Преддипломен проект

Решаване на оптимизационната задача FrozenLake с невробиологичен симулатор по метода поощрение/наказание

Факултет по Математика и Информатика

Студент: Борислав Стоянов Марков

Факултетен номер: 0MI3400048

Учебен план: Изкуствен Интелект

Научен ръководител: Проф. Петя Копринкова-Христова, Институт по Информационни и Комуникационни Технологии (ИИКТ), БАН

София: 03.2023г.

1. Съдържание

2. Увод

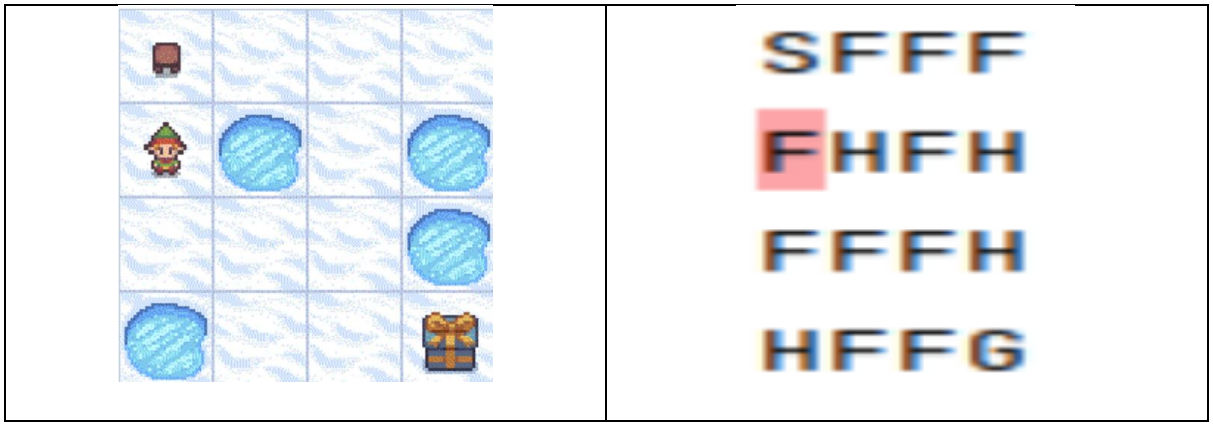
Невробиологията все повече набира скорост в света на изкуствения интелект. Има все повече изследвания на функционирането на нервни клетки, които са довели до създаването на биологично обоснованите spike timing модели на невроните, както и много знания за структурната организация и функционирането на мозъка на бозайниците при вземане на решения. Доказано е, че много от решенията се вземат по метода на поощрението и наказанието (Reinforcement Learning).

Целта на преддипломния проект е да се разработи модел на биологично обоснована (spike timing) невронна мрежа посредством библиотеката NEST Simulator, която е в състояние да решава оптимизационната задача за преминаване на агент през известната среда FrozenLake от пакета Gym посредством reinforcement обучение. Задачата е с дискретни състояния на средата и 4 възможни действия на агента.

Проектът включва кратък обзор в областта на Spike Timing Neural Networks, описание на теоретичната постановка, код на Python с използване на библиотеката NEST Simulator и анализ на резултатите. В проекта се изпробват различни параметри на биологично подобните неврони и решението е илюстрирано с подходящи визуализации и графики съпътстващи обучителния процес.

3. Средата „FrozenLake“[4] в Gym

Средата Frozen Lake е част от подпакета за текстови игри. Това са много опростени игри с визуализация като текст. Имат малък на брой дискретни състояния и контрола на агента е също дискретен с малък брой действия. Този пакет е разработен за обучителни цели по метода на поощрението и наказанието (Reinforcement Learning). Целта е да се сравняват различни решения при една и съща среда. На следващата фигура е показана примерна визуализация на тази среда. Имаме езеро, правоъгълна фигура, разграфена на квадранти, например 4x4 с кодове латинските букви S,F,H,G.



Фигура 3.1 – Примерна визуализация на FrozenLake вляво и текстов еквивалент вдясно.

Значенията на квадрантите са старт-S, заледен-F, дупка-H, цел-G. Човече тръгва от позиция „S” и трябва да стигне до крайната цел „G”, преминавайки по заледени участъци „F“. Някои от квадрантите имат дупки “H” и там текущият опит приключва неуспешно. Поощрението и наказанието е както следва:

- При достигане на крайната цел „G” награда от 1 точка и експериментът приключва.
- При попадане в дупка “H” награда „0“ и експериментът приключва.
- При попадане на заледен участък „F“ награда „0“.

Има една особеност на средата, че на заледените участъци агентът може да се подхлъзне и да не отиде в избраната посока, тоест имаме хлъзгане. Хлъзгането зависи с какви настройки е пусната средата: is_slippery=False|True. Основна информация за средата са дадени в следващата таблица.

Пространство на действията	`Discrete(4)`
Вектор на наблюдението	`Discrete(16)`
Импортиране в Питон	`gym.make("FrozenLake-v0")`

Таблица 3.1. Общи параметри за средата

Агентът може да се придвижва в четири посоки, всяка с код от 0 до 3 включително, а именно: наляво-0, надолу-1, надясно-2, нагоре-3.

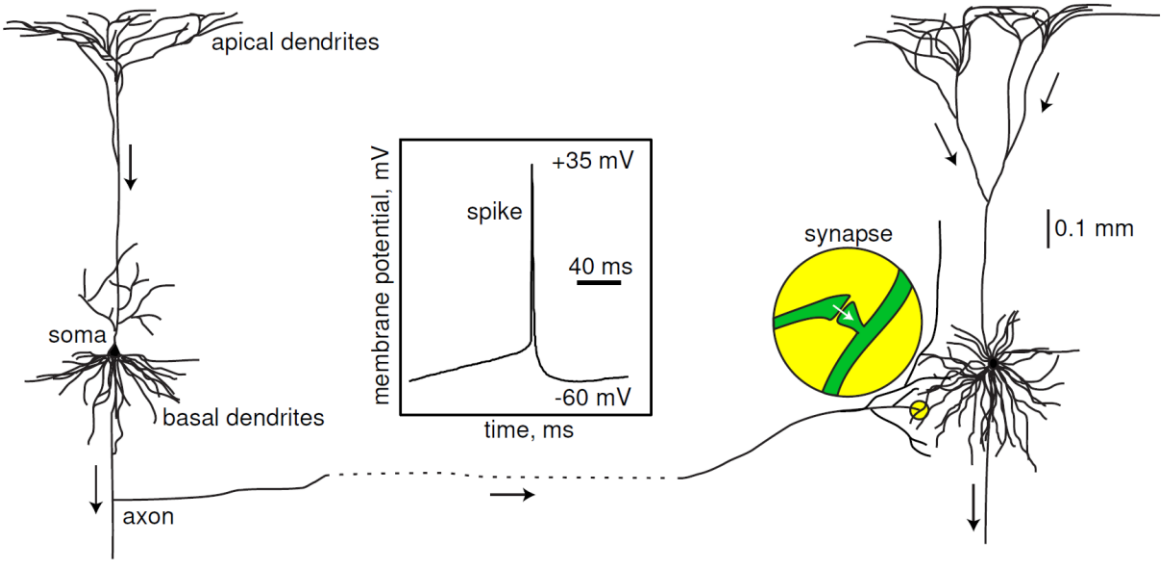
4. Въведение в невробиологичните симулации

Невробиологията е дисциплина, която може да засяга един или друг аспект свързан с работата на нервната система при живите организми. За нашите цели се фокусираме повече върху математическия апарат, отколкото върху биологичната основа. Най-забележимият аспект на нервната система при човека и при други живи организми е начинът на вземане на решения и способността да се обучават с поощрение и наказание. Връзката между реинфорсмънт обучението (Reinforcement learning) и новробиологичните науки се крие в химично имплементираната награда – допаминът ([1] глава 15). Допаминът пренася темпоралната грешка (TD error) до структурите в мозъка, които са отговорни за вземане на решение.

4.1 Основи на невробиологията

Невроните са основните компоненти на нервната система. Това са клетки специализирани в пренасяне и обработка на информация посредством електрохимически и химически сигнали [1]. Невроните се състоят от тяло, дендрити и един аксон. Дендритите са разклонения от тялото, чрез които клетката се свързва с аксони от други неврони или са сензори, в случай на сензорни неврони.

Невронът събира импулси от много входове и когато сумарно тези входове преминат някаква граница, невронът изстрелва потенциал (action potential) или тъй наречения импулс (spike). Това е и фундаменталният начин на комуникация между невроните ([2], Ch.2). Изходният сигнал на неврона са електрически импулси, пътуващи по аксона, наречени спайкове (spikes).



Фиг.4.1.1. Рисунка на два свързани неврони и ин-витро записан спайк

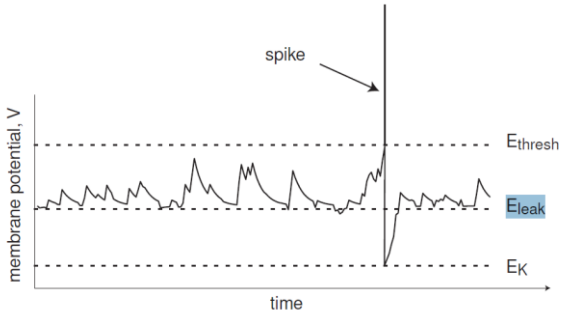
Адаптирано от [3], Глава 1.1

4.2 Математически апарат за моделиране на невроните

Към момента невроните се моделират по много различни начини и има описани десетки видове диференциални уравнения на различни неврони. Един от първите математически формализми на неврони е на Ходжкин и Хъксли, описан през петдесетте години на миналия век. Уравнението се оказва доста сложно за решаване на практически задачи и затова по-късно са предложени опростени модели. Integrate-and-fire са фамилия модели от които най-популярният към момента е leaky integrate-and-fire неврон. Това е идеализация на неврон с утечки (по закона на Ом) , който е суматор на токове.

$$C\dot{V} = I - g_{leak}(V - E_{leak}) \qquad (4.2.1)$$

Тук формализма е взет от [3], C е капацитетът на мембраната на неврона, V е мембранният потенциал, g_{leak} е проводимостта на клетъчната мембрана, E_{leak} е равновесният потенциал на мембраната. На следващата фигура се онагледява действието на неврона.



Фиг. 4.2.1 Потенциална диаграма на суматорен неврон с утечка (Leaky integrate-and-fire neuron). Адаптация от [3].

На фигурата се вижда, че при напрежение надвишаващо E_{thresh} се произвежда токов импулс (spike) и напрежението се връща до стойност E_k. При липса на входни токове или шум се забелязва как напрежението намалява експоненциално, стремейки се към E_{leak}.

4.3 Невробиологичен симулатор NEST

NEST е симулатор за невронни мрежи основани на спайкове (spiking neural network models или SNN) и може да послужи за: модели за обработка на информация, модели на мрежова динамика, модели на обучение и синаптична пластичност [5]. Симулаторът представлява библиотека на Python и може да се вгради в по-голямо приложение. Структурата на програмата е например:

1. Създаване на невронните групи и други устройства (напр. генератори на шум или входен ток)
2. Свързване на групите неврони в желаната структура със съответните тегла на връзките
3. Поставяне на виртуални измервателни уреди и свързване с интересуващите ни групи неврони (напр. волтметри, детектори на импулси);
4. Симулиране с метода nest.Simulate(t), който проиграва една евентуална симулация за време t милисекунди.

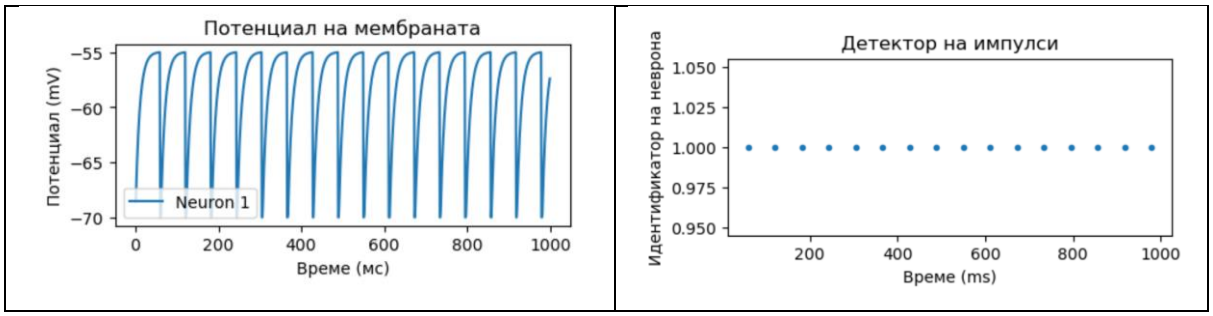
Таблица 4.3.1 Структура на програма използваща NEST simulator

Можем да разгледаме примерен код на програма, състояща се от един неврон, на който подаваме постоянен ток от 376 [pA] и замерваме напрежението на мембраната, както и генерираните импулси.

```
import nest
import matplotlib.pyplot as plt
import nest
nest.ResetKernel()
neuron = nest.Create("iaf_psc_alpha", {"I_e":376.0})
voltmeter = nest.Create("voltmeter")
spikerecorder = nest.Create("spike_recorder")
nest.Connect(voltmeter, neuron)
nest.Connect(neuron, spikerecorder)
nest.Simulate(1000.0)
plt.rcParams["figure.figsize"] = (5,2)
nest.voltage_trace.from_device(voltmeter)
plt.show()
nest.raster_plot.from_device(spikerecorder, hist=False,
title="spikerecorder")
plt.show()
```

Таблица 4.3.2 Примерна програма използваща NEST simulator с един неврон и симулация от 1000 милисекунди

Изходът от програмата са две графики изобразяващи скоковете в напрежението и генерираните импулси.

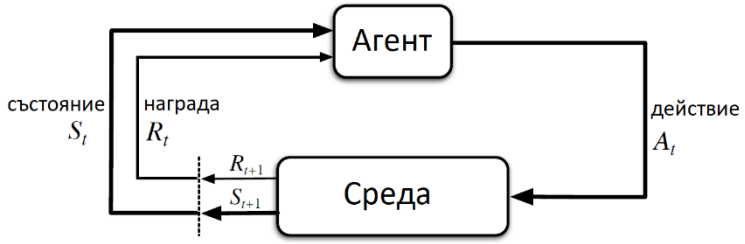


Фигура 4.3.1 Напрежение на мембраната и импулси от неврона

При по-голям ток, импулсите ще следват по-бързо, при по-малък ток на неврона импулсите ще намалеят или ще изчезнат.

5. Подход за решаване на задачата

FrozenLake като задача за Reinforcement Learning е формулирана като агент и среда, взаимодействащи си чрез марковски процес. След всяко действие на агентът A_t средата ни отговаря с ново състояние S_{t+1} и се дава съответната награда R_{t+1}.



Фиг. 5.1. Взаимодействие Агент-Среда при марковски процес на решението, адаптирано от [1]Глава 3.1

Динамиката на марковският процес на решенията (MDP) е определен от следната функция, даваща връзка между състоянията, наградите и действията:

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \quad (5.1)$$

Функцията „p“ е вероятност при условие, че средата се намира в състояние „s“ и агентът е предприел действие „a“ то средата да премине в ново състояние „s‘” и да има награда ”r”. Знакът за равенство с точка означава „по дефиниция“. Състоянието в случая отговаря на клетка от таблица на всеки квадрант от FrozenLake, а действието е посоката на следващата стъпка на агента. Така (5.1) определя какъв би бил резултатът за избраната посока на движение на агента.

За някои приложения е подходящо да се направи извадков модел от опита, за други задачи е подходящо да се направи вероятностен модел. От многото алгоритми за решаване на подобна задача в случая е подходящо да се построи вероятностния модел, тъй като имаме изброими състояния на средата и можем да намерим функцията „p” от (5.1) и ще можем въз основа на нея да дефинираме политика на действията “π”. В случая подходящ алгоритъм е Q-learning от [1]. В началото на пускане на агента не знаем решението и агента трябва да стигне по случаен начин до решението. От това следва, че имаме случаен елемент или т.нар “exploration”, т.е. агента ще действа само на базата на случайности. След научаване на правилното решение ще можем да кажем от къде е минал агента и ще можем да следваме модела или ще имаме т.нар. “exploitation”.

5.1 Таблични методи

Обучението с поощрение и наказание в случай на дискретни и достатъчно малко на брой възможни състояния и действия може да бъде извършено със записване на

вероятностите (5.1) за всички възможни комбинации в масиви или таблици.За да намерим оптималната стратегия π* (политика), водеща до максимално поощрение (минимално наказание), ние оптимизираме стойността на състоянието s при зададена стратегия π v(s) като:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|\ S_t = s]\ = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \forall s \in \mathcal{S} \qquad (5.1.1)$$

или стойността на двойката състояние s, действие a q*(s,a) съответно:

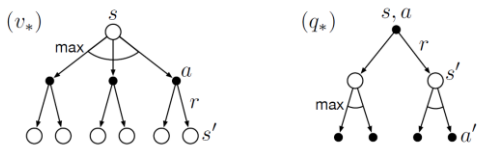
$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|\ S_t = s, A_t = a]\ = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \qquad (5.1.1)$$

Задачата се решава с метода на динамичното програмиране и уравнението на Белман за стойността на състоянието при оптималната политика π*:

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi*}(s,a) \\ &= \max_a \mathbb{E}_{\pi*}\left[G_t \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}_{\pi*}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \end{aligned} \qquad (5.1.2)$$

и за стойността на двойката състояние-действие съответно (вж. [1] глава 3.6):

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right] = \\ &= \sum_{s',r} p(s', r|s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right] \end{aligned} \qquad (5.1.3)$$

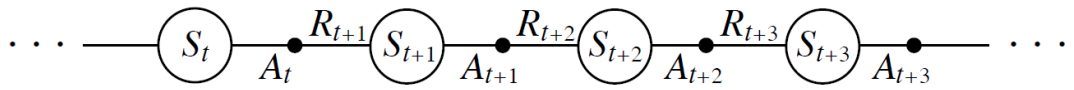


Фиг.5.1.1 Диаграма на избор на действие при използване на v (ляво) и q (дясно) функция съответно. Адаптирано от [1] глава 3.

Фигура 5.1.1. представя диаграмата за намиране на следващото оптимално действие при използването на v и q съответно.

Имайки апроксимация на “q*” избирането на оптималните действия е по-лесно отколкото ако имаме апроксимация на “v*”, защото агентът не трябва да ходи една стъпка напред. Достатъчно е само да избере действието, което максимизира q*(s,a). Оттук и извода, че с цената да пазим таблично двойки (s,a) вместо таблично да пазим само количествена мярка за състоянието (s) ще ни осигури най-добрата стратегия “π*” за агента без да знаем нищо за останалите бъдещи състояния, което дефакто означава, че не е необходимо да знаем динамиката на средата[1].

Алгоритъмът осигуряващ ни най-доброто от Монте Карло методите и динамичното програмиране ни води до алгоритъмът „Sarsa: On-policy TD Control“(вж. [1] глава 6.4).



Фиг.5.1.2 Примерен епизод, вж. [1] глава 6.4

Тук разглеждаме преходът от двойката състояние-действие към друга двойка състояние-действие и съпоставяне на двойките с някаква стойност. Апроксимацията на функцията ще правим с TD(0) (Temporal difference 0) и промяната ще се прави след всеки ход към нетерминално състояние.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \qquad (5.1.3)$$

Алгоритъмът е разписан в по-долната фигура.

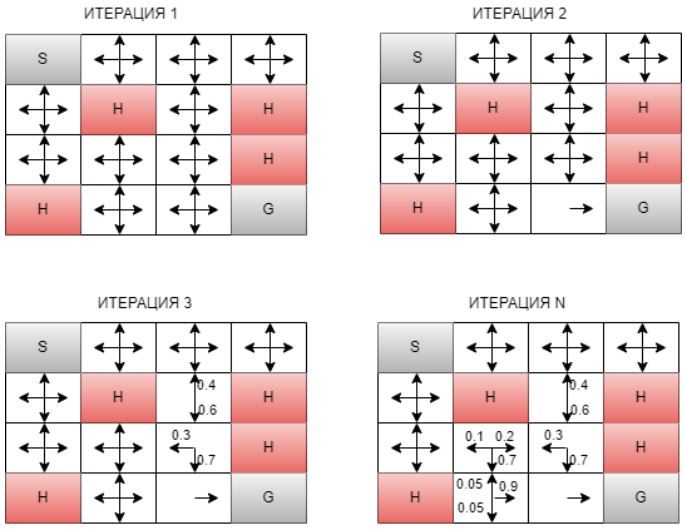
Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Фиг. 5.1.2 Алгоритъм на SARSA TD(0), вж. [1] глава 6

С цел изследване на възможно най-много състояния с използва т.нар. ε-greedy (ε-алчен) избор на следващото действие, което означава, че не винаги предприемаме действието, водещо до максимална печалба, а с някаква вероятност ε вземаме друго неоптимално действие.



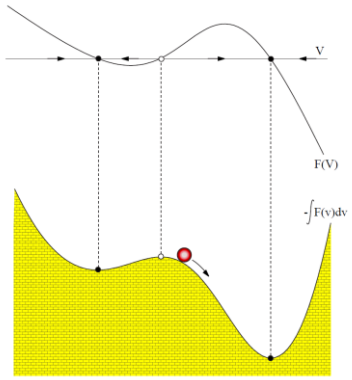
Фиг.5.1.3 Примерни итеративни стъпки за таблично решаване на задачата

На Фигура 5.1.3 са изобразени примерни итерации при решаване на задачата. В таблица, ще имаме по една клетка за всеки квадрант от полето на FrozenLake и всяко такова квадратче ще е разделено на 4. Стрелките означават действието (ляво,дясно,горе, долу),

което максимизира общата награда в края на еизода. В нашият случай FrozenLake може да се стартира като детерминирана среда без хлъзгави участъци, но може да бъде стартирана и като недетерминирана среда с хлъзгане. Недетерминирана означава, че въпреки, че сме посочили посока, като например надясно, агентът може да премине и надолу в определени случаи, което прави решението много по-трудно.

5.2 Победителят печели всичко

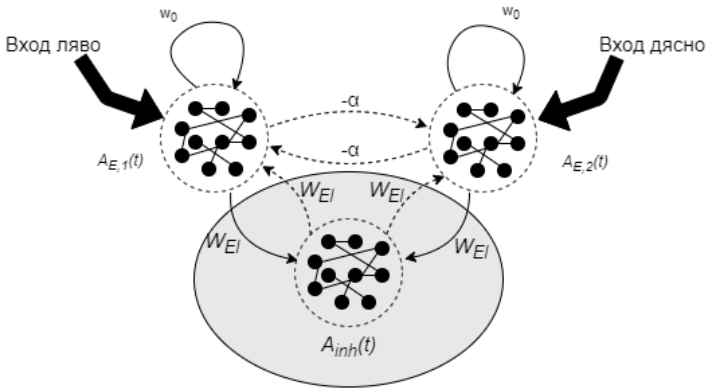
При динамичните системи изборът на различни действия при различни параметри понякога може да се окаже проблем, тъй като невронните групи навлизат в устойчиво равновесно състояние, и не могат да бъдат изместени от него. Това би се изродило в нашият случай като например агентът да отива само надясно да кажем. За тези положения при динамични системи от един неврон споменава Изикевич в [3] в глава 3.2.6. Наистина за системи с повече неврони теоретичното изчисление на практика е невъзможно и единствено симулациите са водещото, тъй като няма добре установена математическа теория за това. Ще разгледаме механичната интерпретация според [3] на устойчиво и неустойчиво равновесно състояние на следващата фигура.



Фиг. 5.2.1 Механична интерпретация на стабилен и нестабилен еквилибриум. Вж. [3].

Топката на фигурата няма маса (без инерция) и се движи към възможно най-ниската точка със скорост пропорционална на наклона. Механичната диаграма се променя по време на симулацията. Искаме да има 4 различни устойчиви положения (по едно за четирите възможни действия на агента) тръгвайки от една нестабилна точка. За целта ще използваме WTA (Winner Takes All) схема, което представлява начин на свързване на невронни групи и като цяло е известен подход в невронните мрежи. WTA има един

изход от K възможни. За WTA са необходими възбуждане и потискане на невроните. Всеки изход е представен от група неврони с положителна връзка (excitatory neurons). Входният сигнал възбужда невроните от всяка група, но всеки от тях потиска всички останали и така се състезава с тях. След време само една от невронните групи се оказва най-силна.

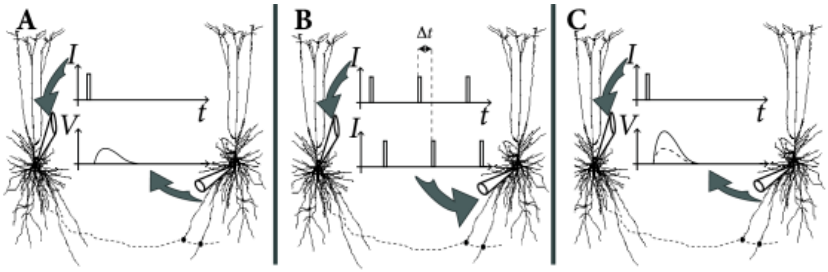


Фиг. 5.2.2 Примерен WTA с два възможни избора и ефективно потискане. Две популации с възбудими неврони взаимодействат с обща група от потискащи неврони. Адаптирано от [6] глава 16.3.

На фигура 5.2.2 са показани как са свързани две невронни групи. Всяка от групите $A_{E,1}$ $A_{E,2}$ действа усилващо на обща група A_{inh}, която пък от своя страна действа потискащо на $A_{E,1}$ и $A_{E,2}$. Групите A_E действат самоусилващо с някакво тегло w_0. Изборът на действие при WTA става като след някакви милисекунди на симулация, достатъчна да се възбудят невроните, преброим спайковете генерирани във всяка от групите A_E . Избираме действието отговарящо на групата с най-много спайкове.

5.3 Обучение с импулсно-времево зависима пластичност(STDP)

Като цяло се приема, че обучението при бозайниците има механизъм на промяна на синаптичната ефективност, както е и при по-простите организми. Исторически теоретичните обосновки се базират на постулата на Хеб (Hebb, 1949), че последователното активиране на два свързани неврона усилва връзката помежду им. Тази промяна наричаме Хебианова пластичност и тя зависи от от пресинаптична активност („pre-synaptic“) и постсинаптична активност („post-synaptic“).



*Фиг.5.3.1. Импулсно-времево зависима пластичност (Spike Timing Dependent Plasticity - STDP) **A**. Вътреклетъчни електроди измерват активността на два свързани неврона(аксоните са пунктираната линия). Подава се тестов токов импулс (I) на пресинаптичния неврон и се отчита активност в постсинаптичния неврон (V) . **B.** По време на активиран протокол за синаптична пластичност на двата неврона са подадени токови импулси в точно определено време. **C.** След извеждане на невроните от протокола на синаптична зависимост отново подаваме тестов токов импулс (I) на пресинаптичния неврон и се отчита повишена активност в постсинаптичния неврон (V) (с пунктир е отчетената активност преди сдвояването, плътната линия е след сдвояването. Фигурата е взета от [6], фиг.19.4.*

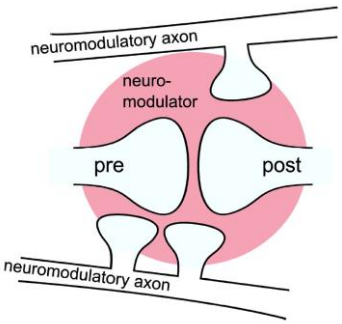
На фигура 5.3.1 са дадения разяснения за STDP като механизъм. Да означим времето на пресинаптичния импулс като t_{pre} и времето на постсинаптичния импулс като t_{post}. Промяната на теглото на пластичния синапс е зависима от времето $|\Delta t| = |t_{post} - t_{pre}|$. В най-опростен вид промяната на теглата се дава с формулите (вж.[6] глава 19.2.2):

$$\Delta w_+ = A_+(w).e^{-\frac{|\Delta t|}{\tau_+}} \quad \textit{във време} \quad t_{post} \quad \textit{при} \quad t_{pre} < t_{post} \tag{5.3.1}$$

$$\Delta w_- = A_-(w).e^{-\frac{|\Delta t|}{\tau_-}} \quad \textit{във време} \quad t_{pre} \quad \textit{при} \quad t_{pre} > t_{post} \tag{5.3.2}$$

Тук $A_+(w)$ и $A_-(w)$ представляват силата на промяната на теглата, τ_+ и τ_- са константи а w е теглото на синаптичната връзка. Сигнали, които имат далечни по време импулси, допринасят много малко към обучението заради експоненциално-намаляващата зависимост, дадена с интервала $|\Delta t|$.

Активирането на пластична активност може да бъде с външен невромодулатор, който се излива извън клетките. Има такъв биологичен механизъм в бозаиниците и други животни. „Изливането“ на невромодулатор около синапсите се нарича обемно подаване (volume transmission). Самото активиране на протокола за обучение е обяснено нагледно на следващата фиг.5.3.2.



Фиг.5.3.2 Обемно подаване на невротрансмитер. „neuromodulatory axon“ – невромодулаторни аксони, „pre“ – аксон от предхождащ неврон, „post” – аксон от последващ неврон. Областта в розово представлява областта на модулираните синапси, за простота е даден само един синапс. Фигурата е взета от [7].

Така обучението на пластичните синапси се контролира посредством друга група от невромодулиращи неврони. Подобен механизъм за активиране на протокола STDP е заложен в Симулатора NEST и се нарича обемен трансмитер (volume_transmitter). С този механизъм обучението е не само на база на пресинаптичните и постсинаптичните неврони, но и на трети невромодулаторен сигнал. Моделът в симулатора NEST, който представя обучаемите синапси с механизъм STDP и поддържа обемен трансмитер, като трети сигнал се нарича „**stdp_dopamine_synapse**“. Параметрите по подразбиране на 'stdp_dopamine_synapse' могат да бъдат отпечатани с кода: nest.GetDefaults('stdp_dopamine_synapse').

```
{'A_minus': 1.5,
 'A_plus': 1.0,
 'b': 0.0,
 'c': 0.0,
 'delay': 1.0,
 'has_delay': True,
 'n': 0.0,
 'num_connections': 0,
 'receptor_type': 0,
```

```
 'requires_symmetric': False,
 'synapse_model': 'stdp_dopamine_synapse',
 'synapse_modelid': 30,
 'tau_c': 1000.0,
 'tau_n': 200.0,
 'tau_plus': 20.0,
 'vt': -1,
 'Wmax': 200.0,
 'Wmin': 0.0,
 'weight': 1.0,
 'weight_recorder': ()}
```

Таблица 5.3.1 Параметри по подразбиране за модела 'stdp_dopamine_synapse'

Значенията на параметрите могат да бъдат намерени от [5] на страницата за модели. Ще изброим по-важните, които ни засягат.

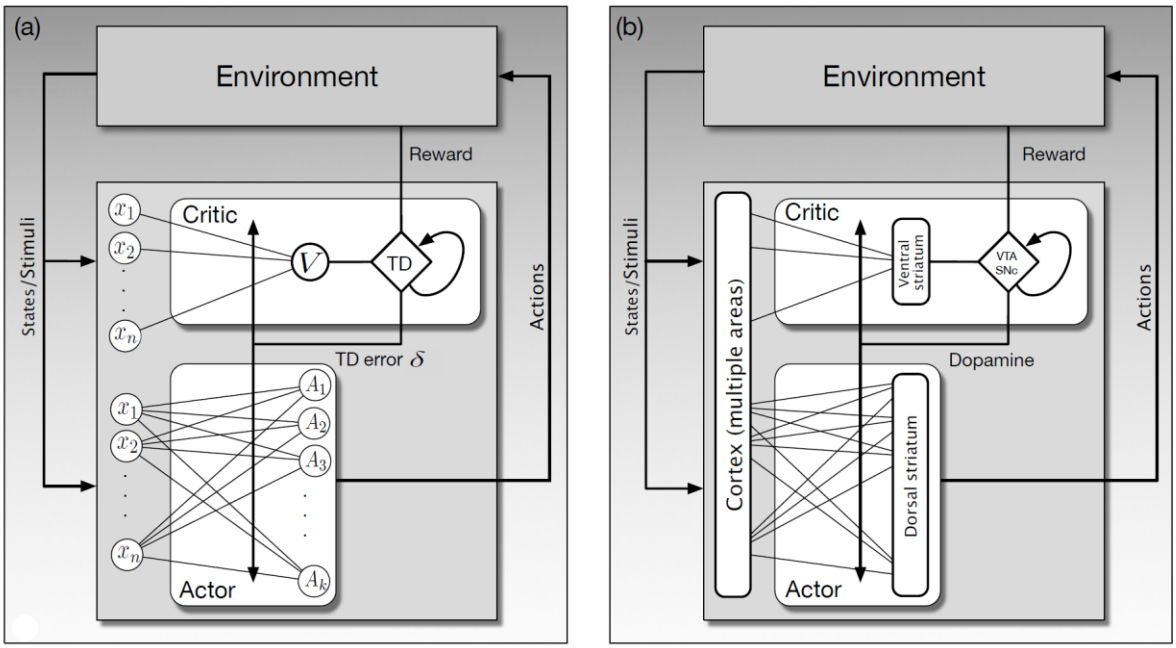
Име на параметъра	Значение
A_plus	Коефициент на обучение, когато пресинаптичният импулс изпреварва по време постсинаптичния импулс за два свързани неврона.
A_minus	Коефициент на обучение, когато постсинаптичният импулс изпреварва пресинаптичния по време импулс за два свързани неврона.
tau_plus	STDP времева константа от уравнение (5.3.1)
tau_c	Времева константа на „следата“ на обучението (eligibility trace)
tau_n	Времева константа на следата на допамина (dopaminergic trace)
b	Допаминова базова концентрация
Wmin	Минимални тегла на обучаемите синапси
Wmax	Максимални тегла на обучаемите синапси

Таблица 5.3.2 По-важните параметри за допаминови синапси и техните описания

5.4 Постановка за решаване на задачата

Най-забележимата допирна точка на обучението по метода на поощрение-наказание и невронауката е дълбоката химическа връзка на допамина, заложена при бозайниците. Допаминът отговаря за преноса на времевата грешка (TD) до съответните структури на мозъка, къде се извършва обучение и се взема решение за по-нататъчно действие. За

методите основани на времевата грешка във време t тя е $\delta_{t-1} = R_t + \gamma V(S_t) - \gamma V(S_{t-1})$. Алгоритмите актьор-критика научават и двете, и политиката за актьора и функциите за очаквана награда. „Актьорът“ е компонент, който научава политиката на действие, а „критиката“ е компонент, който научава да „критикува“ текущо следваната политика от „актьора“. Критиката използва времевата грешка (TD) за да апроксимира функция за състояние-действие за текущата политика ($q_\pi(s,a)$). Смята се, че две структури в стриатума от мозъка на бозайниците отговарят за актор и критика, това са Dorsal striatum и Ventral striatum (вж.[1] глава 15.7).



Фиг. 5.4.1 Актьор-Критика с невронна мрежа и хипотетична невронна имплементация. а) Актьор-критика като изкуствена невронна мрежа. Актьорът променя политиката спрямо TD грешката δ, който получава от критиката. Критиката създава грешката TD от сигнала за награда Reward. Актьорът няма директен достъп до Reward сигнала. Критиката няма директен достъп до действието. b) Хипотетична невро-имплементация на актьор критика. Актьорът и компонентът научаващ функцията за стойност са съответно в вентралната и дорсалната части на стриатума. Времевата грешката (TD) δ се предава от допаминът, генериран от VTA. Фигурата е копирана от [1] фиг 15.5.

Аналогично за текущият проблем ще използваме комбинация от актьор-критика и алгоритъм, научаващ функцията q(s,a). По схемата от фиг. 5.4.1 (b) ще използваме компонентът критика за да научи функцията q(s,a), а компонентът за актьор ще съдържа готовото решение, кодирано в синапсите на връзките.

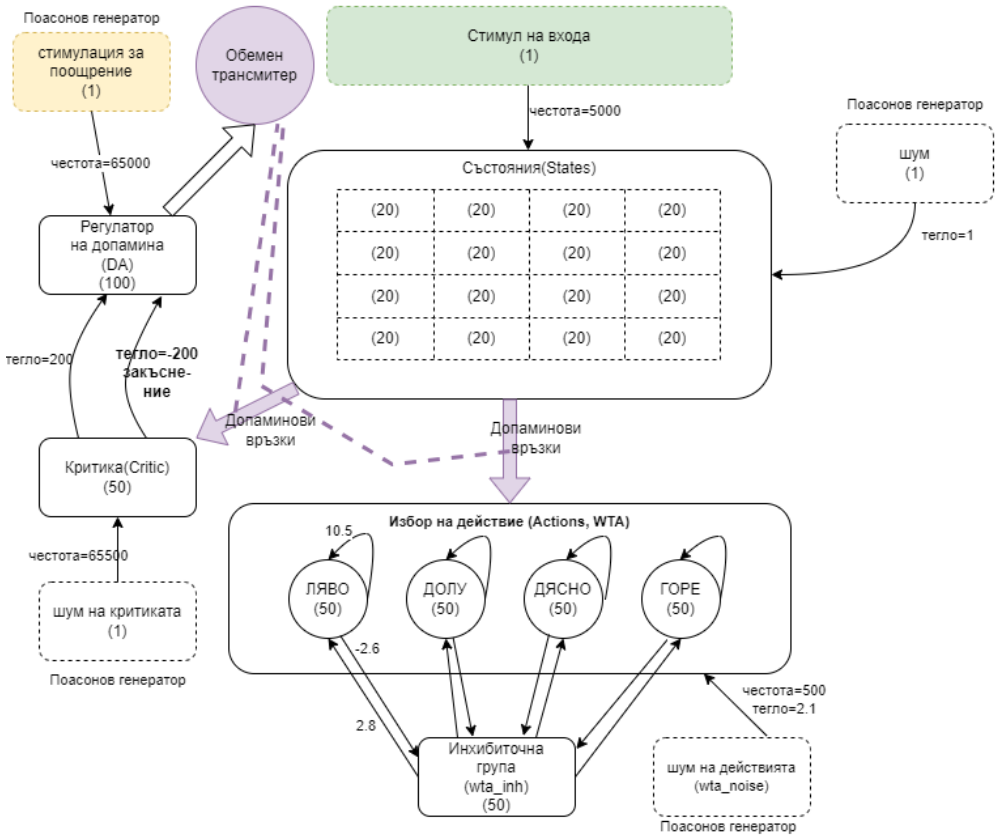
За апроксимация на q(s,a) от компонентът за критика ще използвам алгоритъмът SARSA, описан в 5.1 с известна адаптация. Тъй като активността на невроните се моделира със spike timing, се налага преминаването от спайкове към числени стойности на изхода на всяка група неврони.

За всеки квадрант от таблото на FrozenLake се създава отделна група от по 20 неврона, разположени таблично, както е показано на Фигура 5.4.2. Това са възможните състояния на агента (States). Например ако решаваме 4x4 FrozenLake, ще имаме State от 20x4x4 неврона.

При преместване на агента на различен квадрант се активира само определената група неврони отговаряща за това състояние с определени координати. Активацията става посредством генератор на поасонов шум с определена честота, наречен „стимул“.

Всяка една невронна група ще има връзка към съответната група за действие, както е показано на Фиг. 5.1.3. Начинът на свързване ще бъде разяснен по-подробно по-надолу.

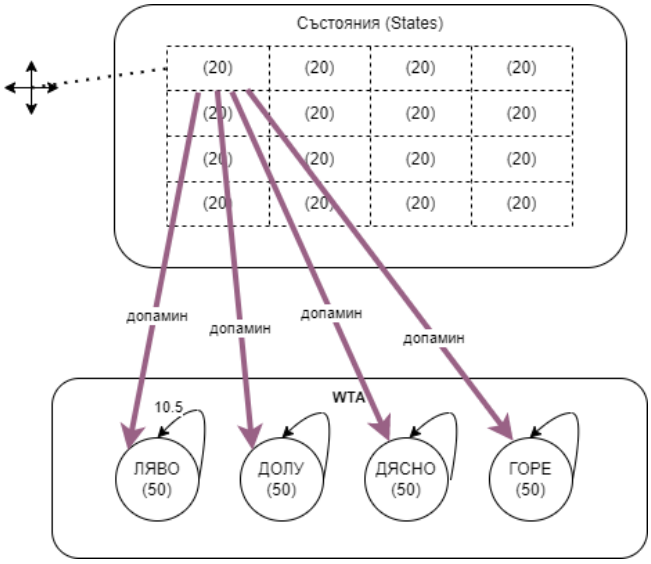
„States“ са свързани към WTA схема, описана в т.5.2 с 4 възможни състояния като свързването е „всеки с всеки“ (в NEST - „all_to_all“). Всяка група от WTA се състои от 50 неврона и отговаря съответно на действията на агента, наречени „Actions”, от 0 до 3 включително, а именно: наляво-0, надолу-1, надясно-2, нагоре-3. Връзките между „States“ и „Actions“ са с допаминови синапси, първоначално с тегла 0.0, които се обучават посредством пластични синапси (вж. 5.3) (STDP, spike-timing dependent plasticity – Markram et al., 1997; Bi and Poo, 1998, 2001). Формулите за промяна на теглата в опростен вид са (5.3.1) и (5.3.2). Диаграмата на свързване е дадена на Фиг.5.4.2.



Фиг.5.4.2 Диаграма на свързване на невронните групи

„States“ са свързани с друга невронна група, наречена „Critic“ от 50 неврона също с допаминови връзки. Тази група представя „q*“ функцията от уравнението на Белман (5.1.2). „Critic“ е свързана с друга група от 100 неврона, отговарящи за нивото на допамина, условно наречена „DA“. Наградата от средата FrozenLake ще се формира като сигнал от поасонов шумогенератор с определена честота пропорционална на наградата. Този вход е наречен „Reward Stimulus“.

На фигура 5.4.3 е показан в подробности начина на свързване на „States“ и „Actions“. В тези синаптични връзки е заложено решението, защото в процеса на обучение на „Critic“, успешните ходове на агента са предпоставка за усилваща връзка от дадено състояние в таблицата, например квадрант (0,2), към определена посока, например „Ляво“.



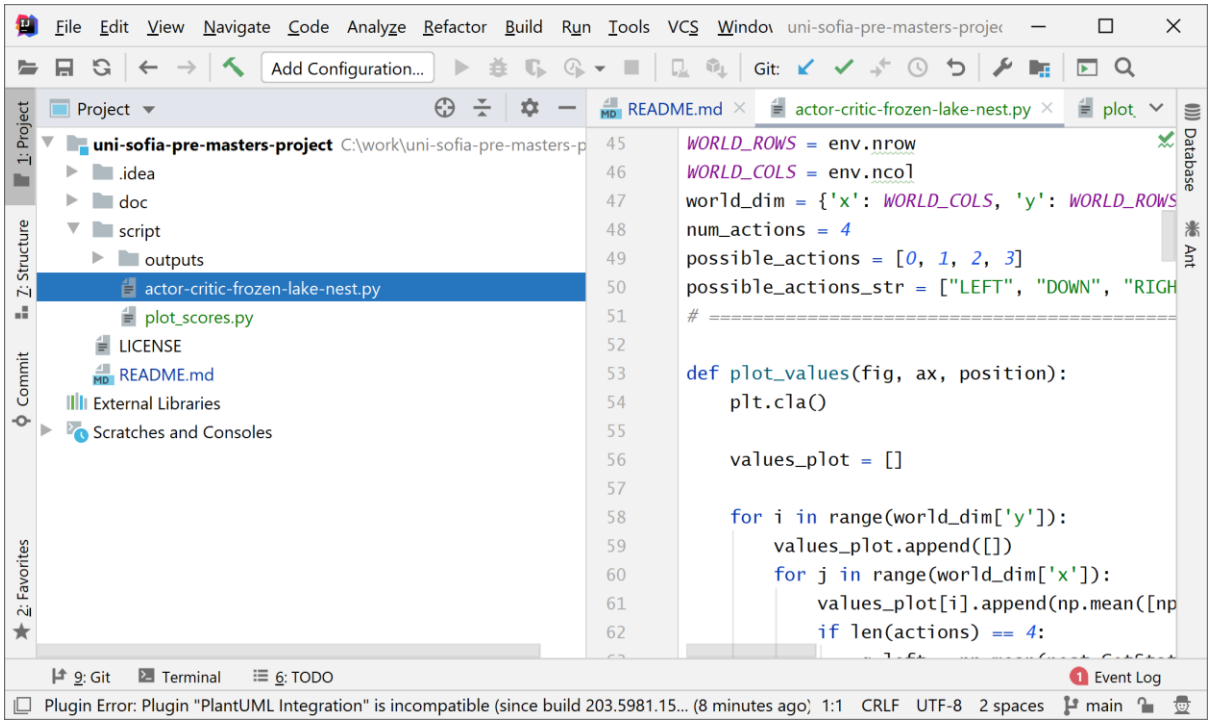
Фиг.5.4.3 Подробно означаване на връзките от клетките на “States” към групите от WTA и значението им като посоки за агента.

„Reward Stimulus“ постъпва през групата „DA“. Така „DA“ отчита очакваната награда, а не абсолютната награда, което съответства на TD грешката $\alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$ от (5.1.3) мащабирана с коефициента α, който се променя на всяка стъпка в зависимост от нивата на допамина. Допаминовите синапси в симулатора NEST получават нивото на допамина, генерирано от DA посредством устройството volume transmitter, описан в 5.3. Тъй като средата FrozenLake дава награда само накрая на успешен експеримент, а в междинните стъпки няма награда, то трябва да изчакаме първо алгоритъмът случайно да намери решението, за да има обучени стойности за последния квадрант преди квадранта “G”. Тук е редно да спомена, че не бихме могли да се справим с отрицателна награда без съществено да променим постановката, тъй като обемният трансмитер на допамин работи на базата на генерирани спайкове, които винаги са положително число (няма как да генерираме отрицателни спайкове).

6. Реализация на проекта

Проектът е реализиран като github публичен проект и може да се разгледа и през браузър (виж Приложения). За да се пусне локално се изисква инсталация на Python, конкретно тук използваме “Python 3.11.0” заедно с Conda (независим от езика мениджър на пакети и система за управление на средата). Използвана е операционна система Линукс – Ubuntu. Връзка към сорс кода е качен в гитхъб (Вж. Приложение 1) и е неразделна част

от този документ. Структурата на приложението е дадена на фигура 6.1. Използваната среда за текстообработка и работа с git е IntelliJ .



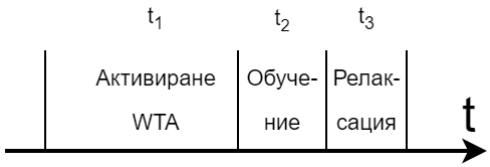
Фигура 6.1. Обща структура на проекта

Подробни инструкции на са дадени в README.md файла.

В централната папка има папка „script“ и в нея имаме “actor-critic-frozen-lake-nest.py” на програмния език Python. С него се стартира процеса на обучение. По време на обучение резултатите от точките (поощрението) се записват във файл „script/outputs/3x3/scores.txt“ за последваща визуализация. Скриптът „script/plot_scores.py“ ще ни визуализира картинка с резултатите след текущото обучение. Процесът на обучение и работа на вече обученият агент не са разделени. За край на обучение се приема момента, когато средно аритметичната награда от последните SOLVED_HISTORY_SCORES_LEN=10 епизода е над SOLVED_MEAN_SCORE=0.5.

6.2 Експериментална част

Основното при такъв тип симулация е как ще се извършва времеделенето и симулацията. Има вариант при който симулацията върви непрекъснато и невронната мрежа получава въздействие от средата чрез външен интерфейс. Тук в този проект е избран по-прост начин, а именно чрез цикъл в който се редуват обучение и въздействие.



Фиг.6.2.1 Времеделене при симулация

На фигура 6.2.1 е показано как става това. В главният цикъл на програмата в който се управлява посоката на агента имаме за всяка стъпка тези три времена, които се редуват за всяка стъпка от дадения експеримент. По-надолу на фиг. 6.2.2 е обяснено по-подробно а времената са споменати в стъпки 5, 8, и 9 от фигурата. Времето t_1= 150ms (в кода означено като константа STEP) е времето в което се активира кръгът Winner Take All за избор на едно от четирите действия. На самите неврони им трябва някакво техническо време да се установи кой ще спечели състезанието и да може ефективно да потисне останалите. Това време може да е от порядъка на 100ms до към 400ms. Опитът показа, че по-големи стойности не променят резултата, а по-малки не дават сигурен резултат. Времето t_2=20 (в кода означено като LEARN_TIME) е времето в което се обучават допаминовите връзки, но само при положителна награда. Времето t_3=50ms (в кода съответно REST_TIME) е времето в което се успокояват невроните от WTA за да се върнат в изходна позиция готови за ново възбуждане през следващият цикъл. Ще аргументирам защо времената са подредени в тази последователност. Интервалът t_2 не трябва да припокрива t_1 защото действието още не е взето от WTA и наградата още не е дадена от средата, съответно няма какво да обучаваме все още. Докато действа t_2 невроните от WTA трябва да са във възбудено състояние, за да е ефективно обучението на допаминовите синапси по закона на Хеб, а именно че възбудените неврони по едно и също време усилват връзката си.

Фиг.6.2.2 Опростена блок-схема на обучението на агента

6.2.1 Обучение при Frozen Lake 3x3 без хлъзгане

Можем да пуснем агента да бъде обучен на среда с 3x3 за да видим как ще се държи обучението и да анализираме резултатите. За целта програмата е направена да приема определени параметри от командния ред, за да не се налага всеки път да се променя сорс кода. Опциите на командния ред са както следва:

1. “-e” – избор на среда, възможни „3x3“ или „4x4“
2. “-s” – избор на хлъзгавост, “true” или “false”
3. “-o” – избор на изходна директория за резултати
4. “-c” – избор на изтриване на изходната директория, “true” или “false”
5. “-n” – максимален брой епизоди

Пускането от операционна система Линукс става по следния начин:

```
python actor-critic-frozen-lake-nest.py -e 3x3 -s false -o outputs/3x3 -c
true -n 60
```

Таблица. 6.2.1.1 Пускане на скрипта за обучение на агента с 3x3 квадранта

Средата FrozenLake 3x3 програмно се прави по следният начин, даден на по-долната таблица. За всеки квадрант задаваме една от латинските букви „S,F,H,G“.

```
env = FrozenLakeEnv(desc=["SFF",
                          "FFH",
                          "FFG"], is_slippery=False)
```

Таблица 6.2.1.2 Примерно инстанциране на средата FrozenLake 3x3

На терминала отпечатваме текущото състояние на средата с функцията „env.render()“, което е текстово представено и масив със сума по всички посоки на допаминовите тегла.

```
[[0.029 0.01  0.   ]
 [0.01  0.402 0.   ]
 [0.482 2.679 0.   ]]
State position:  0 ,  1
SFF
FFH
FFG
  (Up)
chose action: 0   LEFT  at step  7
```

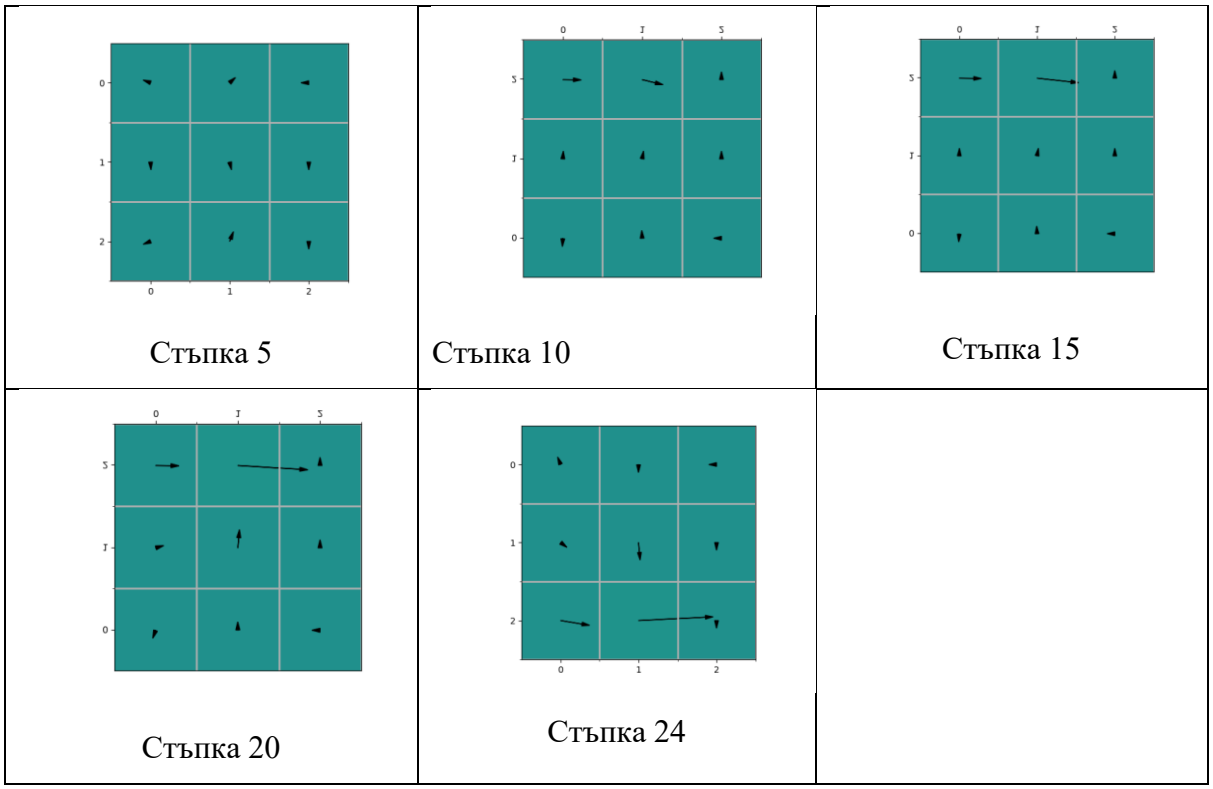
Фиг. 6.2.1.1 Междинен резултат от конзолата от обучението.

Тук на Фиг.6.2.1.1 виждаме двумерен масив с числа, изобразени с точност до третия знак след десетичната запетая, представляващи сума за конкретния квадрант по всички посоки от звеното States към Actions. Този масив се подготвя от функцията „plot_values()“. След това на фигурата се вижда текстова репрезентация от средата, като червената клетка (при положение, че терминалът поддържа цвят) е текущото положение на агента в текущия експеримент 3x3. Отдолу се вижда текущо избраното действие (Up) и следващото действие LEFT. Обучението завършва успешно и резултатът се отпечатва на терминала.

```
Episode 20 finished after 10 timesteps
SOLVED
[[0.029 0.01  0.   ]
 [0.014 0.402 0.   ]
 [0.829 2.906 0.   ]]
====== all_states === all_actions ===
 source   target   synapse_model   weight   delay
-------- -------- --------------- -------- -------
      1      181    dopa_synapse  0.09612   1.000
      1      271    dopa_synapse -0.004897   1.000
      1      361    dopa_synapse -0.007254   1.000
.................
```

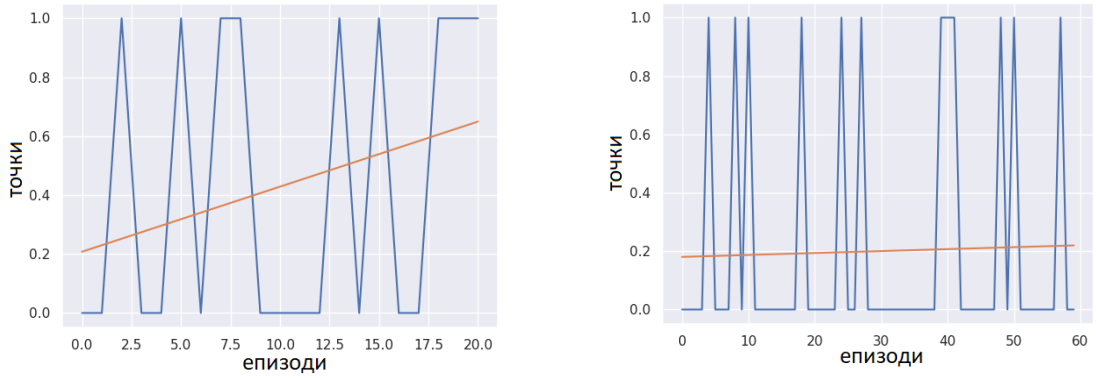
Таблица 6.2.1.3 Край на обучението на агента за размерност 3x3

Като резултат имаме и картинки в директорията “script/outputs/3x3”. Ще разгледаме картинките изобразяващи как се променят коефициентите на теглата States към Actions на всеки 5 стъпки.



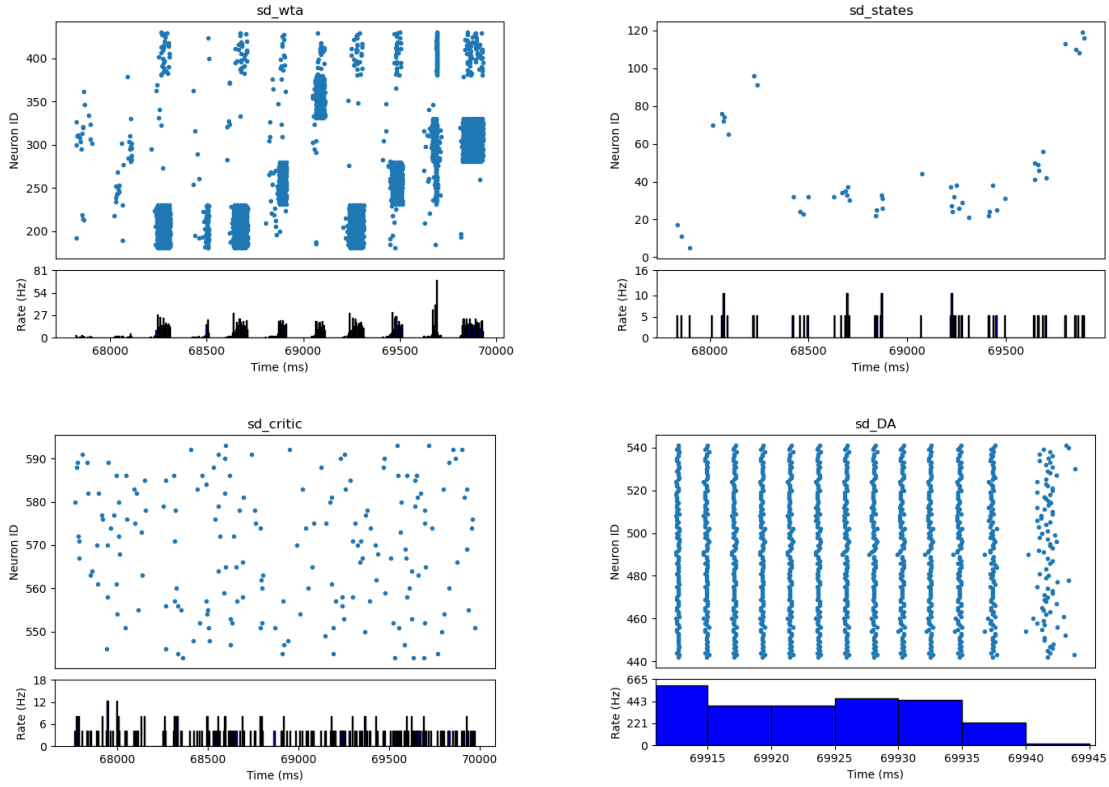
Фиг.6.2.1.2 Визуално представяне на вероятната посока на агента

Да разгледаме резултатите от епизодите. Така ще преценим до колко добре се е обучил агента. На следващата фигура е показан резултат от обучението и базова линия на агент държащ се на случаен принцип.



Фиг.6.2.1.3 Резултат от трениране на 3x3, Ляво: трениран агент, Дясно: агент със случайно действие.

Виждаме, че обучението надминава значително случайното действие - една подходяща отправна точка в случая. Да разгледаме невронните групи и техните спайкове. Тъй като данните от спайковете на епизодите и от стъпките на всеки епизод са твърде много, е показан само последният успешен епизод.



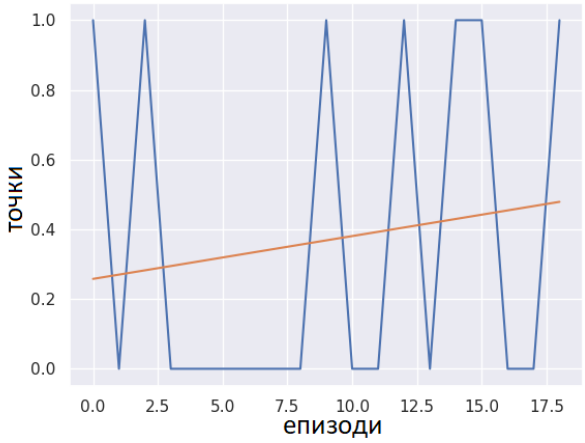
Фиг.6.2.1.4 Импулси от невронните групи, Neuron ID – идентификатор на неврона, Rate(Hz) – честота в херци, Time – време в милисекунди: sd_wta –

импулси във времето генерирани от групата Actions в схемата WTA, sd_states – импулси във времето генерирани от групата States, sd_critic – импулси във времето генерирани от групата Critic, sd_DA – импулси във времето генерирани от групата DA

На горната фигура в раздела sd_wta, което представлява звеното Actions, се виждат интервали с повишена активност на определени групи от неврони, което съответства на вземането на решение. Това е резултатът от последният успешен епизод, който има 11 стъпки. Може да се види от файла “outputs/3x3/output_log.txt” какви са били действията на всяка стъпка. Така например най-долните сини зони означават Ляво. На последната стъпка WTA изглеждат най-уверено, и именно това е Дясно, защото това е действието как от квадрант (1,2) преминава към крайна точка (2,2) (Вж.Фиг.6.2.1.2 последната подфигура). При sd_states се вижда как се активира само по едно състояние в даден момент от време, обозначаващо положението на агента върху дъската. При sd_critic невронната активност е непонятна, но това, което очакваме, е че трябва да е равномерно разпределена. При sd_DA графиката не обхваща целия епизод, а само 30 милисекунди и това е на последната стъпка от епизода, където е дадена наградата.

6.2.2 Обучение при Frozen Lake 3x3 с хлъзгане

Можем да преминем към хлъзгав вариант и зададем “-s true” на скрипта и да подадем друга изходна папка „-o ./outputs/3x3_slippery“.

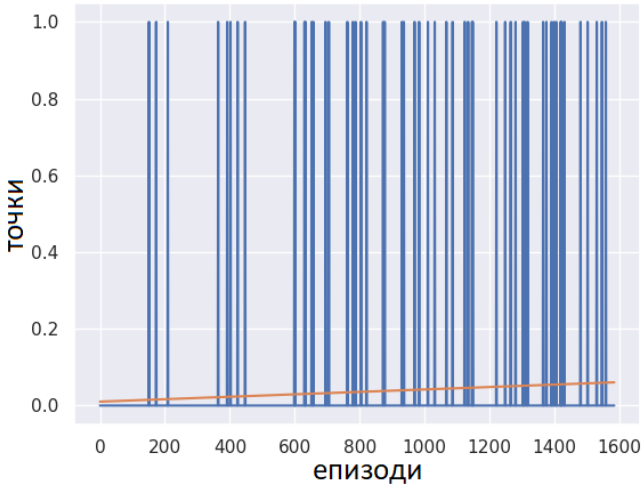


Фиг.6.2.2.1 Резултат при 3x3 с опция за хлъзгава повърхност

На фигура 6.2.2.1 са показани точките от епизодите. Вижда се, че при вариант хлъзгане успеваемостта е сходна за достигане условието за решена задача, но успешните епизоди са по-дълги отколкото успешните без хлъзгане. Това може да се провери ако се разгледат файловете „script/outputs/3x3/output_log.txt“ и „script/outputs/3x3_slippery/output_log.txt“ и е очакван резултат.

6.2.3 Обучение при Frozen Lake 4x4 без хлъзгане

В този вариант има 4 дупки както е видно от фигура 3.1.



Фиг. 6.2.3.1 Резултати от обучение при 4x4 без хлъзгане

Вижда се, че при 4x4 достигане на утвърдително решение не е тривиално. Горната графика е при все още не при достигнато решение, но все пак е видно, че има обучение на агента. Тренирането на подобна задача отнема на DELL XPS, Intel I9 32GB RAM с виртуална машина VirtualBox и Ubuntu около 48 часа. Намалявайки невроните в някои от групите може да ускори изчисленията, но не значително.

6.3 Параметри на постановката и анализ на резултатите

За допаминовите синапси са зададени следните параметри:

```
tau_c = 50.0
tau_n = 20.0
tau_plus = 20.

# Connect states to actions
nest.CopyModel('stdp_dopamine_synapse', 'dopa_synapse', {
    'vt': vol_trans.get('global_id'), 'A_plus': 1, 'A_minus': .5,
"tau_plus": tau_plus,
    'Wmin': -10., 'Wmax': 10., 'b': 0., 'tau_n': tau_n, 'tau_c': tau_c})
```

Таблица 6.3.1 Копиране на модела „stdp_dopamine_synapse“ и подмяна на параметрите

Параметърът γ описан на Фиг. 5.1.2 има стойност 1.0.

На скоростта на обучение можем да повлияем като скалираме поощрението от средата. Към момента това става с емпиричната формула:

```
WEIGHT_SCALING = 100 * 20 / NUM_STATE_NEURONS
nest.SetStatus(nest.GetConnections(reward_stimulus, DA_neurons),
{'weight': float(reward) * WEIGHT_SCALING})
```

Таблица 6.3.4 Скалиране на наградата и влияние на скоростта на обучение

Всъщност скоростта на обучението зависи до колко ще усилим генерираният шум от поасоновия генератор „reward_stimulus“ към допаминовите звена „DA_neurons“ и получената стойност се получава експериментално.

Синапсите имат един механизъм за следа (eligibility trace) в обучението и той може да ни помогне много в текущата задача, но може и да доведе до объркване. Тази следа може да спомогне да проследи откъде е минал агента. Минусът тук е, че ако агентът се е въртял в кръг и не е достигнал целта бързо, то това ще бъде научено в поведението. Това регулираме с константата „tau_c“ когато копираме модела „stdp_dopamine_synapse“. Например възможно е агентът да се върти в кръг и да минава през едни и същи състояния без да достига до край и без да попада в дупка .

Друг полезен механизъм е регулиране на базовата концентрация на допамин „b” от таблица 6.3.3. Давайки положителна стойност, например 0.01 е все едно имаме награда

от средата на всяка стъпка, без да има реална награда. Вътрешният механизъм е заложен в STDP обучението на невроните.Реализирането на ε-greedy политика π от алгоритъма става посредством поасонов шумогенератор „wta_noise“ с честота „WTA_NOISE_RATE“ показан на Фиг.5.4.2 долу вдясно. Давайки по-голяма честота, ще засилим силата на случайния сигнал за вземането на решения в „Actions”. Давайки по-малка честота ще имаме по-малко шум и съответно няма да изследваме нови райнои от картата а ще следваме само наученото, което може и да не е оптимално.

Как влияе времеделенето при експерименталната част от Фиг.6.2.1 ? Първото време определено с константа STEP влияе до колко е сигурно активирането на WTA кръга. Ако е малко и недостатъчно, агентът ще избира винаги едно и също действие. Ако е много голямо, тогава ще чакаме много при изпълнение на експериментите. Второто време влияе до толкова до колкото да се обучат допаминовите синапси. Ако е много голямо, тогава стойностите бързо ще се наситят до Wmax=10 на повечето синапси. Ако е много малко, тогава обучението ще е много дълго. То допринася към коефициента за обучение, който не е в явен вид.

Като цяло 3x3 задачата се решава лесно, докато 4x4 е в пъти по-сложна. Също повечето дупки в 4x4 забавят решението. Първият път агентът трябва сам (на базата на случайно поведение) да намери решение. Знаем, че това може да е много дълъг процес. Разбира се, това може да се превъзмогне, ако включим някакво малко базово ниво на допамина „b“=0.01 в началото и после го махнем. Така в началото всяка стъпка, щом не е дупка, ще приемаме за „наградена“ по изкуствен начин. Това ще помогне в началото но после ще ни пречи, защото агента може да зацикли и да стъпва на едни и същи квадранти, без намерение да стига до крайната цел.

6. Идеи за бъдещо развитие и подобрения

Като идеи за бъдещо развитие мога да посоча подобряване на обучението и премахване на времеделенето от Фиг.6.2.2. Това времеделене е сложено заради техническа трудност да обучим само синапсите, които искаме.

Отделно изложеното решение не е точно както теоретичната част и има разминавания, защото тук не работим с числа в непрекъснатата област, а с дискретни сигнали (спайкове). Може да се експереминентира с изложените параметри за да се доближим

повече до теоретичната част и изчистим решението, тъй като в момента има различни по род аномалии.

7. Източници и използвана литература

[1] Sutton R., Barto A. (2018), Reinforcement Learning: An Introduction, The MIT Press, http://www.incompleteideas.net/book/the-book-2nd.html

[2] O’Reilly R. et al. (2020), Computational Cognitive Neuroscience, Open Textbook, freely available, https://ccnlab.org/

[3] Izhikevich E (2005), Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting, The MIT Press, (https://www.izhikevich.org/publications/dsn.pdf)

[4] FrozenLake, OpenGym, https://www.gymlibrary.dev/environments/toy_text/frozen_lake/

[5] NEST simulator, https://nest-simulator.readthedocs.io/en/v3.3/index.html

[6] Gerstner W, Kistler M, Naud R, Paninski L. (2014), Neuronal Dynamics, Cambridge university press, https://neuronaldynamics.epfl.ch/online/index.html

[7] Potjans W., Morrison A., Diesmann M. (2010), Enabling functional neural circuit simulations with distributed computing of neuromodulated plasticity, Frontiers in COMPUTATIONAL NEUROSCIENCE.

Приложения

1. Сорс код (Source code)

https://github.com/borkox/uni-sofia-pre-masters-project