Învățare Automată - Laboratorul 4

Metode de programare dinamică pentru rezolvarea proceselor markov de decizie

Tudor Berariu

Facultatea de Automatică și Calculatoare

Scopul laboratorului

- Scopul laboratorului îl reprezintă înțelegerea conceptelor de proces markov de decizie, politică, valoare de stare, precum și implementarea unor metode de programare dinamică pentru rezolvarea problemei de control a unui MDP.

- În cadrul laboratorului veți:
  1. implementa algoritmul de iterare a politicilor;
  2. implementa algoritmul de iterare a valorilor de stare.

Proces Markov de decizie finit. Politică

- Un proces Markov de decizie finit este un obiect matematic compus din:
  - o mulțime finită de stări $\mathcal{S}$
    - submulțimea $\mathcal{S}^- \subset \mathcal{S}$ reprezintă stările neterminale.
  - o mulțime finită de acțiuni $\mathcal{A}$
  - o mulțime finită de valori, numite recompense $\mathcal{R}$
  - o funcție $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{R} \to [0, 1]$ ce descrie dinamica mediului:

$$p(s, a, s', r) \overset{not.}{=} p(s', r|s, a) = P\left(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\right)$$

  - un factor de atenuare $\gamma \in [0, 1]$.
- O politică deterministă reprezintă o funcție $\pi : \mathcal{S}^- \to \mathcal{A}$ care indică o acțiune pentru fiecare stare neterminală.
- Un proces Markov de decizie finit și o politică induc traiectorii de tipul:

$$S_0, A_0, R_1, S_1, A_1, \ldots, S_t, A_t, R_{t+1}, S_{t+1}, \ldots, R_T, S_T$$

Valoare de stare

- Câștigul mediu dintr-o stare $s \in \mathcal{S}$ indus de o politică π:

$$v^{\pi}(s) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \middle| S_t = s\right] = \mathbb{E}_{\pi}\left[\sum_{\tau=t+1} \gamma^{\tau-t-1} R_{\tau} \middle| S_t = s\right]$$

$$= \mathbb{E}_{\pi}\left[G_t | S_t = s\right]$$

- Relațiile dintre valorile de stare induse de o politică deterministă (ecuațiile Bellman):

$$v^{\pi}(s) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, \pi(s)) \left[r + \gamma v^{\pi}(s')\right]$$

- Scopul învățării prin recompensă îl reprezintă găsirea unei politici optime π^*:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\pi}\left[G_t\right] = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\pi}\left[v^{\pi}(S_t)\right]$$

Evaluarea unei politici

- Problema evaluării unei politici presupune determinarea funcției valoare $v^\pi : \mathcal{S} \to \mathbb{R}$ indusă de acea politică.

$$v^\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

- Algoritmul **Policy Evaluation** construiește o estimare a acestei funcții pentru o politică detrministă.

$$v(s) \approx v^\pi(s), \forall s \in \mathcal{S}$$

procedure PolicyEvaluation($\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma \rangle, \pi, \epsilon$)
  for all $s \in \mathcal{S}$ **do**
    $v(s) \leftarrow 0$ ▷ Valorile inițiale sunt zero.
  end for
  repeat
    $\delta \leftarrow 0$
    for all $s \in \mathcal{S}^-$ **do**
      $v_{old} \leftarrow v(s)$
      $v(s) \leftarrow \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, \pi(s)) \left[r + \gamma v(s') \right]$
      $\delta \leftarrow \max(\delta, |v(s) - v_{old}|)$
    end for
  until $\delta < \epsilon$
  return v
end procedure

Iterarea politicilor

- Fiind dată o politică deterministă π și funcția valoare v^π indusă de aceasta se poate obține o politică mai bună π' alegând acțiunile în mod lacom în raport cu v^π.

$$\pi'(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v(s') \right]$$

- Algoritmul **Policy Iteration** găsește o politică optimă alternând pași de evaluare și de îmbunătățire a politicii.

$$\pi_0 \rightarrow v^{\pi_0} \rightarrow \pi_1 \rightarrow v^{\pi_1} \rightarrow \pi_2 \rightarrow \ldots \rightarrow \pi^*$$

procedure PolicyIteration($\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma \rangle, \epsilon$)
  for all $s \in \mathcal{S}$ **do**
    $\pi(s) \leftarrow random(\mathcal{A})$
    $v(s) \leftarrow 0$
  end for
  repeat
    repeat ▷ Se evaluează politica curentă.
      $\delta \leftarrow 0$
      for all $s \in \mathcal{S}^-$ **do**
        $v_{old} \leftarrow v(s)$
        $v(s) \leftarrow \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, \pi(s)) \left[r + \gamma v(s') \right]$
        $\delta \leftarrow \max(\delta, |v(s) - v_{old}|)$
      end for
    until $\delta < \epsilon$
    $done \leftarrow True$
    for all $s \in \mathcal{S}^-$ **do** ▷ Se îmbunătățește politica.
      $a_{old} \leftarrow \pi(s)$
      $\pi(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma v(s') \right]$
      $done \leftarrow done \wedge (a_{old} = \pi(s))$
    end for
  until $done$
  return π, v
end procedure

Iterarea politicilor

- Fiind dată o politică deterministă π și funcția valoare v^π indusă de aceasta se poate obține o politică mai bună π' alegând acțiunile în mod lacom în raport cu v^π.

$$\pi'(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v(s') \right]$$

- Algoritmul **Policy Iteration** găsește o politică optimă alternând pași de evaluare și de îmbunătățire a politicii.

$$\pi_0 \rightarrow v^{\pi_0} \rightarrow \pi_1 \rightarrow v^{\pi_1} \rightarrow \pi_2 \rightarrow \ldots \rightarrow \pi^*$$

procedure PolicyIteration($\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma \rangle, \epsilon$)
  for all $s \in \mathcal{S}$ **do**
    $\pi(s) \leftarrow random(\mathcal{A})$ ▷ Politica inițială este aleatoare.
    $v(s) \leftarrow 0$ ▷ Valorile inițiale sunt zero.
  end for
  repeat
    repeat ▷ Se evaluează politica curentă.
      $\delta \leftarrow 0$
      for all $s \in \mathcal{S}^-$ **do**
        $v_{old} \leftarrow v(s)$
        $v(s) \leftarrow \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, \pi(s)) \left[r + \gamma v(s') \right]$
        $\delta \leftarrow \max(\delta, |v(s) - v_{old}|)$
      end for
    until $\delta < \epsilon$
    $done \leftarrow True$
    for all $s \in \mathcal{S}^-$ **do** ▷ Se îmbunătățește politica.
      $a_{old} \leftarrow \pi(s)$
      $\pi(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma v(s') \right]$
      $done \leftarrow done \wedge (a_{old} = \pi(s))$
    end for
  until $done$ ▷ se oprește atunci când politica devine stabilă.
  return π, v
end procedure

Iterarea valorilor

- Algoritmul de iterare a politicilor (slide-ul anterior) alocă efort computațional pentru aproximarea cu precizie a funcției de valoare pentru fiecare politică.
- Algoritmul poate fi modificat prin reducerea acestui efort fără a pierde garanția convergenței cobinând într-o singură operație un pas de evaluare a politicii și unul de îmbunătățire a acesteia.

procedure ValueIteration($\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma \rangle, \epsilon$)
  for all $s \in \mathcal{S}$ **do**
    $v(s) \leftarrow 0$
  end for
  repeat
    $\delta \leftarrow 0$
    for all $s \in \mathcal{S}^-$ **do**
      $v_{old} \leftarrow v(s)$
      $v(s) \leftarrow \max_a \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) [r + \gamma v(s')]$
      $\delta \leftarrow \max(\delta, |v(s) - v_{old}|)$
    end for
  until $\delta < \epsilon$
  for all $s \in \mathcal{S}^-$ **do** ▷ Se extrage politica optimă
    $\pi(s) \leftarrow \underset{a \in \mathcal{A}}{\text{argmax}} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) [r + \gamma v(s')]$
  end for
  return π, v
end procedure

Iterarea valorilor

- Algoritmul de iterare a politicilor (slide-ul anterior) alocă efort computațional pentru aproximarea cu precizie a funcției de valoare pentru fiecare politică.
- Algoritmul poate fi modificat prin reducerea acestui efort fără a pierde garanția convergenței cobinând într-o singură operație un pas de evaluare a politicii și unul de îmbunătățire a acesteia.

procedure ValueIteration($\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma \rangle, \epsilon$)
  for all $s \in \mathcal{S}$ **do**
    $v(s) \leftarrow 0$
  end for
  repeat
    $\delta \leftarrow 0$
    for all $s \in \mathcal{S}^-$ **do**
      $v_{old} \leftarrow v(s)$
      $v(s) \leftarrow \max_a \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) [r + \gamma v(s')]$
      $\delta \leftarrow \max(\delta, |v(s) - v_{old}|)$
    end for
  until $\delta < \epsilon$
  for all $s \in \mathcal{S}^-$ **do** ▷ Se extrage politica optimă
    $\pi(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) [r + \gamma v(s')]$
  end for
  return π, v
end procedure

Problema de rezolvat

- Un agent se mișcă pe o hartă bidimensională.
- La fiecare moment de timp agentul se află într-o celulă și poate alege o acțiune de deplasare către una din cele patru direcții: nord, est, sud, sau vest.
- Efectele unei acțiuni sunt stocastice.
  - Acțiunea agentului reușește cu probabilitatea .8.
  - Agentul va ajunge deviat cu 90° spre stânga cu probabilitatea .1 și deviat spre dreapta cu 90° cu probabilitatea .1.
  - Dacă agentul se îndrreaptă către un perete, atunci rămâne în celula curentă.
- Se cere găsirea unei politici optime pentru astfel de hărți.

Hărțile de test

Hartă simplă cu două stări finale.

```
A:-10
B:1
default:0
xxxxxxxxx
x       Bx
x  xxx  x
x    A  x
x       x
xxxxxxxxx
```

Hartă care cere să fii precaut.

```
A:-10
B:-10
C:1
default:0
xxxxxxxxx
x       Bx
x  xxx Cx
x    A  x
x       x
xxxxxxxxx
```

Mai bine mor decât să sufăr.

```
A:-1
B:1
default:-.5
xxxxxxxxx
x       Bx
x  xxx  x
x    A  x
x       x
xxxxxxxxx
```

Clasa `Maze`

```
class Maze:

    def __init__(self, map_name):
        ...

    @property
    def actions(self):
        ...

    @property
    def states(self):
        ...

    def is_final(self, state):
        ...

    def effects(self, state, action):
        ...
```

Metoda `effects` primește o stare s și o acțiune a și întoarce o listă de tupluri (s', p, r) cu următoarea semnificație:

$$p = P\left(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\right)$$

```
In [2]: m = Maze("simple")

In [3]: m.effects((1, 1), Maze.NORTH)
Out[3]: [((1, 1), 0.9, 0.0), ((1, 2), 0.1, 0.0)]

In [4]: m.effects((1, 2), Maze.SOUTH)
Out[4]: [((1, 3), 0.1, 0.0), ((2, 2), 0.8, 0.0), ((1, 1), 0.1, 0.0)]

In [7]: m.print_policy({s: choice(m.actions) for s in m.states if not m.is_final(s)})
...
```

Cerințe

1. Implementați algoritmul **Policy Iteration** pentru găsirea unei politici optime pentru jocul `Maze`.
2. Implementați algoritmul **Value Iteration** pentru găsirea unei politici optime pentru jocul `Maze`.

- Modificați codul din cele două funcții.

```
def policy_iteration(game, args):
    gamma = args.gamma
    max_delta = args.max_delta
    v = {s: 0 for s in game.states}
    policy = {s: choice(game.actions)
              for s in game.states if not game.is_final(s)}
    return policy, v
```

```
def value_iteration(game, args):
    gamma = args.gamma
    max_delta = args.max_delta
    v = {s: 0 for s in game.states}
    policy = {s: choice(game.actions)
              for s in game.states if not game.is_final(s)}
    return policy, v
```