به تغییر قسمت‌های کمی از کد پیاده سازی شده دارید)

سپس کد قرار گرفته در درایو را دانلود و به پروژه خود اضافه کنید. همانطور که مشاهده می‌کنید، تمامی متودهای مورد نیاز جهت ارتباط با کتابخانه gym، متودهای نمایش سیاست و... از قبل پیاده سازی شده‌اند و می‌توانید با استفاده از آن‌ها پروژه خود را انجام دهید.

الگوریتم ابتدایی که باید پیاده‌سازی کنیدpolicy iteration می‌باشد. این الگوریتم با گرفتن محیط و داشتن دید کامل نسبت به محیط، با شروع از یک سیاست (policy) رندم می‌تواند به سیاست بهینه(optimal policy) دست یابد. این الگوریتم دو مرحله اصلی دارد:

1) **Policy Evaluation**
2) **Policy Improvement**

در بخش اول این الگوریتم، با استفاده از معادله بلمن (Bellman equation) و با توجه به سیاست مورد نظر به محاسبه ارزش هر حالت (state value) می‌پردازد .

در بخش دوم با انتخاب کُنش‌ای (action) که با استفاده از آن مقدار ارزش آن حالت بیشینه بشود، تلاش در بهبود سیاست خود می‌کنیم. سودوکد آن به صورت زیر می‌باشد:

**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   *old-action* $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

شما باید در قسمت کامنت گذاری شده متود *policy_iteration* این الگوریتم را پیاده سازی کنید:

```python
def policy_iteration(env, custom_map, max_ittr=30, theta=0.01, discount_factor=0.9):
    policy = get_init_policy(custom_map)   # it gives a random-walk policy
    V = np.zeros(env.observation_space.n)  # you can change it with any init value
    P = env.P                              # This attribute stores the transition probabilities
                                           # and rewards for each possible action in each possible
                                           # state of the environment.

    # loop till policy_stable becomes True or itter >= max_ittr
    ittr = 0
    policy_stable = False
    while not policy_stable and ittr < max_ittr:
        # policy evaluation

        # policy improvement

        ittr += 1
    return V, policy
```

**ورودی متود :**

- env : محیط پیاده سازی شده با استفاده از کتابخانه gym (توضیحات کامل آن در جلسه رفع اشکال پروژه ارائه خواهد شد)
- custom_map : نقشه مربوط به هر قسمت (این نقشه‌ها از قبل پیاده سازی شده اند و شما تنها باید با توجه به خواسته مسئله از آن ها استفاده کنید)
- max_ittr : بیشترین تعداد پیمایش حلقه درصورتی که شرط استیبل بودن سیاست ارضا نشود.
- theta : مقدار مثبت کوچکی که میزان دقت الگوریتم را تنظیم می‌کند.
- discount_factor : ضریب تنزیل و تاثیرگذار بر افق دید الگوریتم (گاما – γ)

**خروجی متود :**

- V : آرایه ارزش حالت‌ها
- policy : سیاست بهینه بدست آمده

- برای مطالعه بیشتر می‌توانید صفحه 80 کتاب Reinforcement Learning, an introduction (Sutton, Barto) -2018 را مطالعه کنید

الگوریتم دومی که باید پیاده‌سازی کنید first-visit Monte Carlo prediction می‌باشد. همانطور که می‌دانید ارزش یک حالت برابر با تخمین مقدار پاداش دریافتی (expected return) با شروع از آن حالت می‌باشد. یک راه واضح برای تخمین ارزش حالت از روی تجربه، صرفاً میانگین بازده پاداش دریافتی مشاهده شده پس از بازدید از آن حالت است. در صورتی که این بازده به سمت بی‌نهایت میل کند، میانگین باید به مقدار مورد انتظار همگرا شود. این ایده زیربنای تمام روش‌های مونت کارلو است.

در این الگوریتم نیز جمع آوری و محاسبه میانگین پاداش دریافتی بدست آمده از تجربه تحت پیروی از یک سیاست مشخص پس از اتمام اجرا (episode) و رسیدن به یک نقطه پایان (terminal)، نحوه تخمین ارزش حالت برای آن سیاست می‌باشد. همچنین دقت کنید این الگوریتم در هر اجر تنها اولین دفعه ای که به آن حالت رسید(First-visit) تخمین خود را انجام می‌دهد و در صورتی که مجدد آن حالت را مشاهده کرد، دیگر تغییری در مقدار ارزش آن حالت ایجاد نمی‌کند.

سودوکد آن به صورت زیر می‌باشد:

**First-visit MC prediction, for estimating $V \approx v_\pi$**

Input: a policy $\pi$ to be evaluated
Initialize:
    $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
    $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
    Generate an episode following $\pi$: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        Unless $S_t$ appears in $S_0, S_1, \ldots, S_{t-1}$:
            Append $G$ to $Returns(S_t)$
            $V(S_t) \leftarrow$ average$(Returns(S_t))$

همانطور که مشاهده می‌کنید دو بخش مهم در این الگوریتم وجود دارد:

1) Generate an episode following policy Π
2) loop for each step of episode , t= T-1, T-2, ..., 0

در مرحله اول با توجه به سیاست مورد نظر، شروع به پیمایش محیط کرده و اطلاعات مسیر پیمایش شده را ذخیره می‌کنیم. این کار را تا زمانی که به یک ترمینال نرسیده‌ایم ادامه می‌دهیم.

در مرحله دوم مقدار ارزش هر حالتی که در اجرای قبلی مشاهده کردیم را تخمین می‌زنیم. (هر حالت تنها می‌تواند یک بار در هر اپیزود آپدیت شود)

شما باید در قسمت کامنت گذاری شده متود first_visit_mc_prediction این ،الگوریتم را پیاده سازی کنید.

```
334  def first_visit_mc_prediction(env, policy, num_episodes, gamma):
335      # initilize
336      V = np.zeros(env.observation_space.n)
337      N = np.zeros(env.observation_space.n)
338
339      # loop in range num_episodes(for each episode)
340      # for i_episode in range(num_episodes):
341
342      # generate episode w.r.t policy
343
344      # loop for each step of episode , t= T-1, T-2, ..., 0
345
346      return V
```

**ورودی متود :**

- env : محیط پیاده سازی شده با استفاده از کتابخانه gym
- policy : سیاستی که الگوریتم شما با توجه به آن تجربه کسب کرده و شما تلاش در بدست آورد ارزش حالت‌های محیط با توجه به آن دارید
- num_episodes: تعداد دفعات اجرای حلقه برای ایجاد اپیزود
- gamma : ضریب تنزیل و تاثیرگذار بر افق دید الگوریتم (گاما – γ)

**خروجی متود :**

- V : آرایه ارزش حالت‌ها

- برای مطالعه بیشتر می‌توانید صفحه 114 کتاب Reinforcement Learning, an introduction (Sutton, Barto) -2018 را مطالعه کنید

الگوریتم سومی که باید پیاده‌سازی کنید *every-visit Monte Carlo prediction* می‌باشد. این الگوریتم همانند الگوریتم first-visit بوده با این تفاوت که تنها اولین بار رسیدن به آن حالت را مبنا محاسبه ارزش حالت آن حالت در نظر نگرفته و تمامی دفعات مشاهده شده آن حالت را در نظر می‌گیرد.

شما باید در قسمت کامنت گذاری شده متود *every_visit_mc_prediction*، این الگوریتم را پیاده سازی کنید:

```
351  def every_visit_mc_prediction(env, policy, num_episodes, gamma):
352      # initilize
353      V = np.zeros(env.observation_space.n)
354      N = np.zeros(env.observation_space.n)
355
356      # loop in range num_episodes(for each episode)
357      # for i_episode in range(num_episodes):
358
359      # generate episode w.r.t policy
360
361      # loop for each step of episode , t= T-1, T-2, ..., 0
362
363      return V
```

ورودی‌ها و خروجی‌های این متود همانند first-visit می‌باشند.

- برای مطالعه بیشتر می‌توانید صفحه 114 کتاب Reinforcement Learning, an introduction (Sutton, Barto) -2018 را مطالعه کنید

## بخش دوم – ارزیابی و بررسی الگوریتم‌ها

پس از پیاده سازی روش‌های خواسته شده، موارد زیر را بررسی کرده و برای نتایج بدست آمده در هر بخش استدلال ارائه کنید و آن ها را به صورت مناسبی گزارش کنید.

---

**۱.** با اجرای الگوریتم policy iteration با پارامترهای:

hole_reward=-0.1, goal_reward=1, move_reward=-0.1, theta=0.0001

بر روی custom_map_1 و با is_slippery=False مقدار پالیسی بهینه به همراه ارزش حالت را به ازای گاماهای :

discount_factor = [1, 0.9, 0.5, 0.1]

محاسبه کرده و در صورت وجود تفاوت بین نتایج حاصل از هر گاما، علت ایجاد تفاوت را گزارش کنید.

---

۲. مسئله قبل را برای custom_map_2 و با پارامترهای :

hole_reward=-4, goal_reward=10, move_reward=-0.9

مجدد بررسی و حل کنید. (بقیه پارامترها ثابت می‌باشند)

---

۳. با اجرای الگوریتم policy iteration با پارامترهای:

- hole_reward=-5, goal_reward=5, move_reward=-0.5, discount_factor=0.9, theta=0.0001

بر روی custom_map_3 و به ازای:

- is_slippery=False
- is_slippery=True

مقدار پالیسی بهینه به همراه ارزش حالت را محاسبه کرده و در صورت وجود تفاوت بین نتایج حاصل، استدلال مناسب ارائه کنید.

---

۴. مسئله قبل را برای custom_map_4 مجدد بررسی و حل کنید (بقیه پارامترها ثابت می‌باشند)

---

۵. اجرای الگوریتم policy iteration با پارامترهای:

hole_reward=-3, goal_reward=7, theta=0.0001, discount_factor=0.9

بـر روی custom_map_5 و بـا is_slippery=False مقـدار پالیسـی بهینـه بـه همـراه ارزش حالت را به ازای:

move_reward = [-4, -2, 0, 2]

محاسبه کرده و در صورت وجود تفاوت بین نتایج حاصل از هر پاداش، استدلال مناسب ارائه کنید.

---

6. مسئله قبل را برای custom_map_6 مجدد بررسی و حل کنید (بقیه پارامترها ثابت می‌باشند)

---

۷. با اجرای الگوریتم policy iteration با پارامتر های:

hole_reward=-2, goal_reward=50, move_reward=-1و theta=0.0001,
discount_factor=0.9

بـر روی custom_map_7 و بـا is_slippery=True مقـدار پالیسـی بهینـه بـه همـراه ارزش حالت را بدست آورید . بعد از آن سیاست بدست آمده را تحلیل کرده و رفتار عامل را در محیط توجیه کنید. سپس سیاست بهینه بدست آمده را به الگوریتم های

- first_visit_mc_prediction
- every_visit_mc_prediction

با gamma=0.9 و به ازای مقادیر:

num_episodes = [500, 5000]

داده و در نهـایت ارزش حـالت بدسـت آمـده از هـر سـه الگـوریتم را بـا یکـدیگر و مقایسـه و تحلیل کنید. همچنین علت اختلاف در هر بار اجرای الگوریتم های مونته کارلو را توضیح دهید. (به ازای گاماها و تعداد اپیزودهای یکسان، ممکن است ارزش حالت های هر اجرا با هم متفاوت باشند، علت آن را توضیح دهید)

۸. با استفاده از متود *get_policy_direction* سیاست‌های حرکت در سه جهت را برای custom_map_8 ایجاد کنید و سپس با استفاده از الگوریتم‌های:

- first_visit_mc_prediction
- every_visit_mc_prediction

و پارامترهای :

gamma=0.9, num_episodes=1000,  is_slippery=True

مقدار ارزش حالت هر پالیسی را تخمین زده و با یکدیگر مقایسه کنید. (راهنمایی: به تغییر دامنه کلی ارزش حالت و همچنین تفاوت ارزش حالت برای حالت‌هایی با ارزش بیشتر توجه بیشتری کنید)

## الزامات پروژه:

- فایل ارسالی باید شامل کدهای بخش اول و ارزیابی بخش دوم (بعد از پاسخگویی به هر بخش، کد آن بخش را کامنت کرده و بخش بعدی را در ادامه پیاده سازی کنید) به همراه گزارش بخش دوم باشد.
- حتما در گزارش ارسالی شماره دانشجویی هر دو نفر وجود داشته باشد.
- پاسخ‌های بخش دوم باید به صورت تایپی بوده و در گزارش آن از نمودارهای توزیع به دست آمده برای سیاست یا ارزش حالت خواسته شده در آن بخش و با استفاده از توابع آماده موجود در فایل اولیه، استفاده شده باشد.
- پاسخ‌های شما باید کامل، واضح و با ارائه استدلال باشد. از ارسال تنها نمودار خروجی هر بخش خودداری کنید.
- مهلت پروژه تا 24 خرداد می‌باشد.
- از کپی برداری و استفاده از تمارین دانشجویان دیگر به شدت خودداری کنید. درصورت مشاهده شباهت نامتعارف، به هر دو گروه نمره ۱۰۰- داده خواهد شد.

موفق باشید