Универзитет Црне Горе

Природно-математички факултет

Математички софтверски пакети

Функције infer_states и calculate_G_policies

Студент:
Никола Поповић 1/24 Б

Ментор:
мр Никола Пижурица

јун 2025.

Садржај

Увод

У оквиру Active Inferece-а, два процеса су од суштинске важности: инференција скривених стања на основу доступних опажања, и евалуација и селекција политика дјеловања у складу са агентовим циљевима и преференцијама. У програмској имплементацији, ова два процеса често одговарају функцијама infer_states и calculate_G_policies, које представљају централне компоненте у туторијалу Active Inference from Scratch.

Циљ овог рада јесте да пружи прецизно и детаљно математичко утемељење и начин функционисања ових функција у најједноставнијем случају — када агент има један модалитет опажања и један фактор скривеног стања. Оваква поставка представља добру полазну основу за разумијевање основне логике активне инференције и начина на који се врши процјена скривених стања и планирање дјеловања.

1 Математичка формализација проблема

У оквиру активне инференције, агент опажа и дјелује у окружењу на основу унутрашњег генеративног модела који одређује како се скривена стања свијета манифестују у облику опажаја и како се та стања мијењају у зависности од дјеловања агента.

1.1 Скривена стања и опажања

У моделу активне инференције, агент у сваком временском тренутку t добија сензорно опажање o_t и на основу њега врши процјену скривеног стања s_t. Та процјена представља унутрашњу представу свијета која омогућава агенту да разумије и предвиди своје окружење, као и да планира и доноси одлуке.

С обзиром на поједностављени случај са једним модалитетом и једним фактором скривеног стања, означавамо:

- o_t: опажање у времену t, при чему $o_t \in \mathcal{O}$, гдје је $\mathcal{O}$ коначни скуп свих могућих опажања
- s_t: скривено стање у времену t, и $s_t \in \mathcal{S}$, гдје имамо да је $\mathcal{S}$ коначан скуп могућих скривених стања.

1.2 Генеративни модел

Комплетан генеративни модел агента се састоји од:

- **Матрице зависности опажања од скривених стања (A):**

  Представља вјероватноћу опажања o_t за свако могуће скривено стање s_t:
  $$A = P(o_t \mid s_t)$$

- **Транзиционе матрице (B):**

  Дефинише прелазну вјероватноћу скривеног стања, у зависности од претходног стања и акције:
  $$B = P(s_t \mid s_{t-1}, u_{t-1})$$

- **Матрице преференци агента (C)**

  Агент додјељује вјероватноће сваком могућем опажању, које представљају његове преференце или жељена опажања:
  $$C = P(o_t)$$

- **Матрице иницијалне дистрибуције скривених стања (D):**

  Даје почетну вјероватноћу да се систем налази у неком стању у тренутку $t = 0$:

$$D = P(s_0)$$

2 Инференција скривених стања

У оквиру активне инференције, један од кључних задатака агента је да на основу примљеног опажања процијени вјероватноћу тренутног скривеног стања свијета. Овај процес се назива инференција скривених стања и представља обраду информација ради одређивања највероватнијег или најадекватнијег унутрашњег представљања окружења.

2.1 Математичка основа infer_states функције

Циљ агента је да на основу опажања o_t изврши инференцију — односно да процијени вјероватноћу да се у тренутку t налази у сваком од могућих скривених стања:

$$Q(s_t) \approx P(s_t \mid o_t)$$

гдје је $Q(s_t)$ апроксимација апостериорне вјероватноће коју агент интерно одржава. То је вјероватноћа да је систем у одређеном стању s_t, дат опажањем o_t.

Коришћењем Бајесове теореме, апостериорна вјероватноћа се може изразити као:

$$P(s_t \mid o_t) = \frac{P(o_t \mid s_t) \cdot P(s_t)}{P(o_t)}$$

Овдје су:

- $P(o_t \mid s_t)$: вјероватноћа да ће се опажање o_t јавити ако се систем налази у стању s_t (долази из генеративног модела, односно матрице A),
- $P(s_t)$: априорна вјероватноћа стања s_t, тј. интерна претходна претпоставка агента о томе у каквом се стању налази свет,
- $P(o_t)$: маргинализована вјероватноћа опажања o_t, добијена сумирањем преко свих могућих скривених стања:

$$P(o_t) = \sum_{s_t} P(o_t \mid s_t) \cdot P(s_t)$$

У практичној примјени, израз $P(o_t)$ се често не рачуна експлицитно, јер не зависи од конкретног скривеног стања s_t. Будући да је једнак за све вриједности s_t, он не утиче на релативне вјероватноће које упоређујемо. Због тога је довољно израчунати израз који је пропорционалан апостериору:

$$Q(s_t) \propto P(o_t \mid s_t) \cdot P(s_t)$$

Међутим, директно множење вјероватноћа може довести до нумеричке нестабилности, нарочито када се ради са веома малим бројевима. Да би се то избјегло, рачунање се врши у логаритамском простору, гдје се производ претвара у збир:

$$\log Q(s_t) \propto \log P(o_t \mid s_t) + \log P(s_t)$$

Након тога, примјењује се softmax функција на резултат, тако да добијемо исправну расподјелу вјероватноће:

$$\boxed{Q(s_t) = \sigma(\log P(o_t \mid s_t) + \log P(s_t))}$$

2.2 Имплементација infer_states функције

Након разматрања математичке основе, сада ћемо показати како овај процес изгледа у Python коду, и дати објашњење кода линију по линију.

```python
def infer_states(observation_index, A, prior):

  log_likelihood = log_stable(A[observation_index,:])

  log_prior = log_stable(prior)

  qs = softmax(log_likelihood + log_prior)


  return qs
```

Код 2.1: Функција infer_states

- **Улазни параметри**
  - observation_index означава која је вриједност опажања забиљежена у времену t, односно који је индекс у скупу опажања.
  - A је матрица која садржи вјероватноће $P(o_t \mid s_t)$ за све могуће парове опажања и стања. Ред у матрици одговара опажању, а колона стању.

- prior je вектор априорних вјероватноћа $P(s_t)$ који представља тренутну процјену вјероватноће сваког стања прије узимања у обзир новог опажања.

- **log_likelihood = log_stable(A[observation_index, :])**

  За свако конкретно опажање узимамо одговарајући ред из матрице A, који представља вјероватноће да ће то опажање настати у сваком од могућих скривених стања. Рачунамо њихове логаритме користећи нумерички стабилну логаритамску функцију. Ово представља $\log P(o_t \mid s_t)$.

- **log_prior = log_stable(prior)**

  Логаритам априорне вјероватноће, $\log P(s_t)$.

- **qs = softmax(log_likelihood + log_prior)**

  Сабирамо логаритме, а затим примјеном softmax функције нормализујемо резултат тако да добијемо коректну дистрибуцију вероватноћа $Q(s_t)$.

- **return qs**

  Враћамо вектор апостериорних вјероватноћа скривених стања.

3 Рачунање очекиване слободне енергије

У Active Inference-у, агент не само да процјењује скривена стања у којим се налази, већ и бира акције које ће извршити у будућности како би минимизовао изненађење. За ту сврху, агенти процењују очекивану слободну енергију (енгл. expected free energy), која мјери колико је одређени низ акција у складу са циљевима агента, његовим преференцијама и будућим очекивањима.

3.1 Математичка основа функције calculate_G_policies

Како је у Active Inference процесу циљ агента да избјегава неочекивана опажања, то је еквивалентно минимизовању вриједности $-\log P(o_t) = -\log \sum_{s_t} P(s_t) \cdot P(o_t \mid s_t)$.

Уведимо сада вјероватносну расподјелу Q над скупом скривених стања $\mathcal{S}$ (касније ће бити јасно шта она представља). Иимамо да је:

$$-\log \sum_{s_t} P(s_t) \cdot P(o_t \mid s_t) = -\log \sum_{s_t} Q(s_t) \cdot \frac{P(s_t) \cdot P(o_t \mid s_t)}{Q(s_t)}$$

Суму са десне стране можемо посматрати као очекивану вриједност расподјеле која са вјероватноћом $Q(s_t)$ узима вриједност $\dfrac{P(s_t) \cdot P(o_t \mid s_t)}{Q(s_t)}$. Како је функција $-\log$ конвексна, то можемо примијенити Јенсенову неједнакост у вјероватносном простору, чиме добијамо да је:

$$-\log \sum_{s_t} Q(s_t) \cdot \frac{P(s_t) \cdot P(o_t \mid s_t)}{Q(s_t)} \leq -\sum_{s_t} Q(s_t) \log \frac{P(s_t) \cdot P(o_t \mid s_t)}{Q(s_t)}$$

Користећи особине логаритма, имамо:

$$-\sum_{s_t} Q(s_t) \log \frac{P(s_t) \cdot P(o_t \mid s_t)}{Q(s_t)} = \sum_{s_t} Q(s_t) \log \frac{Q(s_t)}{P(s_t) \cdot P(o_t \mid s_t)}$$

Израз са десне стране једнакости називамо варијационом слободном енергијом, а расподјелу Q варијационом расподјелом. Користећи неке једноставне идентитете, можемо је представити на следећи начин, који нам даје значајан теоријски увид:

$$\begin{aligned}
&\sum_{s_t} Q(s_t) \log \frac{Q(s_t)}{P(s_t) \cdot P(o_t \mid s_t)} \\
&= \sum_{s_t} Q(s_t) \log \frac{Q(s_t)}{P(o_t) \cdot P(s_t \mid o_t)} \\
&= \sum_{s_t} Q(s_t) \log \frac{Q(s_t)}{P(s_t \mid o_t)} - \sum_{s_t} Q(s_t) \log P(o_t) \\
&= D_{KL}(Q(s_t) \mid\mid P(s_t \mid o_t)) - \log P(o_t) \qquad (1)
\end{aligned}$$

Из ове једнакости закључујемо да, ако је расподјела $Q(s_t)$ једнака расподјели $P(s_t \mid o_t)$, тада је њихова Кулбак-Лајблерова дивергенција једнака 0 и слободна енергија је тачно једнака изненађењу $-\log P(o_t)$. Дакле, пошто желимо да минимизујемо слободну енергију, то расподјела $Q(s_t)$ треба тежити расподјели $P(s_t \mid o_t)$, односно важи $Q(s_t) \approx P(s_t \mid o_t)$. Такође, примијетимо да слободну енергију можемо додатно смањити кориговањем генеративног модел тако да повећавамо вриједности $P(o_t)$, јер тада $-\log P(o_t)$ постаје мањи.

Ово је било разматрање случаја када агент само пасивно опсервира околину. Позабавимо се сада ситуацијом у којима агент предузима неке акције. Означимо:

- π — политика, тј. низ будућих акција
- $G(\pi)$ — очекивана слободна енергија политике π

Циљ агента је да изабере политику π^* која минимизује $G(\pi)$:

$$\pi^* = \arg\min_{\pi} G(\pi)$$

То ради тако што, за сваки временски тренутак t сваке политике рачуна слободну енергију, и бира ону политику која има најмању укупну слободну енергију. У недостатку савршеног знања о будућим опсервацијама, агент мора да размотри читав спектар могућих опажања. Коначна одлука се стога доноси на основу очекиване вриједности — односно просјека — свих тих потенцијалних исхода, гдје се сваком придаје значај сразмеран његовој вјероватноћи.

$$\begin{aligned} G(\pi) &= \sum_{s_t} Q(s_t \mid \pi) \log \frac{Q(s_t \mid \pi)}{P(o_t) \cdot P(s_t \mid o_t, \pi)} \\ &= \sum_{s_t} Q(s_t \mid \pi) \sum_{o_t} P(o_t \mid s_t) \log \frac{Q(s_t \mid \pi)}{P(o_t) \cdot P(s_t \mid o_t, \pi)} \end{aligned}$$

Како смо у (1) закључили да се повећањем $P(o_t)$ смањује слободна енергија, а имајући у виду да агент активно бира акције како би утицао на окружење и тиме индиректно одређује шта ће вјероватно опазити, у контексту очекиване слободне енергије ту вјероватноћу тумачимо као агентове априорне преференце – расподјелу која кодира колико агент жели да види неку опсервацију. Јасно је да та вјероватноћа не зависи од политике, па услов π изостављамо.

Као што је раније наведено, варијациона апроксимација $Q(s_t)$ се користи да би се приближила стварна расподјела $P(s_t \mid o_t)$ након што се добије неко опажање. Међутим, приликом планирања унапријед — односно прије него што је агент нешто опазио — не постоје опажања који би могли да утичу на расподјелу скривених стања. Због тога се у том случају апроксимација $Q(s_t \mid \pi)$ поставља једнако генеративном априору: $Q(s_t \mid \pi) = P(s_t \mid \pi)$. То одражава вјеровања агента о томе која су стања могућа под датом политиком, прије него што добије било какво опажање.

Како је расподјела $P(o_t \mid s_t)$ дефинисана у самом генеративном моделу, претпоставља се да апроксимација $Q(o_t \mid s_t)$ тачно одговара стварној расподјели. Комбинујући ову једнакост са претходном, добијамо:

$$Q(o_t \mid \pi) = \sum_{s_t} Q(o_t \mid s_t, \pi)\, Q(s_t \mid \pi) = \sum_{s_t} P(o_t \mid s_t, \pi)\, P(s_t \mid \pi) = P(o_t \mid \pi)$$

Користећи добијене једнакости, очекивану слободну енергију можемо рачунати као:

$$\begin{aligned}
G(\pi) &= \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log \frac{Q(s_t \mid \pi)}{P(s_t \mid o_t, \pi)} - \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log P(o_t) \\
&= \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log \frac{Q(s_t \mid \pi)\, P(o_t \mid \pi)}{P(o_t \mid s_t, \pi)\, P(s_t \mid \pi)} - \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log P(o_t) \\
&= \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log \frac{Q(s_t \mid \pi)\, Q(o_t \mid \pi)}{P(o_t \mid s_t)\, Q(s_t \mid \pi)} - \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log P(o_t) \\
&= \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log \frac{Q(o_t \mid \pi)}{P(o_t \mid s_t)} - \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log P(o_t)
\end{aligned}$$

Сада можемо да спојимо логаритме и да их раставимо у другачијем облику, чиме добијамо коначну формулу за рачунање очекиване слободне енергије:

$$\begin{aligned}
G(\pi) &= \sum_{o_t, s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \log \frac{Q(o_t \mid \pi)}{P(o_t) P(o_t \mid s_t)} \\
&= \sum_{o_t} \left[\log \frac{Q(o_t \mid \pi)}{P(o_t)} \sum_{s_t} P(o_t \mid s_t)\, Q(s_t \mid \pi) \right] - \sum_{s_t} Q(s_t \mid \pi) \sum_{o_t} P(o_t \mid s_t) \log P(o_t \mid s_t) \\
&= \sum_{o_t} Q(o_t \mid \pi) \log \frac{Q(o_t \mid \pi)}{P(o_t)} - \sum_{s_t} Q(s_t \mid \pi) \sum_{o_t} P(o_t \mid s_t) \log P(o_t \mid s_t) \\
&= \boxed{D_{KL}\left(Q(o_t \mid \pi) \;||\; P(o_t)\right) + \sum_{s_t} Q(s_t \mid \pi)\, H\left[P(o_t \mid s_t)\right]}
\end{aligned}$$

3.2 Имплеметација функције calculate_G_policies

За почетак дајемо имплементацију неких функција које ће нам бити од користи:

```
def get_expected_states(B, qs_current, action):
  """ Compute the expected states one step into the future, given a particular action """
  qs_u = B[:,:,action].dot(qs_current)

  return qs_u

def get_expected_observations(A, qs_u):
  """ Compute the expected observations one step into the future, given a particular action """

  qo_u = A.dot(qs_u)

  return qo_u

def entropy(A):
  """ Compute the entropy of a set of conditional distributions, i.e. one entropy value per column """

  H_A = - (A * log_stable(A)).sum(axis=0)

  return H_A

def kl_divergence(qo_u, C):
  """ Compute the Kullback-Leibler divergence between two 1-D categorical distributions"""

  return (log_stable(qo_u) - log_stable(C)).dot(qo_u)
```

Код 3.1: Помоћне функције за рачунање очекиване слободне енергије

Сада можемо да пређемо на имплементацију главне функције calculate_G_policies, као и на објашњење њеног рада:

```
def calculate_G_policies(A, B, C, qs_current, policies):

  G = np.zeros(len(policies))
  H_A = entropy(A)
  for policy_id, policy in enumerate(policies):

    t_horizon = policy.shape[0]
```

```
    G_pi = 0.0

    for t in range(t_horizon):

      action = policy[t,0]

      if t == 0:
        qs_prev = qs_current
      else:
        qs_prev = qs_pi_t

      qs_pi_t = get_expected_states(B, qs_prev, action)
      qo_pi_t = get_expected_observations(A, qs_pi_t)

      kld = kl_divergence(qo_pi_t, C)

      G_pi_t = H_A.dot(qs_pi_t) + kld

      G_pi += G_pi_t

    G[policy_id] += G_pi

  return G
```

Код 3.2: Функција calculate_G_policies

- **Улазни параметри**
  - A је матрица зависности опажања од скривених стања, односно $P(o_t \mid s_t)$
  - B је транзициона матрица $P(s_t \mid s_{t-1}, u_{t-1})$
  - C матрица преференци агента, $P(o_t)$
  - qs_current вектор тренутне апроксимативне расподјеле $Q(s_t)$
  - policies низ политика
- **G = np.zeros(len(policies))**

  Креирамо вектор G који има онолико елемената колико има политика, иницијализован нулама. У њему ћемо складиштити очекивану слободну енергију сваке политике.
- **H_A = entropy(A)**

  Рачунамо вектор H_A гдје свака компонента представља ентропију дистрибуције опажања условљену одређеним скривеним стањем, тј:

$$H\_A[s] = -\sum_{o} A[o, s] \log A[o, s] = H\left[P(o \mid s)\right]$$

Ово представља неизвјесност у вези са опажањем коју агенат има ако претпостави да се налази у скривеном стању s. Будући да је опажајни модел исти за све политике, овај вектор рачунамо једном.

- **for policy_id, policy in enumerate(policies):**

  Пролазимо петљом кроз све политике, гдје policy_id представља индекс политике, а policy конкретну политику, односно низ акција.

- **t_horizon = policy.shape[0]**

  Представља број временских корака у политикама.

- **G_pi = 0.0**

  Иницијализује се почетна вриједност очекиване слободне енергије за тренутну политику.

- **for t in range(t_horizon):**

  Пролазимо петљом кроз сваки временски корак унутар тренутне политике.

- **action = policy[t,0]**

  Узимамо конкретну акцију коју политика предвиђа у кораку t.

- **if t == 0: qs_prev = qs_current**

  При првом кораку симулације, за процјену скривених стања узима се њихова иницијална расподјела која је обично дата матрицом D. Тај вектор представља најбоље знање које агент у том тренутку има о свијету.

- **else: qs_prev = qs_pi_t**

  Након првог корака, користи се претходно предвиђене вјероватносне расподјела скривених стања, добијена у току симулације исте политике. Другим ријечима, предвиђено стање из претходног корака користи се као основа за израчунавање наредног стања.

- **qs_pi_t = get_expected_states(B, qs_prev, action)**

  Предвиђамо вјероватноће скривених стања након извршења тренутне акције, користећи динамику система описану матрицом B и претходну процјену стања.

$$qs\_pi\_t = B[:, :, action] \cdot qs\_prev = Q(s_{t+1} \mid \pi)$$

- **qo_pi_t = get_expected_observations(A, qs_pi_t)**

  Рачунамо очекиване вјероватноће опажања на основу предвиђених вјероватноћа скривених стања и матрице A.

  $$qo\_pi\_t = A \cdot qs\_pi\_t = Q(o_{t+1} \mid \pi)$$

- **kld = kl_divergence(qo_pi_t, C)**

  Израчунавамо Кулбак-Лајблерову дивергенцију између предвиђених опажања qo_pi_t и агентових преференци C, односно:

  $$kld = \sum_o qo\_pi\_t[o] \cdot (\log qo\_pi\_t[o] - \log C[o]) = D_{KL}\left(Q(o_t \mid \pi) \mid\mid P(o_t)\right)$$

- **G_pi_t = H_A.dot(qs_pi_t) + kld**

  Рачуна се очекивана слободна енергија у кораку t дате политике по изведеној формули.

  $$\begin{aligned} H\_A \cdot qs\_pi\_t + kld &= \sum_s H\_A[s]\, qs\_pi\_t[s] \; + \; kld \\ &= \sum_{s_t} Q(s_t \mid \pi)\, H\left[P(o_t \mid s_t)\right] + D_{KL}\left(Q(o_t \mid \pi) \mid\mid P(o_t)\right) \end{aligned}$$

- **G_pi += G_pi_t**

  Додајемо допринос очекиване слободне енергије за тренутни временски корак t у укупну очекивану слободну енергију тренутне политике.

- **G[policy_id] += G_pi**

  Након што обрадимо све временске кораке, вриједност укупне очекиване слободне енергије за тренутну политику смештамо у вектор G.

- **return G**

  Функција враћа вектор G који садржи очекивану слободну енергију за све процијењене политику.