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: 01.2023г.

# 1. Съдържание

[1. Съдържание 2](#_Toc127382367)

[2. Увод 3](#_Toc127382368)

[3. Средата „FrozenLake“[4] в Gym 3](#_Toc127382369)

[4. Въведение в неврологичните симулации 5](#_Toc127382370)

[4.1 Основи на неврологията 5](#_Toc127382371)

[4.2 Математически апарат на невроните 6](#_Toc127382372)

[4.3 Невробиологичен симулатор NEST 7](#_Toc127382373)

[5. Подход за решаване на задачата 8](#_Toc127382374)

[5.1 Таблични методи 10](#_Toc127382375)

[5.2 Победителят печели всичко 12](#_Toc127382376)

[5.3 Постановка за решаване на задачата 14](#_Toc127382377)

[6. Реализация на проекта 17](#_Toc127382378)

[6.2 Експериментална част 18](#_Toc127382379)

[6.2.1 Обучение при Frozen Lake 3x3 без хлъзгане 19](#_Toc127382380)

[6.2.2 Обучение при Frozen Lake 3x3 с хлъзгане 23](#_Toc127382381)

[6.2.3 Обучение при Frozen Lake 4x4 без хлъзгане 24](#_Toc127382382)

[6.3 Параметри на постановката и анализ на резултатите 24](#_Toc127382383)

[6. Идеи за бъдещо развитие и подобрения 27](#_Toc127382384)

[7. Източници и използвана литература 28](#_Toc127382385)

[Приложения 28](#_Toc127382386)

[1. Сорс код (Source code) 28](#_Toc127382387)

# 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 ще 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 процеса на изработка на проекта ще се изпробват различни параметри на биологично подобните неврони и решението ще бъде илюстрирано с подходящи визуализации и графики съпътстващи обучителния процес.

# 3. Средата „FrozenLake“[4] в Gym

Средата Frozen Lake е част от подпакета а 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 дествия. 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 и да може да се прави сравнение на резултатите. На следващата фигура е показана примерна визуализация на тази среда. Имаме езеро, правоъгълна фигура, разграфена на квадранти, например 4х4 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

Човечето 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 неврологичните симулации

Неврологията е дисциплина която може да засяга един или друг аспект свързан с работата на нервната система при живите организми. За нашите цели по-интересно е математическият апарат, отколкото биологичната основа. Най-забележимият аспект на нервната система при човека и при други живи организми е начинът на вземане на решения и поразяващи имплементации на алгоритми с поощрение и наказание. Връзката между реинфорсмънт обучението (Reinforcement learning) и новрологичните 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 неврологията

Невроните са основните компоненти на нервната система. Това са клетки специализирани в пренасяне и обработка на информация посредством електрохимически и химически сигнали [1]. Невроните са различни по вид, но обикновено имат 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 станат забележително големи и 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 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.

|  |  |
| --- | --- |
|  | (4.2.1) |

Тук формализма е взет от [3], C е капацитетът на мембраната на неврона, V е мембранния потенциал, gleak e функция на утечка, Eleak e средно напрежение на мембраната в покой. 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 сумарно напрежение надвишаващо Ethresh се произвежда токов импулс (spike) и напрежението се връща до стойност Ek. При липса на входни токове или шум се забелязва как напрежението намалява експоненциално, стремейки се към Eleak.

## 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. Подготовка на невронните групи и шумогенератори 2. Свързване на групите неврони с определено тегло 3. Поставяне на виртуални измервателни уреди и свързване с интересуващите ни групи неврони; 4. Симулиране с метода nest.Simulate(t), който проиграва една евентуална симулация за време t милисекунди. 5. След това може да се наблюдават измервателните уреди(волтметри, детектори на импулси), да се променят теглата на връзките и други параметри и може отново да се премине към стъпка 4 или към край на програмата. |

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.

# Подход за решаване на задачата

FrozenLake е задача, което съвпада с формализма въведен за Reinforcement Learning и можем да кажем, че имаме агент и среда, взаимодействащи си чрез марковски процес. След всяко действие на агентът At средата ни отговаря с ново състояние St+1 и се дава съответната награда Rt+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:

|  |  |
| --- | --- |
|  | (5.1) |

Функцията „p“ е плътност на вероятност при условие, че средата се намира в състояние „s“ и агентът е предприел действие „а“ то средата да премине в ново състояние „s‘” и да има награда ”r”. Знакът за равенство с точка означава „по дефиниция“. Тук интуитивно можем да съпоставим клетка от таблица на всеки квадрант от FrozenLake. Така ще имаме за всяка посока изчислени стойности какво би ни донесло всяко действие по дадената посока.

За някои приложения е подходящо да се направи извадков модел от опита, за други задачи е подходящо да се направи дистрибуционен модел. От многото алгоритми за решаване на подобна задача, подходящо е да се построи дистрибуционния модел MDP(Марковски процес на решенията), тъй като имаме изброими състояния на средата и можем да намерим функцията „p” от (5.1) и ще можем въз основа на нея да дефининираме полица 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 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”.

# Таблични методи

Същността на обучение с поощрение и наказание в най-простата си форма ни кара да търсим такива множества от състояния и пространства на действията, които да са достатъчно малки, че да могат да ни апроксимират функцията на натрупаните награди(value function) като масиви или таблици. В такива случаи методите могат да намерят точната функция на натрупаните награди и да стигнат до оптимален алгоритъм(policy). Табличните методи са фамилия от алгоритми използващи динамично програмиране за да стигнат до оптималния дистрибуционен модел MDP(Markov decision process)описан с функцията от (5.1). Тук следва да напомним, че “p” е разпределение на вероятност за всяко s,a и сумата е 1.

|  |  |
| --- | --- |
|  | (5.1.1) |

При MDP ние съответно оптимизираме функцията q\*(s,a) за всяко действие „a“ и за всяко състояние „s“ или апроксимираме функцията v\*(s) за всяко състояние „s“ давайки оптимални действия. Такива количества, зависещи от състоянието “s” могат точно да ни насочат къде е оптималното решение при забавена награда и да ни посочат оптималните действия. Оптималната функция на Белман „q\*(s,a)“ има следният вид (вж. [1] глава 3.6)

|  |  |
| --- | --- |
|  | (5.1.2) |



Фиг.5.1.1 Диаграма на избор на действие за оптимална “q\*” функция. Адаптирано от [1] глава 3.

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.

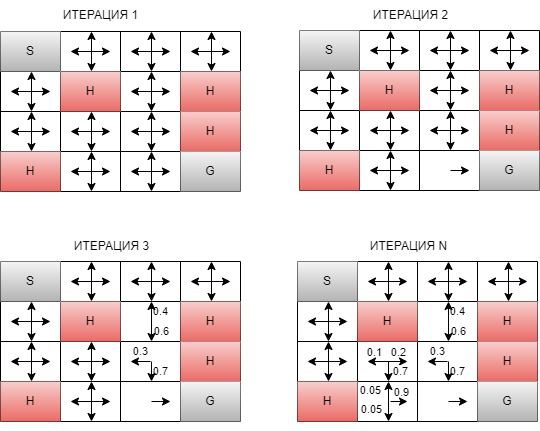
|  |  |
| --- | --- |
|  | (5.1.3) |

Даваме и алгоритъмът разписан в по-долната фигура. Трябва да се има предвид, че това е идеализиран числен метод, а това което се разглежда в текущият проект е само приближение, тъй като се работи със сигнали и имаме сложни динамични процеси с вътрешни шумове и много неизвестни.



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

Тук говорим за ε-greedy(ε-алчен), което на практика означава, че не винаги предприемаме максималното действие, а има вероятност да вземем друго неоптимално действие, с цел да разкриваме нови състояния на средата.



Фиг.5.1.3 Примерни итеративни стъпки за таблично решаване на задачата

На по-горната фигура 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 няшият 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 стабилен еквилибриум, 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 се 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 стабилен и нестабилен еквилибриум 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 устоичиви положения (по една 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.

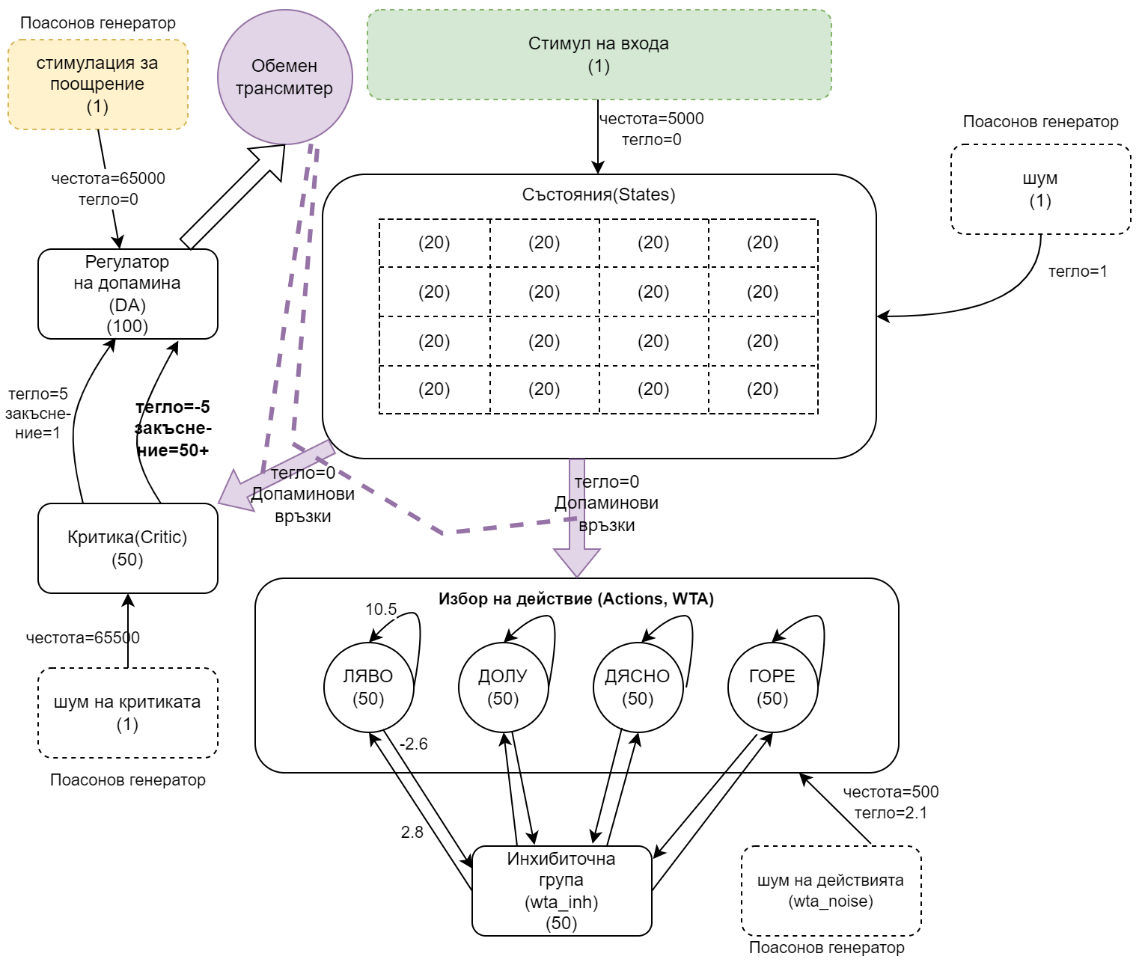


Фиг. 5.2.2 Примерен WTA с два неврона и ефективно потискане. Две популации с възбудими неврони взаимодействат с обща група от потискащи неврони. Адаптирано от [6] глава 16.3.

На фигура 5.2.2 са показани как са свързани две невронни групи. Всяка от групите AE,1 AE,2 действа усилващо на обща група Ainh, която пък от своя страна действа потискащо на АЕ,1 и АЕ,2. Групите АЕ действат самоусилващо с някакво тегло w0. Изборът на действие при WTA става като след някакви милисекунди на симулация, достатъчна да се възбудят невроните, преброим спайковете генерирани във всяка от групите АЕ . Избираме действието отговарящо по индекс.

## 5.3 Постановка за решаване на задачата

За решаване на задачата алгоритъмът е SARSA On-policy TD control, , описан в 5.1 с известна адаптация за невросимулатор. В активността на невроните трябва да се кодират числа. В случаят просто ще сумиране теглата на синапсите свързващи невронните групи като количествена мярка и ще избираме максималното тегло за съответната посока. Ще имаме невронни групи отговарящи за всеки квадрант от таблото на FrozenLake. Всяка една невронна група ще има връзка към съответната група за действие, което ще наподоби Фиг. 5.1.3 Входът от средата ще са невронни групи от по NUM\_STATE\_NEURONS=20 неврона в група, разположени таблично. Ще ги обозначим със „States“ Например ако решаваме 4х4 FrozenLake, ще имаме вход 20х4х4 неврона. При преместване на агента на различен квадрант ще активираме само определената група неврони отговаряща за това състояние с определени координати, например States(0,4). Активацията към определената клетка с NUM\_STATE\_NEURONS неврона на States ще става като на всяка стъпка подаваме шум от генератор на поасонов шум , наречен „Stimulus“ с определена честота и някакво тегло, например 1.0. „States“ ще бъдат вързани към WTA схема от т.5.2 с К=4 възможни състояния като свързването ще е „всеки с всеки“. Всяка група от WTA ще е от по NUM\_WTA\_NEURONS=50 неврона и ще отговаря съответно на действията на агента от 0 до 3 включително, а именно: наляво-0, надолу-1, надясно-2, нагоре-3. Тези невронни групи от WTA съпоставени на действията на агента ще наричаме „Actions“. Връзките на „States“ и „Actions“ ще са с допаминови синапси, първоначално с тегла 0.0 и ще могат да се обучават посредством пластичност STDP (spike-timing dependent plasticity – Markram et al., 1997; Bi and Poo, 1998, 2001),. За STDP също може да се научи повече в [7]. Диаграмата на свързване е дадена на Фиг.5.3.1.



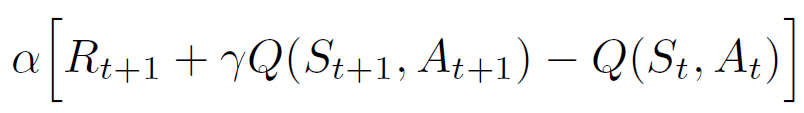
Фиг.5.3.1 Диаграма на свързване на невронните групи

„States“ ще бъдат свързани с друга невронна група, наречена „Critic“ от 50 неврона също с допаминови връзки. Тази нова група ще представя „q\*“ функцията от уравнението на Белман (5.1.2). „Critic“ ще 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 някаква честота пропорционална на наградата. Ще наричаме този вход „Reward Stimulus“.

По подробно можем да направим разбивка на свързването на „States“ и „Actions“ на следващата фигура. Разглеждаме как четирите посоки на агента отговарят на четири връзки от всяка една клетка от „States“. Теглата от всяка една връзка ще означават Q\*(S,A) скалирана с някакъв произволен коефициент, еднакъв за цялата таблица „States“



Фиг.5.3.2 Подробно означаване на връзките от клетките на “States” към групите от WTA и значението им като посоки за агента.

Ще заложим връзка от „Reward Stimulus“ към „DA“. Така получаваме, че в „DA“ ще се отчита очакваната награда, а не абсолютната награда. Това е частта  от (5.1.3) която представлява грешката скалирана с коефициента α. Този коефициент на обучение при нас ще се променя на всяка стъпка в зависимост от нивата на допамина. Именно ако имаме награда повече от очакваната, ще можем да отпушим обучението посредством допаминовия обемен трансмитер (на диаграмата показан като VOLUME TRANSMITTER). Обучение на синапси с допаминов обемен трансмитер е заложен в симулатора NEST под формата на модел „stdp\_dopamine\_synapse“. Увеличавайки спайковете в DA увеличаваме и скоростта на обучение на неврони активирани по едно и също време. Това е същността на STDP като Хебианова пластичност. 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 и така ще имаме обучени стонойности 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. Обемният трансмитер на допамин работи на базата на генерирани спайкове, които винаги са положително число и 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 околната среда). 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 5.1. 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/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.

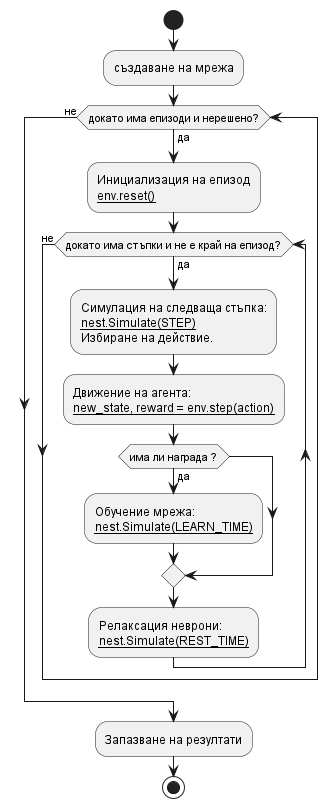
## 6.2 Експериментална част

Основното при такъв тип програми с невросимулатори е въпросът как ще се извършва времеделенето и симулацията. Има вариант при който симулацията върви непрекъснато и се въздейства на невроните от вън чрез външен интерфейс. Тук в този проект е избран по-прост начин, а именно чрез цикъл в който се редуват симулация и въздействие.



Фиг.6.2.2 Времеделене при симулация

На фигура 6.2.2 са дадени времената при главния цикъл на програмата. Времето t1= 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 за 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. По-голямо няма смисъл, защото не променя резултата, а по-малко от посоченото не дава сигурен резултат. Времето t2=20 (в кода означено като LEARN\_TIME) е времето в което се обучават допаминовите връзки, но само при положителна награда. Времето t3=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 t2 не трябва да припокрива t1 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 t2 невроните от WTA трябва да са във възбудено състояние, за да е ефективно обучението по Хебиан, а именно че възбудените неврони по едно и също време усилват връзката си, става дума за States към WTA от Фиг.5.3.2 и съответно States към Critic звеното от Фиг. 5.3.1.



Фиг.6.2.3 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 3х3 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. “-е” – избор на среда, възможни „3x3“ или „4x4“
2. “-s” – избор на хлъзгавост, “true” или “false”
3. “-о” – 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 3х3 квадранта

Средата FrozenLake 3x3 програмно се прави по следният начиn , 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 инстанцииране 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.



Фиг. 6.2.1.1 Междинен резултат от конзолата от обучението.

Тук на Фиг.6.2.4 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 3х3. Отдолу се вижда текущо избраното действие (Up) и следващото действиe 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 3х3

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.

|  |  |  |
| --- | --- | --- |
| Стъпка 5 | Стъпка 10 | Стъпка 15 |
| Стъпка 20 | Стъпка 24 |  |

Фиг.6.2.1.2 Визуално представяне на вероятната посока на агента, апроксимация на q\*(s)

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

|  |  |
| --- | --- |
|  |  |

Фиг.6.2.1.3 Резултат от трениране на 3х3, 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 са много от спайковете на епизодите и от стъпките на всеки епизод, сме взели само последният успешен епизод.

|  |  |
| --- | --- |
|  |  |
|  |  |

Фиг.6.2.1.4 Спайкове от невронните групи: sd\_WTA (Actions), sd\_states(States), sd\_critic(Critic), sd\_DA (DA)

На горната фигура в раздела sd\_WTA, което представлява звеното Actions, се виждат скупчвания на активност, именно това е вземането на решение. Това е от последният успешен епизод, който има 11 стъпки. Може да се види от фала “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.6.2.5 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 „-о ./outputs/3x3\_slippery“.



Фиг.6.2.2.1 Резултат при 3х3 с опция за хлъзгава повърхност

На фигура 6.2.8 сме посочили точките от епизодите 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

Ще пуснем обучение на агента за 4х4 без хлъзгане. Ще припомним, че в този вариант имаме 4 дупки както е видно от фигура 3.1.



Фиг. 6.2.3.1 Резултати от обучение при 4х4 без хлъзгане

Виждаме, че при 4х4 достигане на утвърдително решение не е тривиално. Горната графика все още не при достигнато решение, но виждаме, че обучение на агента има. 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 забърза ичисленията, но не значително.

## 6.3 Параметри на постановката и анализ на резултатите

Първо следва да изброим какви хиперпараметри имаме за обучението и следва да ги анализираме. За да създадем допаминовите връзки, копираме модела „stdp\_dopamine\_synapse“ и подаваме свои 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

Тук задаваме ограничение на връзките до 10 чрез 'Wmin' и 'Wmax'. Принципно биха могли да бъдат и повече, но крайният резултат ще е подобен. 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 6.3.2 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 | Мултипликатор за положително обучение, когато спайкове на източника изпреварват спайковете на целта за два свързани неврона. |
| A\_minus | Мултипликатор за отрицателно обучение, когато спайкове на целта изпреварват спайковете на източника за два свързани неврона. |
| tau\_plus | STDP времева константа, до колко да се считат близо спайковете на източника и целта, в милисекунди. Вж. [7] |
| tau\_c | Времева константа за следа в обучението (eligibility trace) |
| tau\_n | Времева константа за следа при допамина (dopaminergic trace) |
| b | Допаминова базова концентрация |
| Wmin | Минимални тегла на обучаемите синапси |
| Wmax | Максимални тегла на обучаемите синапси |

Таблица 6.3.3 По-важните параметри за допаминови синапси и техните описания

За параметър γ описан на Фиг. 5.1.2 ползваме стойност 1.0. Допълнително скалираме функцията описана като Q(S,A) с някаква константа, например „200“, защото все пак алгоритъмът описва числен метод, докато имплементацията работи със сигнал и няма как да се кодира сигнала точно както е в теоретичното описание. За целта може да се усилват или отслабват дадени връзки за да сме сигурни, че произвеждат спайкове, защото невроните обменят информация само със спайкове.

Коефициентът на обучение е в неявен вид и може да се регулира със скалиране на наградата. 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 Коефициент на обучение α в неявен вид

Всъщност силата 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 обучи няколко невронни групи по-назад във времето, тоест 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“.

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 полица π от алгоритъма става посредством поасонов шумогенератор „wta\_noise“ с честота „WTA\_NOISE\_RATE“ показан на Фиг.5.3.1 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 най-оптимално.

Как влияе времеделенето при експерименталната част от Фиг.6.2.2 ? 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 3х3 задачата се решава лесно, докато 4х4 е в пъти по-сложна. Също повечето дупки в 4х4 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 стъпа, 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, а със сигнали. 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] Reinforcement Learning: An Introduction, 2018, Richard S. Sutton and Andrew G. Barto, [PDF] <http://www.incompleteideas.net/book/the-book-2nd.html>

[2] Computational Cognitive Neuroscience, Randall C. O’Reilly et al., 2020, Open Textbook, freely available, <https://ccnlab.org/>

[3] Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting, Eugene M. Izhikevich, 2005, 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] Neuronal Dynamics, Wulfram Gerstner, M.Kistler, R.Naud, L.Paninski, 2014, Cambridge university press, <https://neuronaldynamics.epfl.ch/online/index.html>

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

# Приложения

## 1. Сорс код (Source code)

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