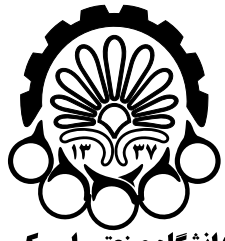
به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

مبانی و کاربردهای هوش مصنوعی (پاییز ۱۴۰۱)

گزارش پروژه - فاز چهارم

استاد درس:

دکتر جوانمردی

نام دانشجو:

محمدجواد رضوانیان

فاز چهارم، شکارچیان ارواح

۱. پاسخ سوال ۱، احتمال مشاهده

در این مرحله قصد داریم که احتمال مشاهده روح را محاسبه کنیم. برای اینکار باید متد `getObservationProb()` را پیاده‌سازی کنیم که موقعیت مکانی روح و موقعیت زندان روح را می‌گیرد و مقدار زیر را برمی‌گرداند:

$$P(noisyDistance \mid pacmanPosition, ghostPosition)$$

برای محاسبه احتمال بالا، باید در ابتدا چک کنیم که احتمال برابر با صفر است یا خیر. برای اینکار چک می‌کنیم که این مقدار برابر با None نباشد. اگر بود، باید مقدار این را برگردانیم که آیا روح در زندان است یا خیر. اگر در زندان باشد، دیگر نیازی به محاسبه نیست چون با احتمال ۱ مکان روح ما در زندان است! پس اگر `noisyDistance` برابر با `None` بود یعنی در زندان است و اگر برابر با None نبود، روح در زندان نیست. (شرط دوم در تابع)

۱.۱ بررسی کد

تابع مورد نظر به صورت زیر پیاده‌سازی شده است که بتواند هر دو شرطی که در قسمت قبلی ذکر شد را چک کند.

```python
    def getObservationProb(self, noisyDistance, pacmanPosition, ghostPosition, jailPosition):
        """
        Return the probability P(noisyDistance | pacmanPosition, ghostPosition).
        """
        "*** YOUR CODE HERE ***"
        if noisyDistance is None:
            return float(ghostPosition == jailPosition)
        if ghostPosition == jailPosition:
            return float(noisyDistance is None)
        estimatedDistance = manhattanDistance(pacmanPosition, ghostPosition)
        return busters.getObservationProbability(noisyDistance, estimatedDistance)
```

۲.۱ خروجی‌های آزمایش

خروجی کد مربوطه با استفاده از دستور داده شده به صورت زیر خواهد بود:

```
python autograder.py -q q1
```

```
P4 python autograder.py -q q1
D:\Projects\Python\Pacman World\P4\autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib and slated for removal in Python 3.12; see th
e module's documentation for alternative uses
  import imp
Starting on 1-24 at 19:21:07

Question q1
===========
*** PASS: test_cases\q1\1-ObsProb.test
***     PASS

### Question q1: 2/2 ###


Finished at 19:21:07

Provisional grades
==================
Question q1: 2/2
------------------
Total: 2/2

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

همانطور که مشاهده می‌کنید، عامل ما نمره ۲ از ۲ را برای این سوال کسب کرده است.

۲. مشاهده استنتاج دقیق

در ابتدا برای حل این مسئله، موقعیت مکانی عامل و زندان‌ها را نگه می‌داریم. باید باور عامل خود را که با استفاده از تابع زیر بدست آمده، نگهداری کنیم:

$$P(ghostPosition_t \mid noisyDistance_1, noisyDistance_2, \dots, noisyDistance_t)$$

حالا باید بتوانیم معادله بالا را پیاده‌سازی کنیم.

۱.۲ بررسی کد

برای محاسبه مقدار باور بالا، باید از تابعی که در سوال اول کامل کردیم کمک بگیریم. ما برای بروزرسانی تمام موقعیت‌ها، لازم داریم که احتمال تمام موقعیت‌ها را محاسبه کنیم (حلقه‌ی for) و سپس مقدار آن را در باور عامل ذخیره کنیم.

```
    def observeUpdate(self, observation, gameState: GameState):
        "*** YOUR CODE HERE ***"
        pacmanPosition = gameState.getPacmanPosition()
        jailPosition = self.getJailPosition()
        beliefs = self.beliefs  # P(ghostPosition[t] | noisyDist[1:t])
        for position in self.allPositions:
            pr_noisy_given_real = self.getObservationProb(
                observation, pacmanPosition, position, jailPosition)
            beliefs[position] *= pr_noisy_given_real
        self.beliefs.normalize()
```

۲.۲ خروجی

خروجی قطعه کد ما بعد از اجرای دستور آمده در دستور کار به صورت زیر خواهد بود:

```
python autograder.py -q q۲ --no-graphics
```

```
P4 python autograder.py -q q2 --no-graphics
D:\Projects\Python\Pacman World\P4\autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib and slated for removal in Python 3.12; see th
e module's documentation for alternative uses
  import imp
Starting on 1-24 at 19:48:54

Question q2
===========
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\1-ExactUpdate.test
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\2-ExactUpdate.test
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\3-ExactUpdate.test
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\4-ExactUpdate.test

### Question q2: 3/3 ###


Finished at 19:48:54

Provisional grades
==================
Question q2: 3/3
------------------
Total: 3/3

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

۳. استنتاج دقیق با گذشت زمان

در اینجا، هدف ما کامل کردن باور عامل است. ما می‌خواهیم احتمال اینکه موقعیت بعدی با توجه به موقعیت قبلی چه قدر است را محاسبه کنیم. در واقع می‌خواهیم ببینیم که مقدار احتمال زیر چه قدر است:

$$P(next\ position_{t+1} \mid current\ position_t)$$

برای محاسبه این مقدار، لازم است که از تابع `self.getPositionDistribution()` استفاده کنیم. تابع انتقال ما به صورت یک آرایه‌ای از موقعیت‌ها و احتمال آنها نگهداری می‌شود. تابع انتقال ما، یک آرایه‌ی دوبعدی است که بعد اول آن موقعیت فعلی و بعد بعدی موقعیت بعدی است و مقدار آن برابر با احتمال محاسبه شده توسط معادله بالا است.

حالا باید با استفاده از احتمال بدست آمده و باور قبلی، مقدار باور جدید را بروزرسانی کنیم و مقدار آن را به ازای موقعیت‌های متفاوت حساب کنیم.

یعنی می‌توان مقدار باور جدید را به صورت زیر بدست آورد (مقدار t برابر با باورهای عامل است):

$$belief_t = \sum_t \left(P(next\ position_{t+1} \mid current\ position_t) * (old\ belief_t)\right)$$

۱.۳ بررسی کد

تابع `elapseTime()` بعد از پیاده‌سازی به صورت زیر درخواهد آمد:

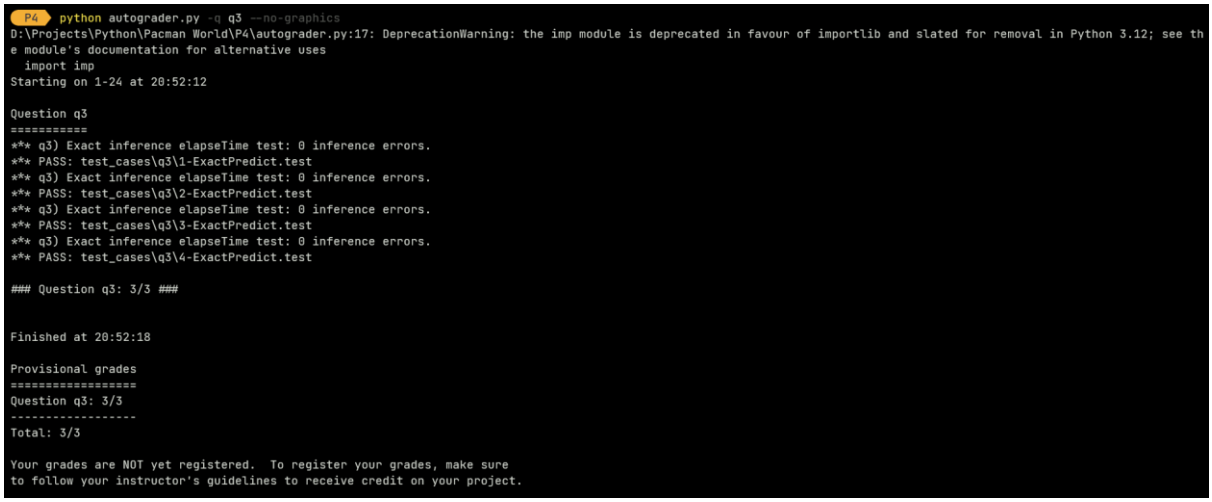
```
def elapseTime(self, gameState):
        "*** YOUR CODE HERE ***"
        beliefs = self.beliefs
        new_beliefs = DiscreteDistribution()
        pr_transition = {}

        for position in self.allPositions:
            pr_transition[position] = self.getPositionDistribution(gameState, position) # P(to | from)

        for position in self.allPositions:
            new_beliefs[position] = sum(
                pr_transition[prevPosition][position] * beliefs[prevPosition]
                for prevPosition in beliefs
            )
        self.beliefs = new_beliefs
```

٥

۲.۳ خروجی

خروجی پروژه ما به صورت زیر خواهد بود:

```
python autograder.py -q q۳ --no-graphics
```



همانطور که مشاهده می‌کنید عامل ما توانسته است که تمام ۳ تست را پاس کند.

۴. استنتاج دقیق با تست کامل

در ابتدا باید یک لیست از نزدیک‌ترین روح‌ها بدست بیاوریم. سپس فاصله‌ی روح‌ها را تا عامل پک‌من در موقعیت فعلی بدست بیاوریم. سپس بعد از بررسی احتمال بعد از اقدامات مجاز، باید دوباره فاصله‌ی آن را محاسبه کرد. باید کمترین مقدار ممکن را بدست بیاوریم. با توجه به اینکه جست‌وجوی ما از نوع حریصانه است، همین که نزدیک ترین روح را پیدا کنیم به سمت آن حرکت می‌کنیم.

۱.۴ بررسی کد

برای سهولت در پیاده‌سازی، تابع `self.distancer.getDistance()` را با تابع `dist()` جایگزین می‌کنیم. برای تمیزی کد، دو تابع داخلی به اسم `dist-from-pacman()` برای محاسبه فاصله از عامل ما و تابع دیگری به اسم `distance-from-closest-ghost-after-action()` برای محاسبه فاصله بعد از انجام اقدام تعریف می‌کنیم.

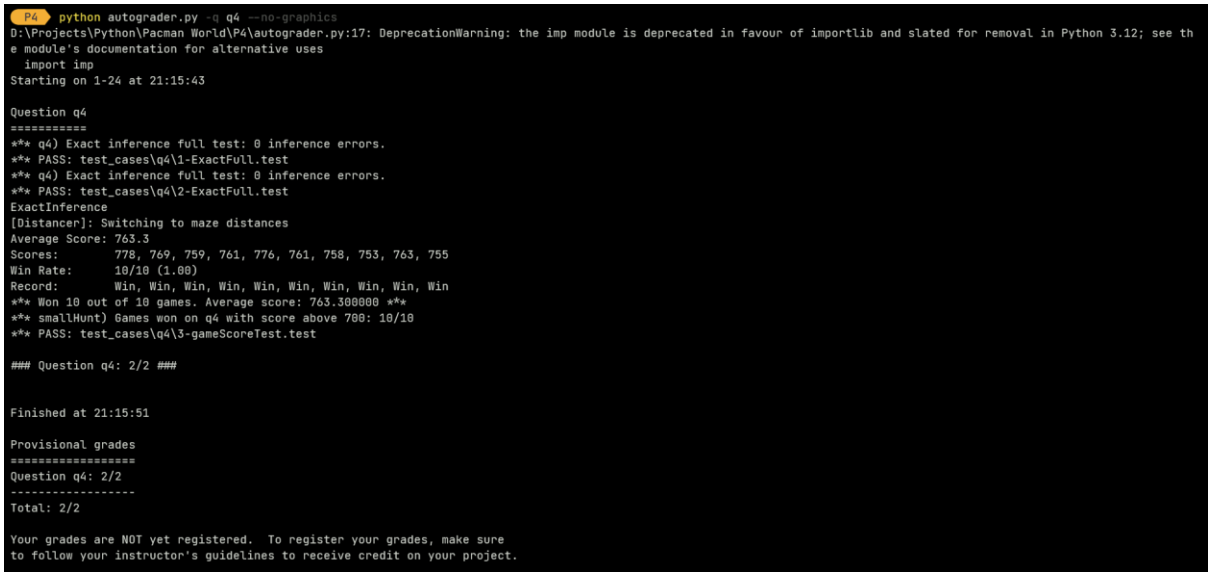
برای پیدا کردن نزدیک‌ترین احتمال وجود روح، از عبارت `distribution.argMax()` استفاده می‌کنیم.

```
    def chooseAction(self, gameState):
        pacmanPosition = gameState.getPacmanPosition()
        legalActions = [a for a in gameState.getLegalPacmanActions()]
        livingGhosts = gameState.getLivingGhosts()
        livingGhostPositionDistributions = \
            [beliefs for i, beliefs in enumerate(self.ghostBeliefs)
             if livingGhosts[i + 1]]
        "*** YOUR CODE HERE ***"
        dist = self.distancer.getDistance
        ghostPositions = [distribution.argMax()
                          for distribution in livingGhostPositionDistributions]

        def dist_from_pacman(ghostPosition):
            return dist(pacmanPosition, ghostPosition)

        closestGhostPosition = min(ghostPositions, key=dist_from_pacman)

        def distance_from_closest_ghost_after_action(action):
            newPacmanPosition = Actions.getSuccessor(pacmanPosition, action)
            return dist(newPacmanPosition, closestGhostPosition)

        greedyAction = min(legalActions, key=distance_from_closest_ghost_after_action)
        return greedyAction
```

۲.۴ خروجی

خروجی ما بعد از اجرای دستور آمده در دستورکار، به صورت زیر است:

```
python autograder.py -q q٤ --no-graphics
```



می‌بینیم که عامل ما توانسته است که هر دو تست را پاس کند و میانگین نمره کسب شده برابر با ۷۶۳ است و هر ۱۰ بار برنده شده است.

۵. تقریب استنتاج اولیه و باورها

در ابتدا باید ذرات را مقداردهی کرد. برای اینکار باید بتوانیم که مقادیر را به صورت برابر در موقعیت‌های مجاز توزیع کنیم. (برای اینکار با یک سرچ کوچک در اینترنت و مشاهده‌ی نمونه کارهای مختلف، با این روش حال کردم 😅)

۱.۵ بررسی کد

تابع `initializeUniformly()` که کار خاصی ندارد و فقط باید ذرات را مقداردهی کند. تابع کمکی به نام `evenlyDistributedParticles()` تعریف میکنیم تا بتواند با روشی که در اینترنت پیدا کردیم ذرات را به صورت uniform توزیع کند. تابع `getBeliefDistribution()` هم وظیفه‌اش این است که لیستی از ذرات را تبدیل به شیء از جنس `DiscreteDistribution` کند. کد پیاده‌سازی توابع مورد نظر به صورت زیر است:

```
    def initializeUniformly(self, gameState):
        "*** YOUR CODE HERE ***"
        self.particles = evenlyDistributedParticles(self.numParticles, self.legalPositions)

def evenlyDistributedParticles(numParticles, legalPositions):
    numCopies = numParticles // len(legalPositions)
    numRemaining = numParticles % len(legalPositions)
    particles = legalPositions * numCopies
    particles += random.sample(legalPositions, k=numRemaining)
    return particles

    def getBeliefDistribution(self):
        "*** YOUR CODE HERE ***"
        distribution = DiscreteDistribution(Counter(self.particles))
        distribution.normalize()
        return distribution
```

۲.۵ خروجی

خروجی کد ما با استفاده از دستور مذکور در دستور کار به صورت زیر است:

```
python autograder.py -q q۵
```

```
P3 python autograder.py -q q5
D:\Projects\Python\Pacman World\P3\autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib and slated for removal in Python 3.12; see th
e module's documentation for alternative uses
  import imp
Starting on 1-21 at 21:56:58

Question q5
===========

*** PASS: test_cases\q5\1-tinygrid.test
*** PASS: test_cases\q5\2-tinygrid-noisy.test
*** PASS: test_cases\q5\3-bridge.test
*** PASS: test_cases\q5\4-discountgrid.test

### Question q5: 3/3 ###


Finished at 21:56:58

Provisional grades
==================
Question q5: 3/3
------------------
Total: 3/3

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

۶. مشاهده استنتاج تقریبی

در ابتدا نیاز داریم که بررسی کنیم که آیا مجموع باورهای ما برابر با صفر است یا خیر (همان حالت خاص)؛ اگر وزن ذرات ما برابر با صفر باشد باید با استفاده از تابعی که در قسمت قبل تعریف کردیم، ذرات را دوباره به طور یکنواخت توزیع کنیم. پس باید احتمال تمام ذرات را داشته باشیم و سپس به وسیله آن با استفاده از وزن ذرات توزیع وزنی آن ها را ایجاد کنیم.

۱.۶ بررسی کد

در ابتدا باید یک احتمال با توجه به موقعیت پکمن، یک موقعیت بالقوه روح و موقعیت زندان استفاده کنیم. برای اینکار از همان تابع `self.getObservationProb()` استفاده می‌کنیم تا وزن ذرات را بدست بیاوریم. سپس با استفاده از شرط پایین حلقه (`if beliefs.total() == ۰`) وزن ذرات را چک می‌کنیم که صفر نباشد. حالا که مطمئن شدیم که مقدار توزیع ذرات صفر نیست، حالا باید لیست جدید را بسازیم.

```
    def observeUpdate(self, observation, gameState: GameState):
        "*** YOUR CODE HERE ***"
        pacmanPosition = gameState.getPacmanPosition()
        jailPosition = self.getJailPosition()
        beliefs = self.getBeliefDistribution()

        for particle in beliefs:
            beliefs[particle] *= self.getObservationProb(observation, pacmanPosition, particle,
jailPosition)

        if beliefs.total() == 0:
            self.initializeUniformly(gameState)
        else:
            self.particles = random.choices(list(beliefs), k=self.numParticles,
weights=list(beliefs.values()))
```

۲.۶ خروجی

خروجی ما با توجه به اجرای دستور مقابل، به صورت زیر خواهد بود:

```
python autograder.py -q q٦ --no-graphics
```

```
Question q6
===========
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\1-ParticleUpdate.test
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\2-ParticleUpdate.test
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\3-ParticleUpdate.test
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\4-ParticleUpdate.test
*** q6) successfully handled all weights = 0
*** PASS: test_cases\q6\5-ParticleUpdate.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 174.8
Scores:        186, 185, 192, 165, 189, 169, 175, 189, 188, 190
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** Won 10 out of 10 games. Average score: 174.800000 ***
*** oneHunt) Games won on q6 with score above 100: 10/10
*** PASS: test_cases\q6\6-ParticleUpdate.test

### Question q6: 3/3 ###


Finished at 22:05:43

Provisional grades
==================
Question q6: 3/3
------------------
Total: 3/3

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

۷. استنتاج تقریبی با گذشت زمان

باید در ابتدا یک لیستی از ذرات داشته باشیم تا مقادیر جدید را در آن ذخیره کنیم. سپس باید با توجه به وزن ذرات، یک نمونه‌برداری از آنها انجام دهیم و مقادیر جدید با مقادیر قبلی جایگزین کنیم.

۱.۷ بررسی کد

در ابتدا یک لیست خالی برای ذرات جدید می‌سازیم. به ازای تمام ذرات موجود باید توزیع آنها را بدست بیاوریم و با استفاده از تابع `sample()` یک نمونه‌برداری با توجه به وزن ذرات انجام دهیم. حالا باید موقعیت جدید را در لیست ذرات اضافه کنیم و در نهایت مقادیر را با مقادیر اولیه جابجا کنیم.

```
    def elapseTime(self, gameState):
        "*** YOUR CODE HERE ***"
        particles = []
        for particle in self.particles:
            nextPositionDistribution = self.getPositionDistribution(gameState, particle)
            nextPosition = nextPositionDistribution.sample()
            particles.append(nextPosition)
        self.particles = particles
```

۲.۷ خروجی

خروجی ما بعد از اجرای دستور روبه‌رو به صورت زیر است:

```
python autograder.py -q q۷ --no-graphics
```

```
Starting on 1-24 at 22:10:41

Question q7
===========
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\1-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\2-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\3-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\4-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\5-ParticlePredict.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 378.6
Scores:        377, 374, 379, 382, 381
Win Rate:      5/5 (1.00)
Record:        Win, Win, Win, Win, Win
*** Won 5 out of 5 games. Average score: 378.600000 ***
*** smallHunt) Games won on q7 with score above 300: 5/5
*** PASS: test_cases\q7\6-ParticlePredict.test

### Question q7: 3/3 ###


Finished at 22:11:17

Provisional grades
==================
Question q7: 3/3
------------------
Total: 3/3

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

۸. مشاهدات مشترک فیلتر ذرات

هیچ توضیح خاصی ندارم برای این قسمت.

۱.۸ بررسی کد

برای توزیع ذرات از همان تابع `evenlyDistributedParticles()` استفاده می‌کنیم. با استفاده از این تابع، دیگر نیازی به استفاده از `itertools.product()` برای ضرب کارتزین نیست. سپس با استفاده از `shuffle()` میتوانیم ترتیب تصادفی را بدست بیاوریم. سپس باید لیست ذرات را بروز کنیم.

```
    def initializeUniformly(self, gameState):
        "*** YOUR CODE HERE ***"
        positionsForEachGhost = [ evenlyDistributedParticles(self.numParticles, self.legalPositions)
for _ in range(self.numGhosts) ]
        for positions in positionsForEachGhost[1:]:
            random.shuffle(positions)
        self.particles = list(zip(*positionsForEachGhost))
```

۲.۸ خروجی

```
P4 python autograder.py -q q8 --no-graphics
D:\Projects\Python\Pacman World\P4\autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib and slated for removal in Python 3.12; see th
e module's documentation for alternative uses
  import imp
Starting on 1-24 at 22:22:41

Question q8
===========
*** q8) Joint particle filter initialization test: 0 inference errors.
*** PASS: test_cases\q8\1-JointParticleInit.test

### Question q8: 1/1 ###


Finished at 22:22:41

Provisional grades
==================
Question q8: 1/1
------------------
Total: 1/1

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

۹. مشاهدات مشترک فیلتر ذرات (بخش دوم)

برای اینکار، باید به ازای تمام باورها، تمام روحها را مورد پیمایش قرار دهیم. سپس احتمال و وزن آن را محاسبه کنیم و عملیات نمونه‌برداری را مجددا انجام دهیم. حالت خاصی که در اینجا ممکن است رخ دهد این است که وزن همه ذرات صفر باشد که دوباره باید مثل قبل عمل کنیم.

۱.۹ بررسی کد

برای حل این مسئله، باید یکبار روی ذرات و یکبار روی روحها پیمایش انجام دهیم و موقعیت زندان هر روح را به صورت مجزا محاسبه کنیم. باقی مسئله مثل قبل می‌باشد و تنها فرق آن، قسمت زندان و جست‌وجوی همزمان روی تمام روحها است.

```
    def observeUpdate(self, observation, gameState):
        "*** YOUR CODE HERE ***"
        pacmanPosition = gameState.getPacmanPosition()
        jailPositions = [self.getJailPosition(i) for i in range(self.numGhosts)]
        beliefs = self.getBeliefDistribution()
        for particle in beliefs:
            for i in range(self.numGhosts):
                beliefs[particle] *= self.getObservationProb(observation[i],
                                                             pacmanPosition,
                                                             particle[i],
                                                             jailPositions[i])
        if beliefs.total() == 0:
            self.initializeUniformly(gameState)
        else:
            self.particles = random.choices(list(beliefs), k=self.numParticles,
                                            weights=list(beliefs.values()))
```

۲.۹ خروجی

خروجی کد ما به صورت زیر است.

```
P4 python autograder.py -q q9 --no-graphics
D:\Projects\Python\Pacman World\P4\autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib and slated for removal in Python 3.12; see th
e module's documentation for alternative uses
  import imp
Starting on 1-24 at 22:41:23

Question q9
===========
*** q9) Joint particle filter observeUpdate test: 0 inference errors.
*** PASS: test_cases\q9\1-JointParticleUpdate.test
*** q9) Joint particle filter observeUpdate test: 0 inference errors.
*** PASS: test_cases\q9\2-JointParticleUpdate.test
*** q9) successfully handled all weights = 0
*** PASS: test_cases\q9\3-JointParticleUpdate.test
*** q9) Joint particle filter observeUpdate test: 0 inference errors.
*** PASS: test_cases\q9\4-JointParticleUpdate.test

### Question q9: 3/3 ###


Finished at 22:41:24

Provisional grades
==================
Question q9: 3/3
------------------
Total: 3/3

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

بدلیل اینکه باید دوحلقه تو در تو را پیمایش کنیم، زمان محاسبه به طور چشم‌گیری افزایش می‌یابد (نزدیک به ۲ ثانیه بیشتر)

۱۰. زمان سپری شده فیلتر ذرات مشترک و تست کامل

۱.۱۰ بررسی کد

تنها کاری که باید انجام دهیم، این است که موقعیت قبلی ذرات (روح‌ها) را به همراه شاخص روح (i) به تابع محاسبه توزیع مکان داده تا با موقعیت جدید هر روح را به شرط موقعیت قبلی همه روح‌های صفحه به ما بدهد و آن را در متغیر nextPosDistribution ذخیره کرده و سپس آن را همانگونه که صورت سوال خواسته برگردانیم.

```
    def elapseTime(self, gameState):
        newParticles = []
        for oldParticle in self.particles:
            newParticle = list(oldParticle)  # A list of ghost positions

            # now loop through and update each entry in newParticle...
            "*** YOUR CODE HERE ***"
            for i in range(self.numGhosts):
                nextPosDistribution = self.getPositionDistribution( gameState, oldParticle, i,
self.ghostAgents[i])
                newParticle[i] = nextPosDistribution.sample()
            """*** END YOUR CODE HERE ***"""
            newParticles.append(tuple(newParticle))
        self.particles = newParticles
```

۲.۱۰ خروجی

خروجی کد ما به صورت زیر است:

```
P4 python autograder.py -q q10 --no-graphics
D:\Projects\Python\Pacman World\P4\autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib and slated for removal in Python 3.12; see th
e module's documentation for alternative uses
  import imp
Starting on 1-24 at 22:43:01

Question q10
============
*** q10) Joint particle filter elapseTime test: 0 inference errors.
*** PASS: test_cases\q10\1-JointParticlePredict.test
*** q10) Joint particle filter elapseTime test: 0 inference errors.
*** PASS: test_cases\q10\2-JointParticlePredict.test
*** q10) Joint particle filter elapseTime test: 0 inference errors.
*** PASS: test_cases\q10\3-JointParticleFull.test

### Question q10: 3/3 ###


Finished at 22:43:46

Provisional grades
==================
Question q10: 3/3
------------------
Total: 3/3

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```