СОФИЙСКИ УНИВЕРСИТЕТ “СВ. КЛИМЕНТ ОХРИДСКИ”

Факултет по математика и информатика

Катедра „Компютърна информатика”

ДИПЛОМНА РАБОТА

на тема

„Решение на задача за reinforcement обучение на обект с непрекъснати състояния с невробиологичен симулатор NEST“

Дипломант: Борислав Стоянов Марков

Магистърска програма: Изкуствен Интелект

Факултетен номер: 0MI3400048

Научен ръководител: Проф. Петя Копринкова-Христова,

Институт по Информационни и Комуникационни

Технологии (ИИКТ), БАН

София: 2023г.

Съдържание

1. Увод

1.1 Мотивация

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).

1.2 Цел и задачи

Целта на дипломната работа 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 вземане на решения за обект с непрекъснати състояния от пакета Gym. Задачите са да се подготвят варианти на скриптове способни да илюстрират решение, както и да се обосноват стъпките довели до всяка основна промяна на тези скриптове, довеждащи до по-добър резултат.

Дипломната работа 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 разработка на магистърската теза се изпробват различни параметри на биологично подобните неврони и решението е илюстрирано с подходящи визуализации и графики, съпътстващи обучителния процес.

1.3 Очаквани ползи

В областта на невросимулациите няма много публикуван код с цялостни решения, в голяма част от литературата има само загатнати моменти за експерименти, но за съжаление без съпътстващия код. Поради това скриптовете разгледани тук ще подпомагат текущи и бъдещи дипломанти да вземат и използват текущите постановки за друг тип задачи. Скриптовете лесно могат да бъдат адаптирани за подобни задачи в областта на реинфорсмънт обучението.

1.4 Структура на дипломната работа

Дипломната работа се състои от 7 глави както следва:

1. В Глава 1 „Увод“ се въвежда областта на решавания проблем и се поставят целите на дипломната работа
2. В Глава 2 „Обзор на областта“ се обобщават публикувани изследвания в текущата област с невронни мрежи и с невробиологични мрежи.
3. В Глава 3 „Средата „Cart Pole“ в Gym“ се разглежда текуща формулировка на решаваната задача и се дават общи сведения за средата.
4. В Глава 4 „Въведение в невробиологичните симулации“ са описани математически модели на неврони и работата с NEST симулатор.
5. В Глава 5 „Подход за решаване на задачата“ се описват методи за решаване на реинфорсмънт задачи, разглежда се обучение и синаптична пластичност, както и други техники общи за реинфорсмънт обучението и невронните симулации.
6. В Глава 6 „Реализация на проекта“ се разглеждат варианти на невронни мрежи и имплементация със скриптове на Python, дават се последователни стъпки за подобрения.
7. В Глава 7 „Заключение“ се обобщават проведените експерименти и получените резултати, както и се дават идеи за бъдещо развитие.

2. Обзор на областта

CartPole е популярна задача и има много материали в Интернет за нейното решение или частично решение. Повечето решения се базират на класически изкуствени невронни мрежи, но има и други решения с адаптивна критика и ръчно кодирана логика.

А.Барто и Р.Сътън [14] предлагат дискретизация на непрекъснатите стойности на състоянието на обекта и решение с адаптивен невроноподобен елемент през ранната 1983 година.

Jian Xu [11] прави няколко опита да реши задачата без изкуствен интелект и сравнява решенията с невронна мрежа и без невронна мрежа. Успява да реши задачата за 5 реда код и показва, че решението зависи от последните два параметъра на средата.

```
def theta_omega_policy(obs):
    theta, w = obs[2:4]
    if abs(theta) < 0.03:
        return 0 if w < 0 else 1
    else:
        return 0 if theta < 0 else 1
```

Таблица 2.1 Решение на CartPole с 5 реда код от Jian Xu [11]

Разбира се, това решение е уникално, но в него участва човешкият интелект и не подлежи на адаптация за друг вид задачи.

Kenji Doya [14] предлага сравнение на решение за CartPole с методите на актьор-критика в непрекъснатото пространство на състоянията и градиентни методи, използвайки времевата грешка TD(0) и TD(λ).

Greg Surma [12] успява да намери решение с Deep Q-Learning само в рамките на 130 епизода обучение с класически изкуствени невронни мрежи. Siddharth Kale [13] също има подобна публикация и успява да реши проблема след 200 епизода също с класически изкуствени невронни мрежи.

Huanneng Qiu и колектив [16] предлагат решение в областта на spike neural nets (SNN) и сравняват метода си с класически невронни мрежи със сигмоидалната активационна функция на невроните.

Mahmoud Akl и колектив [17] предлагат метода r-STDP(reward-modulated spike timing dependent plasticity), както и обучение с учител за решаване на няколко задачи от областта на реинфорсмънт обучението, включително и CarPole. Методите за обучение използват вече обучени SNN мрежи, както и такива конвертирани от класически изкуствени невронни мрежи до SNN. Експериментите показват, че при промяна на един или няколко параметъра на средата r-STDP може да помогне да се подобри поведението на вече обучената невронна мрежа. Резултатите за CartPole видимо показват решение

след 300 епизода. Съответно има и експерименти с промяна на балансираното рамо и други параметри на средата.

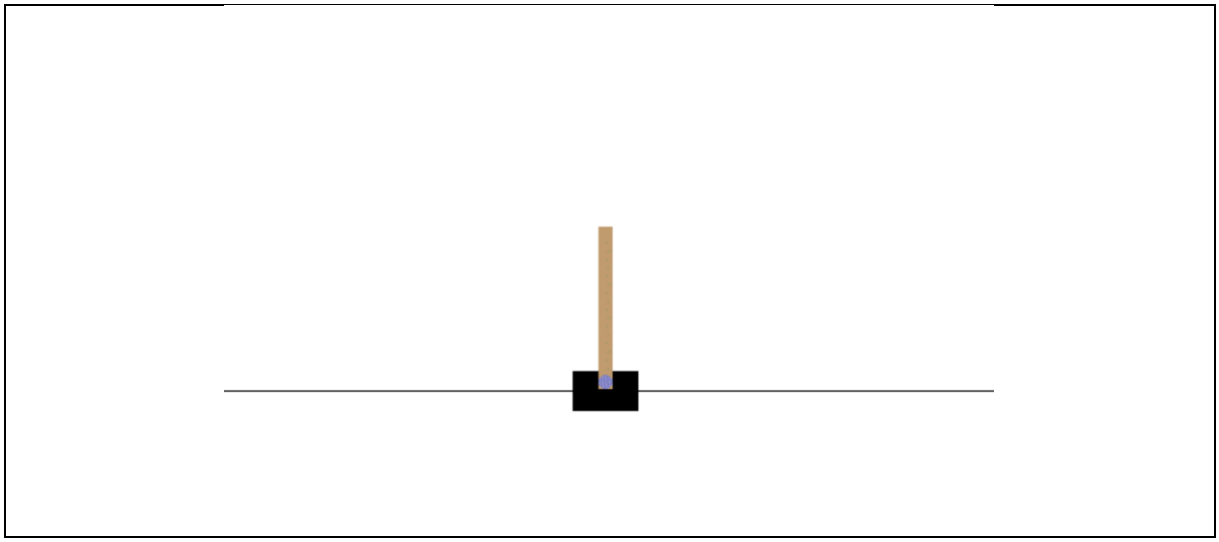
Mahmoud Akl и колектив [18] дават варианти за решение на няколко задачи от реинфорсмънт обучението с дискретни и непрекъснати състояния, включително и CartPole чрез използване на класически изкуствени невронни мрежи, преминавайки към SNN. Използва се алгоритъм за обучение BPTT (Back Propagation Through Time) върху вече конвертираните SNN мрежи. По този начин SNN могат да бъдат тренирани с методи от класическите изкуствени невронни мрежи.

Други задачи от областта на реинфорсмънт обучението с SNN мрежи се предлагат от Zhenshan Bing и колектив [19]. Робот се научава да следва алея, използвайки SNN и алгоритъм за обучение r-STDP. Основните предизвикателства са кодирането на сензорните данни в спайкове на входа и декодирането на сайковете на изхода на мрежата. Предложеният модел не използва предвижданата грешка, въпреки, че в мозъка се наблюдава този феномен. Роботът може да се научи да следва различни шаблони на пътя, които дори се променят в един сценарий.

3. Средата „Cart Pole“[4] в Gym

Средата Cart Pole е част от групата обекти с класическо управление в Gym, които са със случайно начално състояние и непрекъснат вектор на състоянието. Този пакет е разработен за обучителни цели по метода на поощрението и наказанието (Reinforcement Learning, реинфорсмънт обучение). Целта на Gym е да могат да бъдат сравнявани различни решения при една и съща среда.

На следващата фигура е показана примерна визуализация на средата Cart Pole. Обектът (агентът) се състои от количка, която може да се движи наляво и надясно и закрепено към нея рамо с две степени на свобода. Управляващото въздействие е дискретно с две възможни стойности (действия).



Фигура 3.1 – Примерна визуализация на Cart Pole.

При движение налявно и надясно рамото се балансира. При нарушаване на баланса епизодът приключва. Тъй като целта е да се запази баланс за по-дълго, поощрение се дава на всяка стъпка в която рамото не се счита за паднало.

Основна информация за средата е дадена в следващата таблица.

Пространство на действията	`Discrete(2)`
Вектор на състоянието	`(4,)`
Максимални стойности на вектора на състоянието	`[4.8,inf,0.42,inf]`
Минимални стойности на вектора на състоянието	`[-4.8,-inf,-0.42,-inf]`
Импортиране в Питон	`gym.make("CartPole-v1")`

Таблица 3.1. Общи параметри за средата

Агентът може да се придвижва в две посоки, всяка с код от 0 и 1 включително, а именно: наляво-0, надясно-1.

4. Въведение в невробиологичните симулации

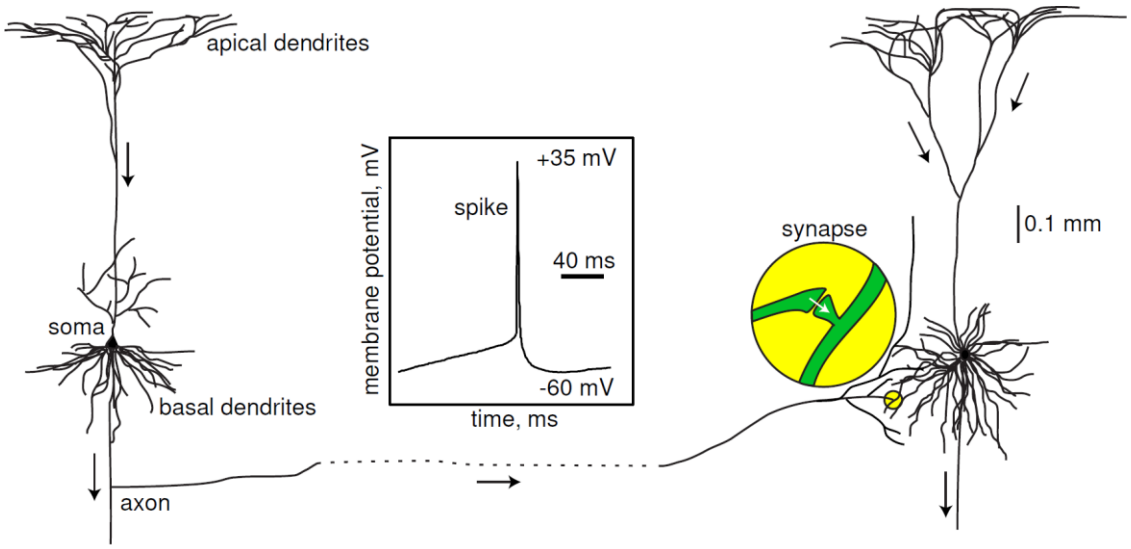
Невробиологията е дисциплина, която засяга различни аспекти свързани с работата на нервната система при живите организми. Настоящата дипломна работа се фокусира повече върху математическия апарат, отколкото върху биологичната основа. Най-

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 генерира 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). След произвеждане на импулс, напрежението на мембраната се връща до равновесния си потенциал.



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

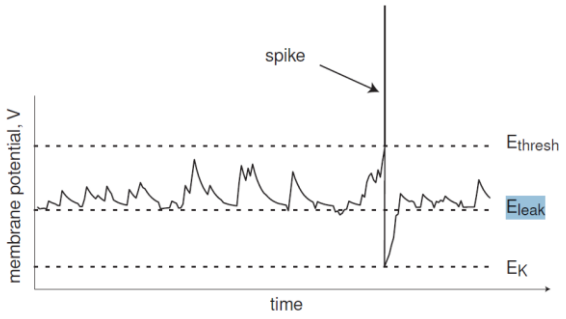
Адаптирано от [3], Глава 1.1

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 [3].

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

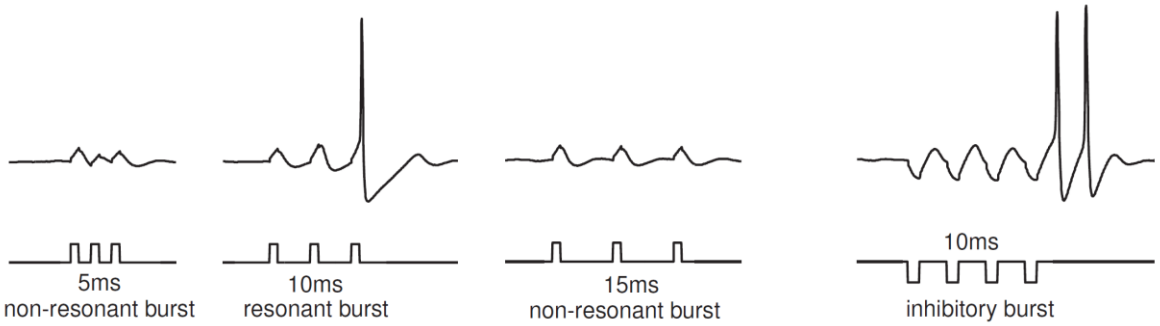
Тук C е капацитетът на мембраната на неврона, V е мембранният потенциал, g_{leak} е проводимостта на клетъчната мембрана, E_{leak} е равновесният потенциал на мембраната. На следващата фигура се онагледява действието на неврона.



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}.

Един феномен, както и трудностите свързани с предизвикване или непредизвикване на импулси от невроните е показан от Изикевич [3] на диаграма с еднакви по амплитуда импулси, но с различен период на излъчване и резултатите са изненадващи.



Фиг. 4.2.2 Резонансен отговор на мезенцефалния V неврон на мозъчния ствол на плъх към импулси инжектиран ток с период от 10 ms. Адаптация от [3].

Неврона се стимулира с три токови импулса. Когато честотата на стимулация е висока (5 милисекунден период), имитирайки силен вход, неврона дори не генерира спайк. Въпреки това, стимулация с по-ниска честота (10 милисекунден период), който резонира с честотата на подпраговите трептения на неврона предизвиква пикова реакция, независимо дали стимулацията е възбуждаща или инхибираща. Стимулация с още по-ниска честота (период от 15 ms) не може отново да предизвика пикова реакция. По този начин невронът е чувствителен само към входовете с резонансна честота. Също импулсите, приложени към кортикален пирамидален неврон, предизвикват отговор само в първия случай (малък период), но не и в останалите случаи.

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. NEST предоставя над 50 невронни модела, много от които се основават на съответна научна публикация. Изборът варира от прости неврони с интегриране и запалване със синапси, базирани на ток или проводимост, през моделите на Изикевич или AdEx до моделите на Ходжкин-Хъксли. 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 симулация за време t милисекунди.

Таблица 4.3.1 Структура на програма използваща NEST simulator

Невронната мрежа в NEST се състои от три основни типа елементи: неврони, връзки между тях (синапси) и устройства за стимулиране или запис. Невроните могат да бъдат подредени в пространствена структура в двумерно или тримерно пространство.

Нови неврони или устройства се създават с командата Create(), която приема като аргументи името на модела на желания тип елемент, по желание броя на елементите, които трябва да бъдат създадени, и инициализиращите параметри. Функцията връща NodeCollection от манипулатори към създадените елементи, които може да бъдат присвоени на променлива за по-късна употреба. NodeCollection е компактно представяне на манипулаторите на елементите, които са цели числа, наречени идентификатори. Много от функции на NEST очакват или връщат NodeCollection. По този начин е лесно да се прилагат функции към големи набори от елементи с едно извикване на функция.

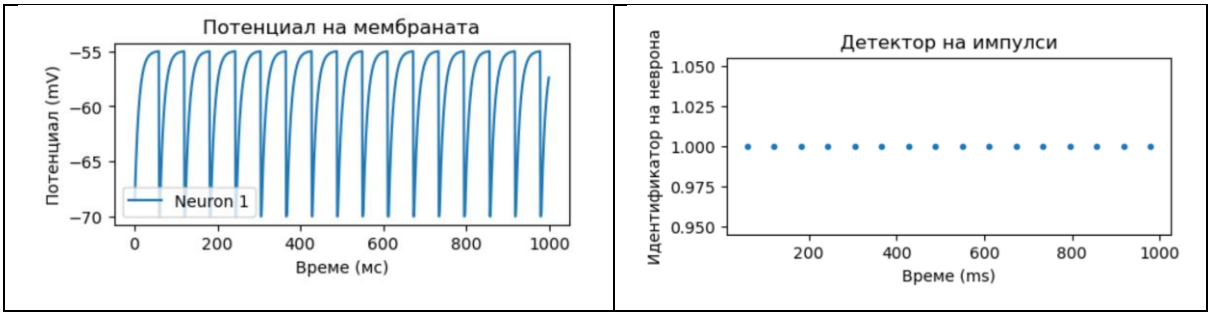
Таблица 4.3.2 показва примерен код на програма, състояща се от един неврон, на който се подава постоянен ток от 376 [pA] и се замерва напрежението на мембраната, както и генерираните импулси.

```
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

Редът, в който са зададени аргументите на Connect(), отразява потока от събития: ако невронът генерира спайкове, той изпраща събитие до записващото устройство за спайкове. Обратно, волтметърът периодично изпраща заявки до неврона, за да поиска неговия мембранен потенциал в този момент. Това може да се разглежда като перфектен електрод, забит в неврона.

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



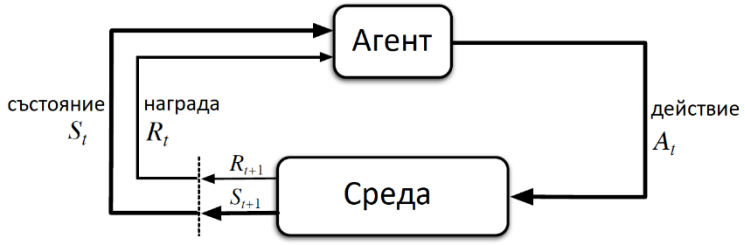
Фигура 4.3.1 Напрежение на мембраната и импулси от неврона

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

5. Подход за решаване на задачата

Обучението с поощрение и наказание (Reinforcement learning) има за цел да научи как да съпоставя ситуациите към действията, така че да се максимизира цифров сигнал за получаване на награда. На обучаемия агент не се казва какви действия трябва да направи, но вместо това трябва да се открие кои действия носят най-голяма награда, като се изпробват различни възможни действия. В редица случаи действията могат да засегнат не само непосредствената награда, но също и следващата ситуация и чрез това всички последващи награди. Тези две характеристики – търсене на принцип проба-грешка и отложено възнаграждение са двете най-важни отличителни черти на реинфорсмънт обучението. То е различно от обучението с учител, едно от най-изучаваните в областта на машинното самообучение. Обучението с учител изисква зададено предварително маркирано обучаващо множество данни, предоставено от външен източник. Реинфорсмънт обучението също е различно и от машинно самообучение без учител, което обикновено е свързано с намиране на структура, скрита в колекции от немаркирани данни. Макар реинфорсмънт обучението да не разчита на примери за правилно поведение, то за разлика от обучението без учител се опитва да увеличи максимално сигнала за награда, вместо да се опитва да намери скрита структура.

CartPole като задача за Reinforcement Learning е формулирана като агент и среда, взаимодействащи си чрез марковски процес. След всяко действие на агента A_t средата 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

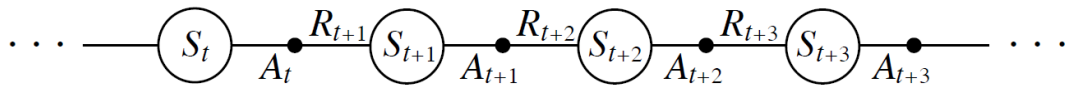
Целта на агента е да максимизира кумулативната награда в дългосрочен план. Поредицата от получаваните наградите след стъпка на времето t означаваме с

$R_{t+1}, R_{t+2}, R_{t+3}, \ldots$. Целта е да се максимизира очакваната награда, означена с G_t, в резултат от поредица от действия. В най-опростеният случай кумулативната наградата е сумата от награди за всяка стъпка:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_{t+T} \tag{5.1}$$

Където T е крайната стъпка в поредицата, наричана **епизод**. Епизодът приключва с успех или неуспех в зависимост от играта. Епизодите са независими един от друг и всеки епизод стартира независимо от това дали предшестващият го е приключил с успех или неуспех.

На следващата фигура е представено схематично как изглежда един епизод на взаимодействие между агент и среда разгънат по време, нарастващо от ляво надясно. Кръговете са състоянията, определяни от средата и бидейки в едно състояние, агентът трябва да определи следващото действие A (action).



Фиг.5.1 Примерен епизод, вж. [1] глава 6.4

Времето за приключване на епизод, T, е случайна променлива и варира за всеки епизод различно. За случаите на процеси без фиксиран краен момент $T = \infty$ формулировката за G_t става неподходяща, тъй като функцията, която трябва да се максимизира става безкрайност. Поради това в теорията на реинфорсмънт обучението се въвежда т. нар. discount фактор γ:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{5.2}$$

Където $\gamma \in [0,1]$. Смисълът е, че награда, получена k стъпки в бъдещето ще си струва толкова колкото и награда получена веднага умножена с γ^{k-1}.

Всички алгоритми за реинфорсмънт обучение включват част за апроксимация на функции-стойност (value functions) – функции на състоянието или на двойка състояние-действие, която ни казва колко добре е агентът да се намира в дадено състояние (или до колко е добро дадено действие, когато агентът е в определено състояние). Терминът „до колко е добро“ се свежда до очакваната награда. Тези функции-стойност са дефинирани

от определено поведение на агента, наричано политика (policy) и означавано с π. По-формално политиката е съответствие между състояние и вероятност да се избере дадено действие от агента. Ако агентът следва политика π във време t, тогава π(a|s) е вероятността A_t=a при S_t=s. Знакът „|“ във функцията π(a|s) определя вероятностно разпределение върху a ∈ $\mathcal{A}(s)$ за всяко $s \in \mathcal{S}$. Реинфорсмънт обучението определя как да се измени политиката на агента в зависимост от резултата от натрупания опит.

Функцията-стойност за дадено състояние s при дадена политика π се означава като $v_\pi(s)$ и е очакваната награда, когато агентът стартира от състояние s и следва политиката π. За марковски процес (MDP, Markov Decision Process) v_π се дефинира като:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|\ S_t = s]\ = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\ \middle|\ S_t = s\right], \forall s \in \mathcal{S} \qquad (5.3)$$

Където $\mathbb{E}_\pi[.]$ означава очакваната стойност на случайна величина, при агент който следва политика π, а t е дискретен момент във времето. Стойността на терминалното състояние е винаги със стойност нула. Функцията v_π се нарича стойност на състоянието (state-value function) за политика π.

Аналогично се дефинира стойността, при избиране на действие “a” от състояние “s”, следвайки политика π ,отбелязвана като q_π (s,a):

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|\ S_t = s, A_t = a]\ = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\ \middle|\ S_t = s, A_t = a\right] \qquad (5.4)$$

Функцията q_π (s,a) се нарича стойност на действието (action-value function) за политика π. Функциите v_π и q_π биват апроксимирани на базата на опита получен от агента.

Решаването на задача с реинфорсмънт обучение се свежда до намиране на политика, която постига висока награда в дългосрочен план. За крайни MDP може точно да се дефинира оптимална политика по следния начин: стойностните функции определят частично подреждане върху политиките и дадена политиката е дефинирана като по-добра от или равна на друга политика, ако нейната очаквана възвръщаемост е по-голяма от или равна на възвръщаемостта от тази на втората за всички състояния. По-формално $\pi \geq \pi$’ тогава и само тогава когато $v_\pi(s) \geq v_{\pi'}(s)$ за всички $s \in \mathcal{S}$. Винаги има поне една политика, която е по-добра или равна на всички останали политики. Тя е оптималната

политика. Възможно е да има повече от една оптимална политика. Всички оптимални политики се означават с π_* . Всички те споделят една и съща функция-стойност v*, дефинирана като

$$v_*(s) \doteq \max_{\pi} v_\pi(s) \tag{5.5}$$

за всички $s \in \mathcal{S}$.

Оптималните политики също споделят една и съща стойност на действието, отбелязвана като q* , и дефинирана като:

$$q_*(s,a) \doteq \max_{\pi} q_\pi(s,a) \tag{5.6}$$

за всички $s \in \mathcal{S}$ и $a \in \mathcal{A}(s)$. Така зависимостта на q* от v* е както следва:

$$q_*(s,a) \doteq \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \tag{5.7}$$

Състоянието на средата S за задачата CartPole е вектор от реални числа. За да се минимизира някоя от функциите-стойност те се апроксимират с параметризирани зависимости, които могат да бъдат обучени. Например функцията v(s) се апроксимира с функцията $\hat{v}(s, \boldsymbol{w}) \approx v_\pi(s)$, където $\boldsymbol{w} \in \mathbb{R}^d$ е вектор от настройваеми параметри. Функцията $\hat{v}$ може да бъде многослойна невронна мрежа от тип SNN, каквато е предложена в следващите глави.

5.1 Методи с темпорална грешка (TD)

Методите с темпорална грешка (TD)[1] и методите Монте Карло използват опит за решаване на проблема с апроксимацията на функцията за стойност. Чрез натрупване на известен опит следвайки политика π, и двата метода актуализират оценката си за V от v_π за нетерминалните състояния S_t, възникващи в конкретният опит. Монте Карло методите изчакват, докато оценката на дадено състояние след посещението стане известна, след което използват тази оценка като цел за $V(S_t)$. Метод Монте Карло за всяко посещение, подходящ за нестационарни среди е:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{5.1.1}$$

където G_t е очакваната стойност след стъпка t, α е коефициент на обучение. Докато Монте-Карло методите трябва да изчакат до края на епизода за да определят увеличението на $V(S_t)$, то TD методите трябва да изчакат само до следващата стъпка. В момента t+1 вече може да се направи актуализация на апроксимираната функция,

знаейки наградата R_{t+1} и имайки апроксимация на $V(S_{t+1})$. Едностъпковия TD метод прави промяната

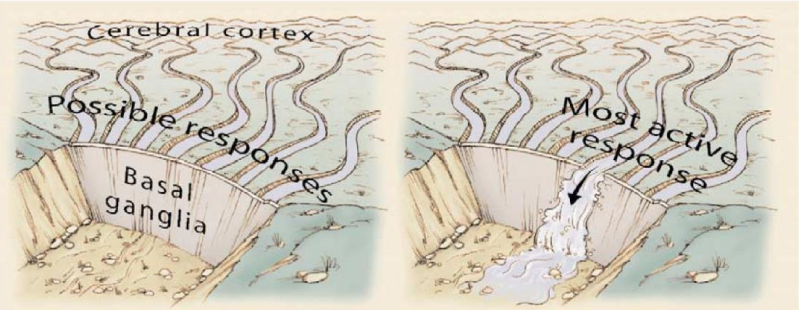
$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (5.1.2)$$

веднага след преминаване в състояние S_{t+1} и получена награда R_{t+1}. Такъв TD метод се нарича TD(0) и той е частен случай на TD(λ) и n-стъпковите методи TD. В следващите раздели се разглежда как се представя апрокцимацията на функциите-стойност за отделните състояния в областта на невронауката и как се извършва оптимизацията за постигане на максимална награда в дългосрочен план в мозъка.

5.2 Контрол на моториката и реинфорсмънт обучението

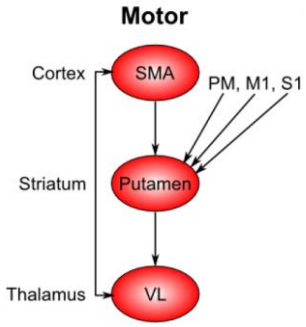
Сетивно-моторната верига включва обработката на сензорните входни въздействия за определяне на желаното двигателно действие в следващ момент. Това е най-основната функция на всяка нервна система. Човешкият мозъкът има огромен брой такива вериги, обхващащи еволюционната времева скала от най-примитивните рефлекси в периферната нервна система, до най-абстрактните нива на вземане на решения, които включват най-високите нива на обработка в префронталната кора (PFC).

На подкорно ниво малкият мозък (cerebellum) и базалните ганглии (basal ganglia) са двете основни контролни области, всяка от които има специално адаптирани механизми за обучение, които се различават от общите кортикални механизми за обучение. Базалните ганглии са специализирани за учене от сигнали за награда/наказание, в сравнение с очакванията за награда/наказание, в следствие на което наученото оформя избора на действие, който организмът ще направи при различни обстоятелства. Именно това е и реинфорсмънт обучение. Малкият мозък (cerebellum) е специализиран да се учи от грешки, по-специално грешки от сензорните резултати, свързани с двигателните действия, по отношение на очакванията за тези сензорни резултати свързани с тези двигателни действия. По този начин малкият мозък може да усъвършенства изпълнението на даден двигателен план, за да го направите по-точен, ефективен и добре координиран.



Фиг. 5.2.1 Ролята на базалните ганглии в избора на действие. Cerebral cortex - мозъчна кора, Basal ganglia – базални ганглии, Possible responses – възможни отговори, Most active response – най-активен отговор. Много възможни реакции се считат възможни и базалните ганглии избират тази реакция с най-голяма активност да бъде изпълнена. Фигурата е репродуцирана от [23]

Базалните ганглии влияят на избора на действие в широк диапазон от фронтални кортикални области, благодарение на последователност от паралелни вериги на свързаност. На следващата фигура е дадена схема за избора на действие при моторните действия.



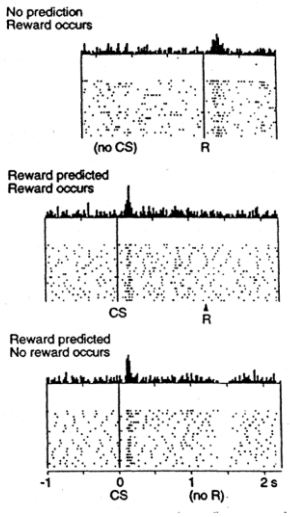
Фиг.5.2.2 Двигателна верига: SMA = допълнителна двигателна зона – свързаният стриатум (putamen) също получава от премоторна мозъчна кора (PM) и първична моторна (M1) и соматосензорна (S1) области – всичко необходимо за правилно контекстуализиране на двигателни действия. Адаптирано от [2], фиг.7.2

Базалните ганглии се учат да избират действията, които носят награда.

Разделението на работата между фронталния кортекс и базалните ганглии е такова, че фронталният кортекс комбинира много различни възможни действия, с обогатени модели на свързаност от други кортикални области, осигуряващи обобщения на високо ниво на текущата среда, които след това активират набор от различни възможни действия, и след това базалните ганглии избират най-доброто (най-вероятно да бъде наградено) от тези действия, за да го изпълнят действително. В по-широк смисъл, фронталният кортекс е размит творчески тип, с милион идеи, но не способност да се фокусира върху реалния свят и му е трудно да стесни нещата до точката на действителност и да направи каквото и да е: нещо като мечтател. Междувременно базалните ганглии са истински тип поемане на отговорност и може да вземе трудните решения и да свърши нещата[2]. Вероятно това е причината там да са две отделни системи (фронтален кортекс и базални ганглии), които въпреки това работят много тясно заедно, за решаване на цялостния проблем за избор на действие.

5.3 Допаминът и ролята му в TD реинфорсмънт методите за обучение

На следващата фигура е представен експеримент, доказващ връзката на определени звена в мозъка и реинфорсмънт обучението с условен стимул и награда.



Фиг.5.3.1 Характерни модели на задействане на допаминергичните неврони във вентралната тегментална област (VTA) и substantia nigra pars compacta (SNc), в проста задача за създаване на условен рефлекс. Фигурата е взета от [22].

Преди обучението за създаване на условния рефлекс, когато бъде получена награда, допаминовите неврони повишават своята активност (горен панел – хистограмата отгоре показва броя на невронни спайкове в текущия момент). След като животното се научи да свързва условен стимул (CS) (напр. тон) с наградата, допаминовите невроните реагират в началото на CS, а не при получаването на самата награда. Ако награда бъде задържана след CS, има спад или пауза в освобождаването на допамин, което показва, че е имало някаква прогноза за наградата и когато тя не успее да бъде постигната, има отрицателна грешка в прогнозата. Този общ модел на възбуждане на невронни групи при различни условия е в голяма степен съвместим с модели за реинфорсмънт обучение, базирани на грешка при прогнозиране на възнаграждението [22].

От изчислителна гледна точка най-простият модел на грешка при прогнозиране на възнаграждението е на Rescorla-Wagner [24] и е просто разликата между действителната награда и очакваната награда:

$$\delta = r - \hat{r} \tag{5.3.1}$$

$$\delta = r - \sum xw \tag{5.3.2}$$

където δ („делта“) е грешката при прогнозиране на наградата, r е действително получената сума на наградата и $\hat{r} = \sum xw$ е очакваното възнаграждение, което се изчислява като претеглена сума от входните стимули x с тегла w. Теглата се адаптират, за да се опитат да предвидят точно действителните стойности на възнаграждението и всъщност тази делта стойност уточнява посоката, в която трябва да се променят теглата:

$$\Delta w = \delta x \tag{5.3.3}$$

Това е идентично с правилото за делта обучение, включително претеглената зависимост от активността на стимула x – иска се промяна само на теглата за стимули, които действително присъстват (т.е. ненулеви x).

Когато предвижданото на наградата е правилно, тогава действителната стойност на наградата се анулира от прогнозата, както е показано във втория панел на Фиг. 5.3.1. Това правило предсказва точно и другите случаи, показани на фигурата (грешки при прогнозиране на положителни и отрицателни награди). Това, което моделът на Rescorla-Wagner не успява да улови, е секретирането на допамин до началото на условния стимул (CS) във втори панел на фигурата. Малко по-сложният модел, представен в 5.1 известен

като темпорални разлики (TD) улавя това започване на стимула (CS), като въвежда време в уравнението (както и името предполага). В сравнение с Rescorla-Wagner, TD просто добавя един допълнителен термин към делта уравнението, представляващо бъдещите стойности на възнагражденията, които може да дойдат по-късно във времето:

$$\delta = (r + f) - \hat{r} \tag{5.3.4}$$

където f представлява бъдещите награди и очакваната награда $\hat{r} = \sum xw$ трябва да се опита да предвиди както текущата награда r, така и тази бъдеща награда f. В проста задача за условния рефлекс, където CS надеждно прогнозира последваща награда, началото на CS води до увеличаване на стойността на f, защото след като CS пристигне, има голяма вероятност за награда в близко бъдеще. Освен това самото f не е предвидимо, тъй като началото на условния стимул (CS) не е предсказано от по-ранно събитие (и ако беше, тогава тази по-ранна реплика би била истинския стимул и щеше да стимулира допаминовия синтез). Следователно, очакването на r-шапка не може да анулира стойността f, и започва производство на допамин. Въпреки че тази f стойност обяснява CS-началото на допаминовия синтез, тя повдига въпроса как може системата да знае какви награди предстоят в бъдеще? Като всичко, свързано с бъдещето, то по същество просто трябва да се познае, като се използва миналото като отправна точка по възможно най-добрия начин. TD прави това, като се опитва да налага последователност в оценките на възнаграждението във времето. На практика оценката в момент t се използва за обучение на оценка във време t-1 и т.н., за да се поддържа всичко възможно най-последователно във времето и в съответствие с действителните награди, които се получават във времето.

Цената на текущото състояние в момента t, означена за удобство с V(t), е сума от всички настоящи и бъдещи награди, като бъдещите награди са намалени с „гама“ фактор, който дава интуитивната представа, че наградите по-нататък в бъдещето струват по-малко от тези, които ще настъпят по-рано:

$$V(t) = \mathbb{E}[G(t)] = r(t) + \gamma^1 r(t+1) + \gamma^2 r(t+2) + \cdots \tag{5.3.5}$$

или в рекурсивен вид:

$$V(t) = r(t) + \gamma V(t+1) \tag{5.3.6}$$

Така оценката на V(t) е:

$$\hat{V}(t) = r(t) + \gamma\hat{V}(t+1) \quad (5.3.7)$$

Това уравнение дефинира оценката на наградата в настоящия момент t, по отношение на бъдещата й оценка в момент t+1. Ако тази оценка е точна, то:

$$0 = \big(r(t) + \gamma\hat{V}(t+1)\big) - \hat{V}(t) \quad (5.3.8)$$

Това математически означава, че темпоралните методи (TD) се опитват да поддържат оценките последователни във времето по такъв начин, че техните разлики трябва да са нула. Следователно грешката при прогнозиране на наградата е:

$$\delta = \big(r(t) + \gamma\hat{V}(t+1)\big) - \hat{V}(t) \quad (5.3.9)$$

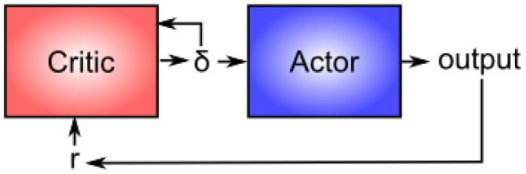
тоест:

$$f = \gamma\hat{V}(t+1) \quad (5.3.10)$$

Правилото за обучение TD може да се използва за обяснение на голям брой различни феномени на създаване на условен рефлекс и съответствието му с активиране на допаминовите неврони в мозъка. То представлява истински триумф на подхода за изчислително моделиране за разбиране (и прогнозиране) на мозъчната функция.

5.4 Актьор-критика архитектура за обучение на моторния апарат

Актьор-критика са фамилия алгоритми, които научават както политики на поведение π, така и функциите стойност V. „Актьорът“ е компонент, който научава политиката за действие, а „критиката“ е компонентът, който прогнозира бъдещите приходи и така помага на актьора да избере действие, което максимизира приходите. Критиката използва TD алгоритъм, за да научи функцията-стойност на състоянието за текущата политика на актьора π. Функция-стойност позволява на критиката да критикува действието на актьора за направените избори чрез изпращане на TD грешки, до актьора.



Фиг.5.4.1 Основна структура на архитектурата актьор-критика за управление на двигателни действия. Critic – Критика, Actor – Актьор, output – избор на действие , Фигурата е взета от [2] Фиг.7.6

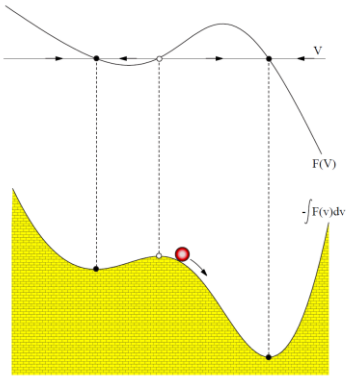
Критиката е отговорна за обработката на сигнала за възнаграждение (r), превръщайки го в грешка при прогноза на възнаграждението (δ), които са подходящи за стимулиране на обучението и на двете, съответно критиката и актьора. Актьорът е отговорен за генерирането на двигателните действия при съответния сензорен вход и не го прави обработвайте награда или очакване за награда директно. Това е ефективно разделение на работата е и от съществено значение за обучението и за трансформиране на наградите в грешки при прогнозиране на наградата, в противен случай системата ще се научи прекалено много само и единствено на прости задачи, които тя би следвало да е овладяла отдавна.

От изчислителна гледна точка, ключовата идея е разграничение между актьора и критиката, където се предполага, че наградите произтичат поне отчасти от правилно изпълнение от актьора. Сигналът за грешка при прогнозиране на възнаграждението, произведен от допаминовата система, е добър тренировъчен сигнал, тъй като стимулира по-интензивно учене в началото на процеса на придобиване на умения, когато наградите са по-непредсказуеми, и намалява силата на обучение с усъвършенстването на умението. Ако системата вместо това научи директно въз основа само на наградата (а не на очакваната награда) то тя ще продължи да учи умения, които отдавна са усвоени и това вероятно би довело до редица лоши последствия [2].

5.5 Победителят печели всичко

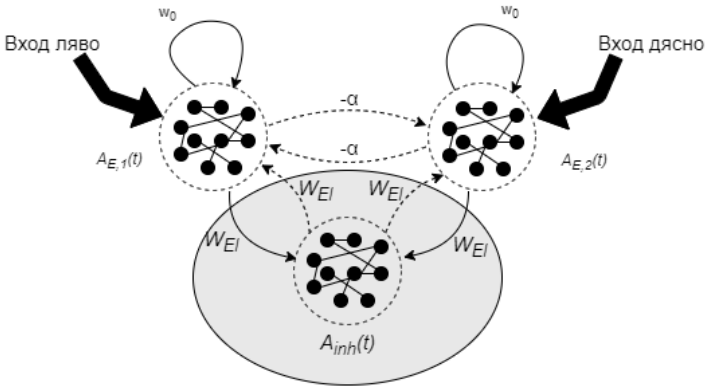
При динамичните системи от неврони изборът на различни действия при различни параметри понякога може да се окаже проблем, тъй като невронните групи навлизат в устойчиво равновесно състояние, и не могат да бъдат изместени от него. За този проблем при динамична система от един неврон споменава Изикевич в [3] в глава 3.2.6.

Механичната интерпретация според [3] на устойчиво и неустойчиво равновесно състояние е показана на следващата фигура.



Фиг. 5.5.1 Механична интерпретация на устойчиво и неустойчиво равновесно състояние. Вж. [3].

Топката на фигурата няма маса (без инерция) и се движи към възможно най-ниската точка със скорост пропорционална на наклона. Ако настъпи промяна в повърхността, по която се движи топката (бифуркация), се променят и възможните равновесни състояния. За да се избегне този ефект се използват схемата „победителят печели всичко“ WTA (Winner Takes All). WTA има един изход от K възможни, като всеки изход е представен от група възбудими неврони (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.

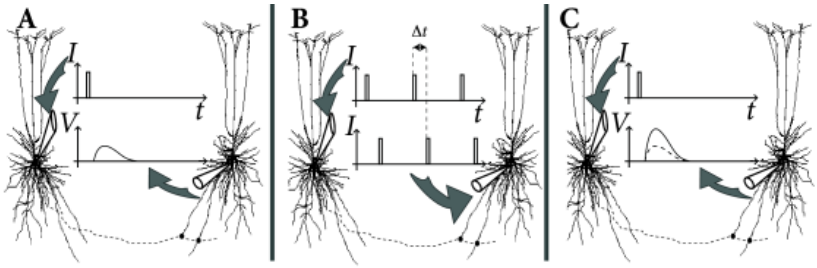


Фиг. 5.5.2 Примерен WTA с два възможни избора и ефективно потискане. Две популации с възбудими неврони взаимодействат с обща група от потискащи неврони. Адаптирано от [6] глава 16.3.

На горната фигура 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 определен интервал от милисекунди на симулация, достатъчна да се възбудят невроните, се преброяват спайковете генерирани във всяка от групите A_E и се избира действието отговарящо на групата с най-много спайкове.

5.6 Обучение с импулсно-времево зависима пластичност (STDP)

Обучението в мозъка има механизъм на промяна на синаптичната ефективност. Исторически теоретичните обосновки се базират на постулата на Хеб [25], 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.5.6.1. 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.*

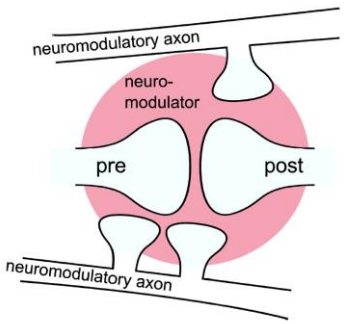
На фигура 5.6.1 са дадени разяснения за STDP като механизъм. Ако моментът на генериране на пресинаптичния импулс е t_{pre} , а на постсинаптичния импулс - t_{post} , то промяната на теглото на пластичния синапс е зависима от времето $|\Delta t| = |t_{post} - t_{pre}|$ (вж.[6] глава 19.2.2) по следния начин:

$$\Delta w_+ = A_+(w).e^{-\frac{|\Delta t|}{\tau_+}} \quad във\ време \quad t_{post} \quad при \quad t_{pre} < t_{post} \tag{5.6.1}$$

$$\Delta w_- = A_-(w).e^{-\frac{|\Delta t|}{\tau_-}} \quad във\ време \quad t_{pre} \quad при \quad t_{pre} > t_{post} \tag{5.6.2}$$

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 „Изливането“ на невромодулатор около синапсите се нарича обемно подаване (volume transmission). Самото активиране на протокола за обучение е обяснено нагледно на следващата фиг.5.6.2.



Фиг.5.6.2 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 5.6.1 Параметри по подразбиране за модела 'stdp_dopamine_synapse'

Значенията на параметрите могат да бъдат намерени от [5] на страницата за модели. Те са показани в таблицата:

Име на параметъра	Значение
A_plus	Коефициент на обучение, когато пресинаптичният импулс изпреварва по време постсинаптичния импулс .
A_minus	Коефициент на обучение, когато постсинаптичният импулс изпреварва пресинаптичния импулс по време .
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

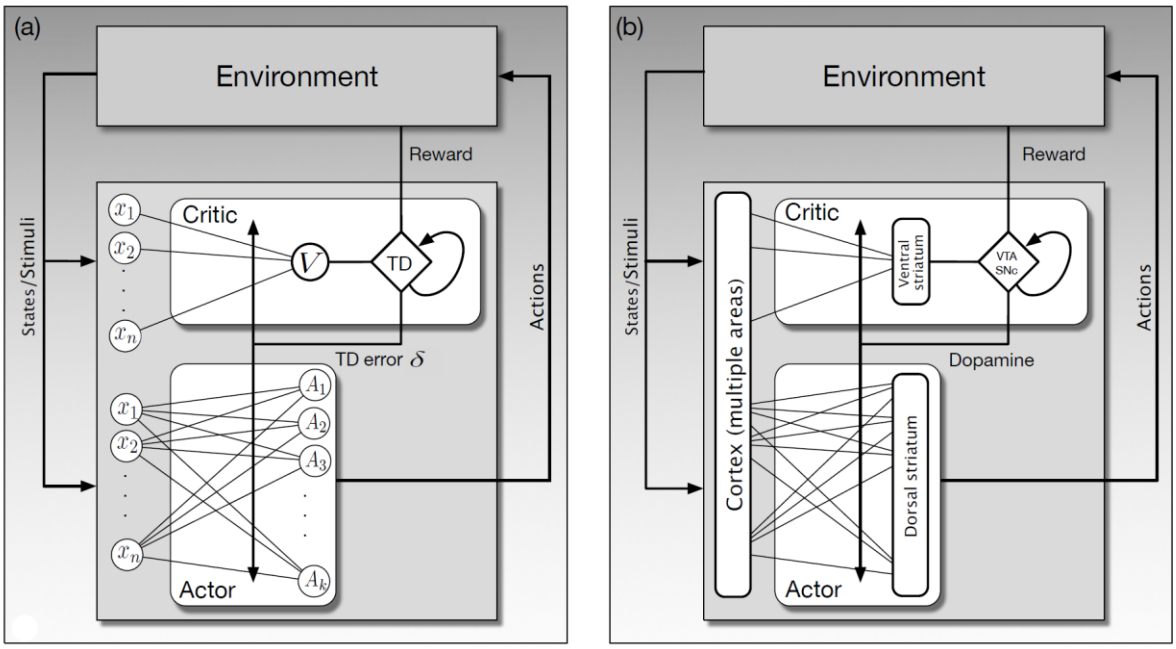
Таблица 5.6.2 По-важните параметри за допаминови синапси и техните описания

5.7 Обща постановка за решаване на задачата

Най-забележимата допирна точка на обучението по метода на поощрение-наказание и невронауката е дълбоката химическа връзка на допамина. В глави 5.3 и 5.4 беше разгледано как допаминът отговаря за преноса на времевата грешка (TD) до съответните структури на мозъка, където се извършва обучение и се взема решение за по-нататъшно действие. За методите основани на времевата грешка във време t тя е

$$\delta_{t-1} = R_t + \gamma V(S_t) - V(S_{t-1}). \quad (5.7.1)$$

Алгоритмите актьор-критика научават, както политиката за актьора, така и предсказването на очакваната награда. Смята се, че две структури в стриатума от мозъка на бозайниците отговарят на елементите актьор и критика, това са Dorsal striatum и Ventral striatum (вж.[1] глава 15.7).

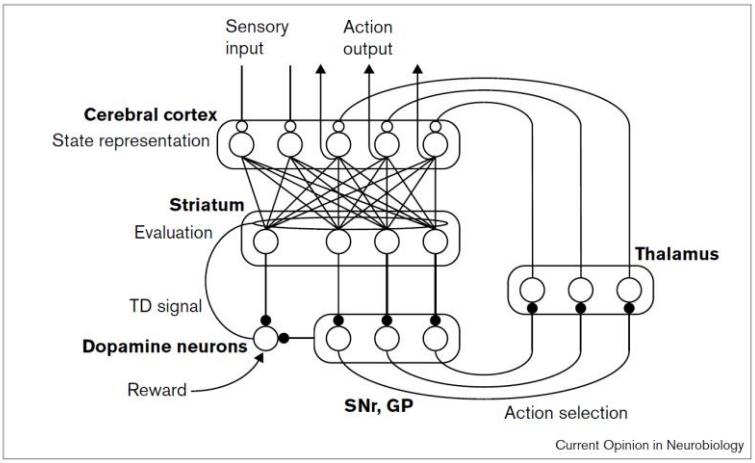


Фиг. 5.7.1 Актьор-Критика с невронна мрежа и хипотетична невронна имплементация. а) Актьор-критика като изкуствена невронна мрежа. Актьорът променя политиката спрямо TD грешката δ, който получава от критиката. Критиката създава грешката TD от сигнала за награда Reward. Актьорът няма директен достъп до Reward сигнала. Критиката няма директен достъп до действието. b) Хипотетична невро-имплементация на актьор критика. Актьорът и компонентът научаващ функцията за стойност са съответно във вентралната и дорсалната части на стриатума. Времевата грешката (TD) δ се предава от допамина, генериран от VTA. Фигурата е копирана от [1] фиг 15.5.

Аналогично в настоящата дипломна работа се използва комбинация от актьор-критика и алгоритъм, научаващ функцията v(s) по схемата от фиг. 5.7.1 (b)

За апроксимация на v(s) от компонента за критика ще се използва алгоритъм подобен на SARSA, с известна адаптация.

Кенджи Дойа [8] представя мненията на неврообщността в своята обзорна статия за ролята на базалните ганглии в реинфорсмънт обучението.



Фиг. 5.7.2 Схематична диаграма на кортико-базалния ганглиен цикъл за контрол на моторните функции и евентуалните роли в реинфорсмънт обучението. Фигурата е взета от [8] фиг.2.

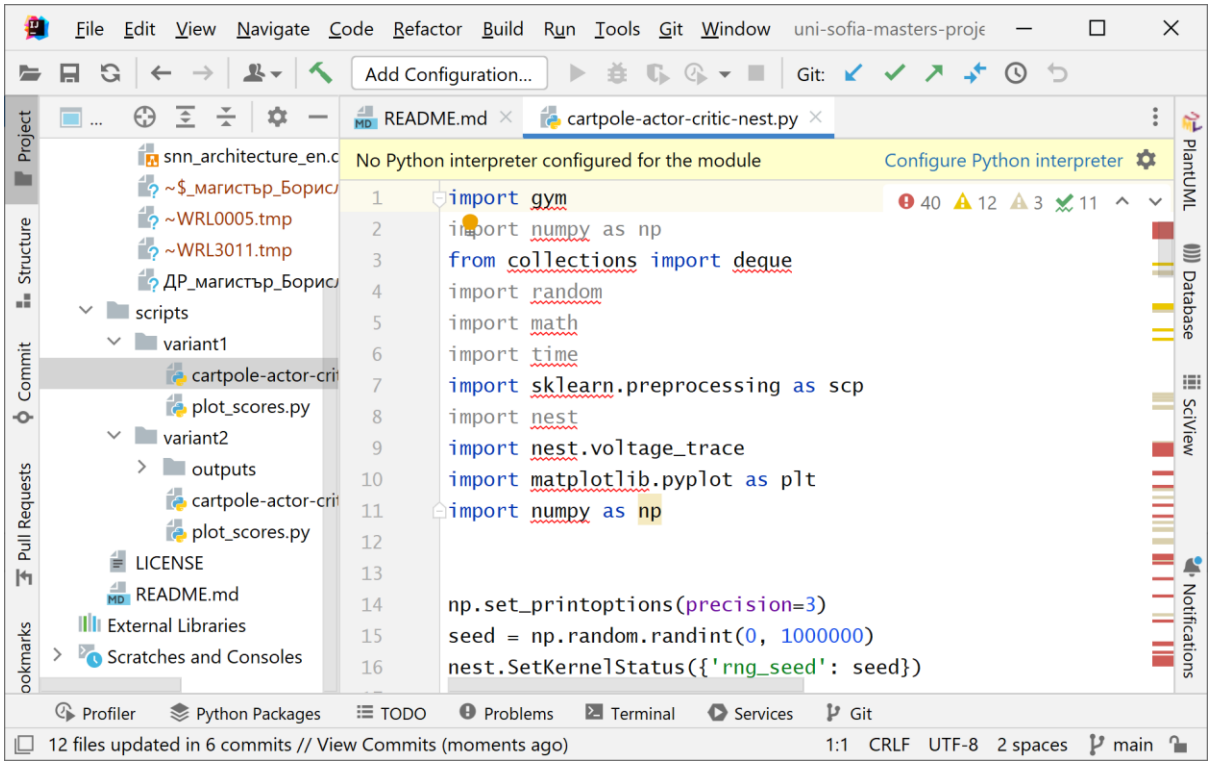
Невроните в стриатума предсказват бъдеща награда за текущото състояние и действията на кандидат-действията. Грешка в прогнозата на бъдеща награда, т.е. TD грешката, е кодирана в активността на допаминовите неврони и се използва за обучение в кортико-стриаталните синапси. Едно от действията-кандидати е избрано в SNr и GP в резултат на състезание на прогнозирани бъдещи награди. Директният и индиректен пътища в globus pallidus са пропуснати за простота на диаграмата. Запълнените и празните кръгове означават съответно инхибиторни и възбуждащи синапси. Тази фигура много прилича на Фиг.5.7.1 по начина на свързване и представяне на TD грешката.

6. Реализация на проекта

6.1 Общи положения

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

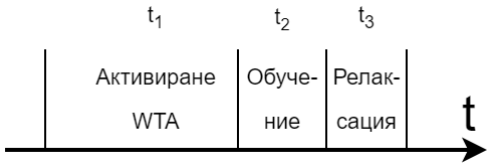
Подробни инструкции на са дадени в README.md файла.

В централната папка има папка „scripts“ и в нея имаме подпапки за вариантите. Всеки вариант има скрипт “cartpole-actor-critic-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 „scripts/variantX/outputs/scores.txt“ за последваща визуализация. Скриптът „scripts/variantX/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 няма посочени конкретни данни на официалният сайт, но някои автори[11] споменават че решение се приема момента, когато средно аритметичната награда от последните 100 епизода (в скрипта е променлива SOLVED_HISTORY_SCORES_LEN) е над 195 точки (в скрипта е променлива SOLVED_SCORE). Различни източници приемат различни точки като решение.

6.2 Експериментална част

Ще бъдат разгледани два варианта за евентуално решение. Като начало се дават общите неща и за двата варианта.

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 тази дипломна работа е избран по-прост начин, а именно чрез цикъл в който се редуват обучение и въздействие.



Фиг.6.2.1 Времеделене при симулация

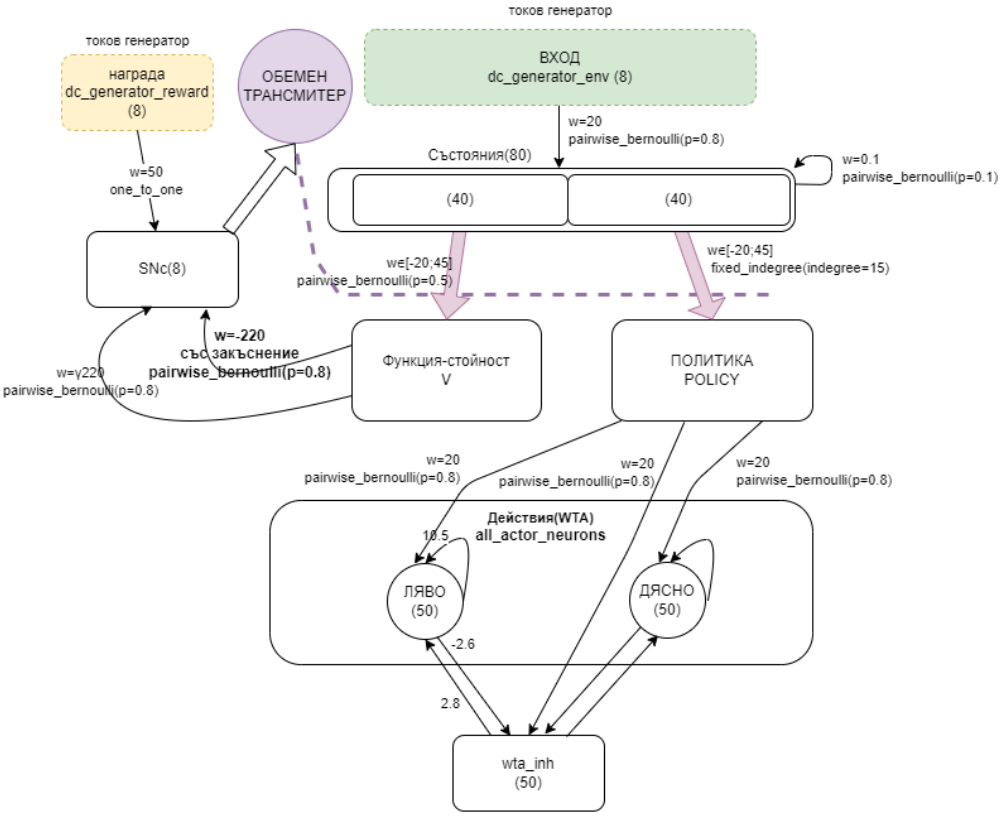
На фигура 6.2.1 е показано как става това. В главният цикъл на програмата в който се управлява посоката на агента всяка стъпка е разделена на тези три интервала. Фиг. 6.2.2 показва блок-схемата на алгоритъма. Времето t_1 (в кода означено като константа STEP) е времето (в милисекунди) в което се активира кръгът Winner Take All за избор на едно от четирите действия. На самите неврони им трябва време да се установи кой ще спечели състезанието, за да може ефективно да потисне останалите. Това време може да е от порядъка на 40ms до към 400ms. Опитът показа, че по-големи стойности не променят резултата, а по-малки не дават сигурен резултат. Времето t_2 (в кода означено като LEARN_TIME) е времето в което се обучават допаминовите връзки, но само при положителна награда и може да варира от порядъка на 10ms до 30ms. Времето t_3 (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 следващия цикъл, може да е от порядъка на 40ms до 50ms. 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 обучение. В интервала 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 и за двата варианта

6.2.1 Вариант 1 за решение на CartPole

Тъй като активността на невроните се моделира със spike timing, се налага преминаването от спайкове към числени стойности на изхода на всяка група неврони. Входът от околната среда се моделира от 8 токови генератора. Състоянието на средата е представено от две групи по 40 неврона, общо 80 неврона. Има една група неврони за функцията-стойност V от 40 неврона и една група за функцията на политиката π от 40 неврона. Свързването е представено на Фигура 6.2.1.1.

При подаване на състояние от средата се активират различни нива на токовите генератори на входа и се активира само определената група неврони от Състояния (STATES), отговаряща за това състояние.



Фиг.6.2.1.1 Диаграма на свързване на невронните групи за вариант 1

Типът свързване на STATES със V е на случаен принцип тип Бернули (pairwise_bernoulli). Това означава, че за всяка двойка неврони от групи A и B връзка ще бъде направена с вероятност „p“. Пример за такъв тип свързване се дава със следния код.

```
n, m, p = 10, 12, 0.2
A = nest.Create('iaf_psc_alpha', n)
B = nest.Create('iaf_psc_alpha', m)
conn_spec_dict = {'rule': 'pairwise_bernoulli', 'p': p}
nest.Connect(A, B, conn_spec_dict)
```

Таблица 6.2.1.1 Примерен код за свързване по двойки тип Бернули.

При увеличаване на бройката неврони във всяка от групите би се увеличил и тока в приемащия неврон, затова свързване тип „всеки със всеки“ не се препоръчва. Връзките между STATE и V са с допаминови пластични синапси, първоначално с равномерно разпределени случайни тегла в интервала [-20;+45] (вж. 5.3) (STDP, spike-timing dependent plasticity, [26] [27]). Формулите за промяна на теглата в опростен вид са (5.3.1) и (5.3.2). Връзката обхваща само едната половина на STATE, т.е. 40 неврона и с код на Python се изразява като STATE[40:].

Свързването на STATES със POLICY е фиксиран брой входящи връзки (fixed_indegree), при което също има случаен елемент. За такъв тип свързване между две групи A и B, всеки неврон от приемащата група B има фиксиран брой входящи връзки. По този начин при нарастване на бройката неврони в групата A сумарните токове в групата B няма да нарастнат. Примерен код е даден в следващата таблица:

```
A = nest.Create('iaf_psc_alpha', 5)
B = nest.Create('iaf_psc_alpha', 3)
conn_spec_dict = {'rule': 'fixed_indegree', 'indegree': 2}
syn_spec_dict = {'weight': [[1.2, -3.5],[0.4, -0.2],[0.6, 2.2]]}
nest.Connect(A, B, conn_spec_dict, syn_spec_dict)
```

Таблица 6.2.1.2 Примерен код за свързване тип фиксиран брой входящи връзки.

Връзките между „STATE“ и „POLICY“ са с допаминови пластични синапси, първоначално със случайни тегла в интервала [-20;+45] (вж. 5.3) (STDP). Връзката обхваща само втората половина на „STATE“, т.е. 40 неврона и с код на Python се изразява като STATE[0:40].

Звеното „STATE“ е свързано към невронните групи за действия (WTA) с тип pairwise_bernoulli (p=0.8). Всяка група от WTA се състои от 50 неврона и отговаря съответно на действията на агента, наречени „action_left” и „action_right“. Връзките между „States“ и „Actions“ не са обучаеми и не се променят.

Групата „V“ са свързани с друга невронна група, наречена „SNc“ от 8 неврона и се състои от връзки с фиксирани тегла, които не се обучават. Връзките са на две групи, от които едната е със закъснение, равна по време на една стъпка, за да формира очакваната грешка δ, както е уточнено в 5.3. При липса на грешка, т.е. δ=0 няма да има активиране на обемния трансмитер за подаване на допамин и няма да има обучение на допаминовите групи. При грешка различна от нула, т.е. δ≠0 ще има активиране на SNc и съответно ще има подаване на допамин към всички допаминови връзки и ще има обучение на теглата на допаминовите връзки.

Разглеждайки гореизложената схема на свързване като Атьор-критика, актьорът са невронните групи POLICY + WTA, докато критиката е V+SNc. Условието актьорът да няма директен достъп до Reward сигнала и критиката да няма директен достъп до действието е изпълнено.

Векторът на състоянието (Observation Space) се състои от 4 стойности.

индекс	Вид наблюдение	Минимална ст-т-	Максимална ст-т
0	Позиция на количката	-4.8	+4.8
1	Скорост на количката	-∞	+∞
2	Ъгъл на балансираното рамо	-0.418 rad	+0.418 rad
3	Ъглова скорост на балансираното рамо	-∞	+∞

Таблица 6.2.1.3 Входни параметри от средата на CartPole

Ако съпоставим четири токови генератора на четирите входни стойности и ги скалираме с някакъв параметър решението би могло да работи частично. За целта на настоящата работа обхвата на всеки един елемент на състоянието е разделен на два интервала, обхващащи положителната и отрицателната част от дефиниционната му област. Въведено е и подходящо скалиране, така че изходящите стойности да са във фиксиран интервал. С код на Python това става лесно с помощта на MinMaxScaler (от пакета scikit-learn) и функция, разделяща всяка стойност на два елемента. Когато позицията на количката е отрицателна стойност, то тя бива разделена на две стойности (0, abs(s(0))). За целта използваме следната функция, дадена в таблица.

```
def transform_state(s):
    transformed = np.array([
        abs(s[0]) if s[0] > 0 else 0,
        abs(s[0]) if s[0] < 0 else 0,
        abs(s[1]) if s[1] > 0 else 0,
        abs(s[1]) if s[1] < 0 else 0,
        abs(s[2]) if s[2] > 0 else 0,
        abs(s[2]) if s[2] < 0 else 0,
        abs(s[3]) if s[3] > 0 else 0,
        abs(s[3]) if s[3] < 0 else 0])
    return transformed
```

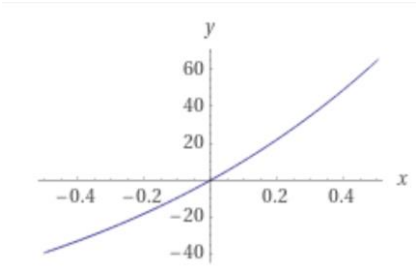
Таблица 6.2.1.4 Преобразуване на входните параметри и преминаване от пространство с размерност 4 към пространство от размерност 8

Разбира се, вероятно има и по кратък начин да се направи подобно преобразуване, но за нагледност и простота, кода е оставен по този начин. След такова преминаване към размерност 8 на входните параметри се извършва скалиране.

```
scaler = scp.MinMaxScaler(feature_range=(0.01, 1), copy=True, clip=True)
scaler.fit([[0, 0, 0, 0, 0, 0, 0, 0], [+1.5, +1.5, +1.5, +1.5, +0.13,
+0.13, +2.1, +2.1]])
...
new_transformed_state_scaled =
scaler.transform(new_transformed_state.reshape(1, -1)).reshape(-1)
```

Таблица 6.2.1.5 Преобразуване на входните параметри към интервал с фиксирана стойност [0.01;1]

Скалирането е с фиксиран интервал на изходящите стойности от 0.01 до 1. Входните минимални и максимални стойности са зададени с фиксирани стойности и са получени на базата на експерименти по емпиричен начин. Така например няма значение дали скоростта на количката е по-голяма от 1.5 или по-малка от -1.5. Допълнително изходящите стойности се преобразуват през експоненциална функция $y(x) = 100.(e^x - 1)$, като идеята е по-ниските стойности да влияят по-слабо и отивайки към по-високи стойности, влиянието да се засилва. Факторът 100 е подбран емпирично.



Фиг. 6.2.1.2 Допълнително експоненциално преобразуване на входа

Таблицата показва примерни стойности на състоянието на средата и техния еквивалент подаван като входни токове към мрежата.

Вход от средата, размерност 4	Трансформация към размерност 8	Трансформация със скалиране	Трансформация $100.(e^x - 1)$ [pA]

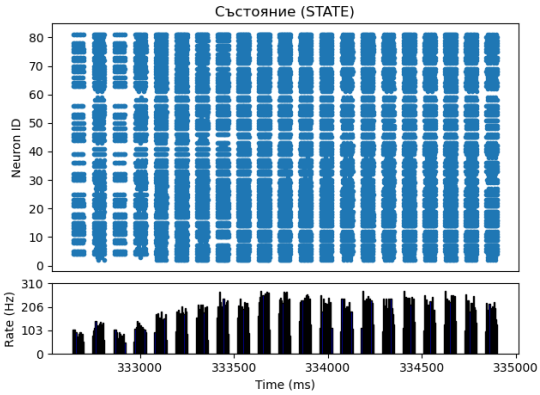
[0.033, 0.016, 0.048, 0.003]	[0.033, 0.0, 0.016, 0.0, 0.048, 0.0, 0.003, 0.0]	[0.032, 0.01, 0.021, 0.01, 0.375, 0.01, 0.012, 0.01]	[3.214, 1.005, 2.095, 1.005, 45.484, 1.005, 1.163, 1.005]
[0.117, 1.38, -0.069, -1.999]	[0.117, 0.0, 1.38, 0.0, 0.0, 0.069, 0.0, 1.999]	[0.087,0.01, 0.921, 0.01, 0.01, 0.538, 0.01, 0.952]	[9.097, 1.005, 151.112, 1.005, 1.005, 71.209, 1.005, 159.201]

Таблица 6.2.1.6 Тристъпково преобразуване на входните параметри от средата CartPole във формат удобен за захранване на токови генератори

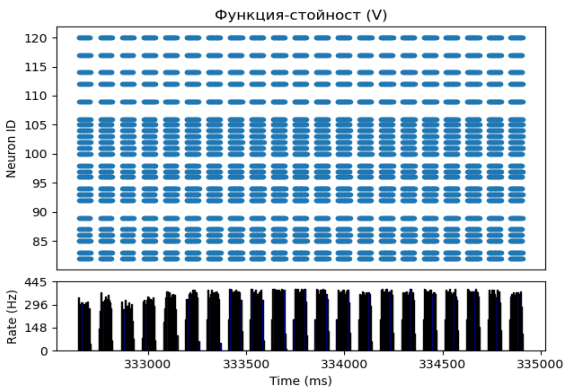
Така скалиран тока става между 1.005 и 171.82 пико ампера. Допълнително може да се регулира приноса на всеки един токов генератор с теглото на връзката от INPUT към STATE, която към момента е w=20.

Тук е редно да се спомене, че не бихме могли да се справим с отрицателна награда без съществено да се промени постановката, тъй като обемният трансмитер на допамин работи на базата на генерирани спайкове, които винаги са положително число.

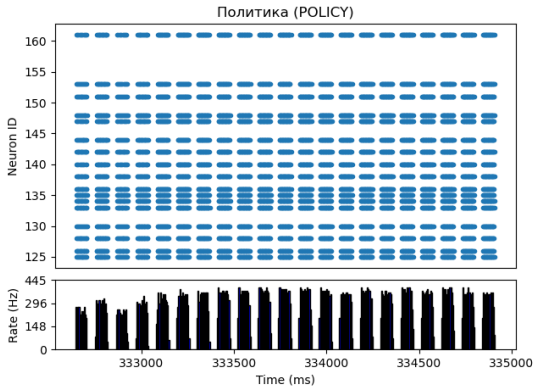
Фигурата по-долу показва симулация със 100 епизода. Като резултат има получената награда за всеки епизод, която показва колко дълго е балансирано рамото преди да падне.



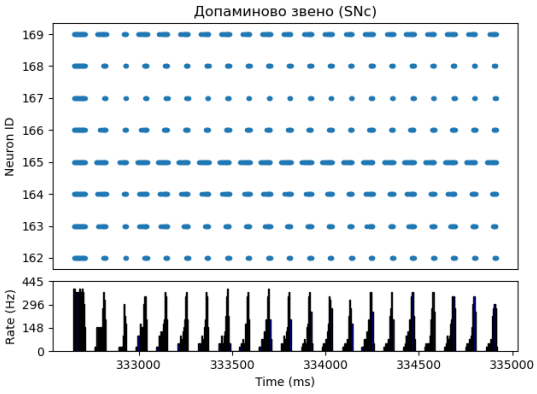
(А)



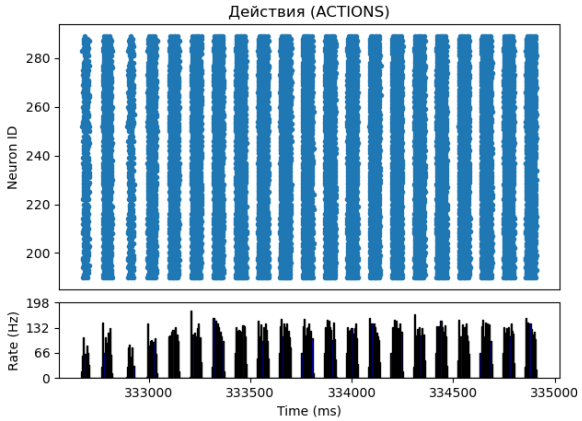
(Б)



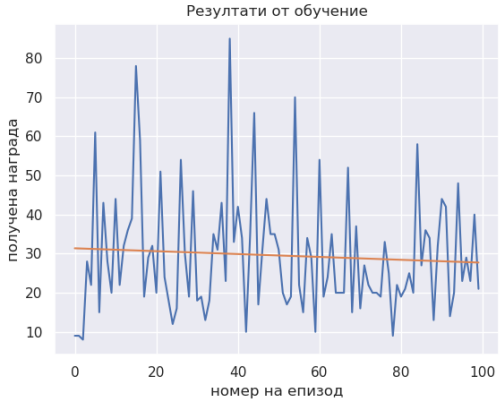
(В)



(Г)



(Д)



(Е)

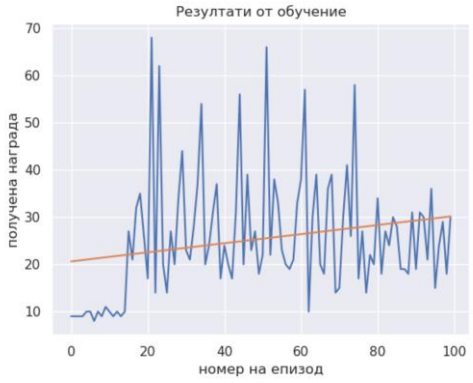
Фиг. 6.2.1.3 Резултати от обучение, вариант 1 за 100 епизода, А,Б,В,Г,Д – Спайкове и хистограма на различни невронни групи за последният епизод, Е – резултат от обучение и получени точки

Вижда се, че действително има епизоди, които са успели да балансират рамото над 80 стъпки (Фиг. 6.2.1.3,Е). За да се прецени дали е случайност като отправна точка се разглежда резултата при случайно избиране на действие [11].

Фиг. 6.2.1.4 Отправна точка (baseline) със случайна политика на агента за 1000 сепизода, копирано от [11].

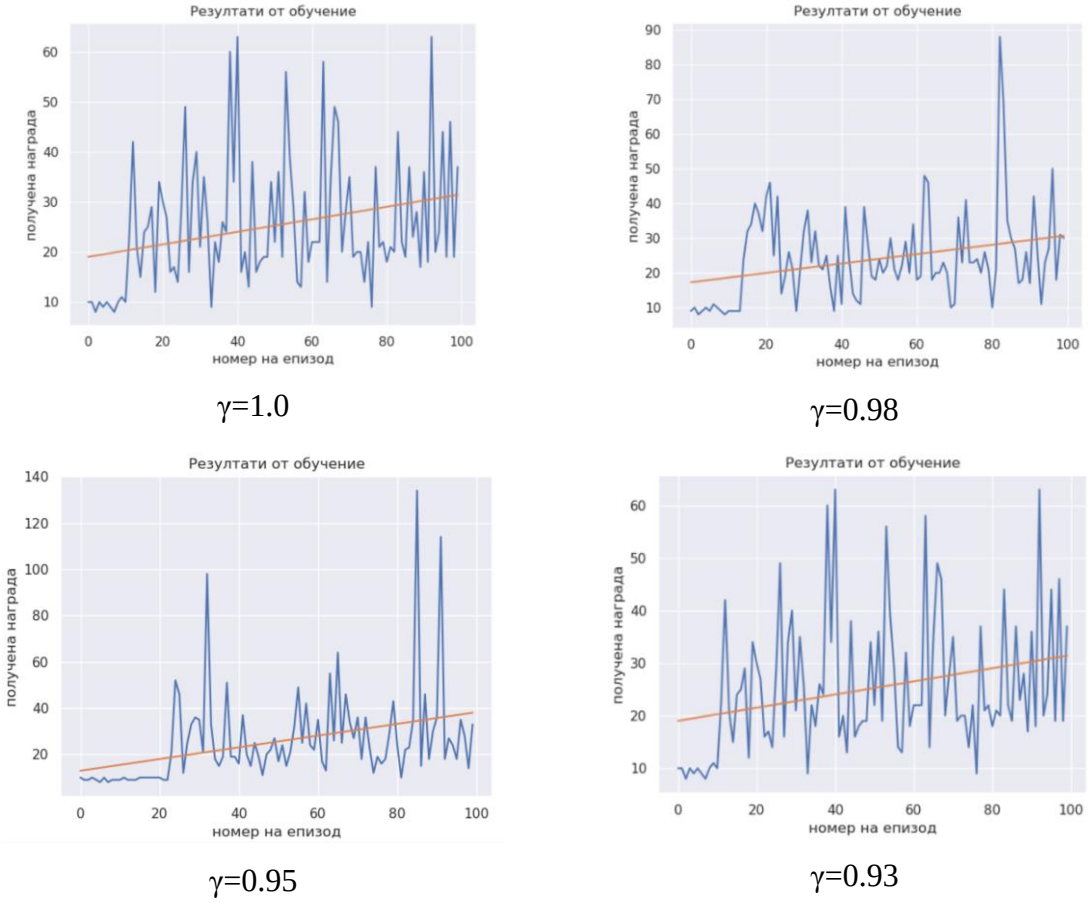
Средната стойност на кумулативната награда от отправната точка (baseline) е 22, докато във вариант 1, разгледан тук е около 30. Това означава, че този вариант не може да обучи агент да реши напълно задачата, а можем да считаме, че е малко по-добър от агент с хаотична политика.

При следващия експеримент функцията на наградата е променена на $\max(10.\cos(17.\,\text{state}[2]), 0)$. Такава функция дава награда в интервала от 0 до 10.0 и зависи изцяло от ъгъла на балансираното рамо.



Фиг. 6.2.1.5 Резултати от обучение при преправена функция на наградата, вариант 1 за 100 епизода

Колкото по-изправено е балансираното рамо, толкова е по-голяма наградата. Резултатът не е много по-различен от преди и можем да считаме, че няма подобрение. Промяна на други хиперпараметри, като например γ =0.93;0.95; 0.98;1.0 също не дава подобрение.



Фиг. 6.2.1.6 Резултати от обучение на вариант 1 при промяна на γ

6.2.2 Вариант 2 за решение на CartPole

Текущата мрежа и начин на свързване от вариант 1 не могат да бъдат подобрени с изброените методи в горната подточка, поради което се налага да се помисли за друг начин на свързване и подаване на входния сигнал. За целта е разработен втори скрипт, спазващ принципите на свързване и разграничаване на актьор-критика, и също зависещ от темпоралната грешка TD(0).

Скриптът от вариант 2 е направен 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. “-o” – избор на изходна директория за резултати
2. “-c” – избор на изтриване на изходната директория, “true” или “false”
3. “-n” – максимален брой епизоди
4. “-v” – отпечатване на подробни съобщения в терминала.

Пускането от операционна система Линукс става по следния начин:

```
python cartpole-actor-critic nest.py -n 250 -v
```

Таблица. 6.2.2.1 Пускане на скрипта за обучение от папка “variant2” за 250 епизода с подробно отпечатване на логовете (verbose)

Обучението завършва успешно и резултатът се отпечатва на терминала.

```
Episode 249 finished after 337 timesteps and reward 338.0
Mean score:  242.3
Save scores
====== all_states === all_actions ===
 source   target   synapse model   weight   delay
-------- -------- --------------- -------- -------
      1       65    dopa_synapse    1000.   1.000
      1       94    dopa_synapse    1000.   1.000
..................
```

Таблица 6.2.2.2 Край на обучението на агента за вариант 2

Като резултат има и картинки в директорията “scripts/variant2/outputs”.

Компонентът критика ще научи функцията V(s), а компонентът за актьор ще съдържа готовото решение, кодирано в синапсите на връзките. Тъй като активността на невроните се моделира със spike timing, се налага преминаването от спайкове към числени стойности на изхода на всяка група неврони.

Аналогично на вариант 1 се налага преобразуване на входния сигнал от обръжаващата средата до такъв сигнал, който е подходящ да захрани подобна симулационна невробиологична мрежа. Векторът на състоянието се състои от 4 стойности и се преобразува до вектор от 8 стойности както следва:

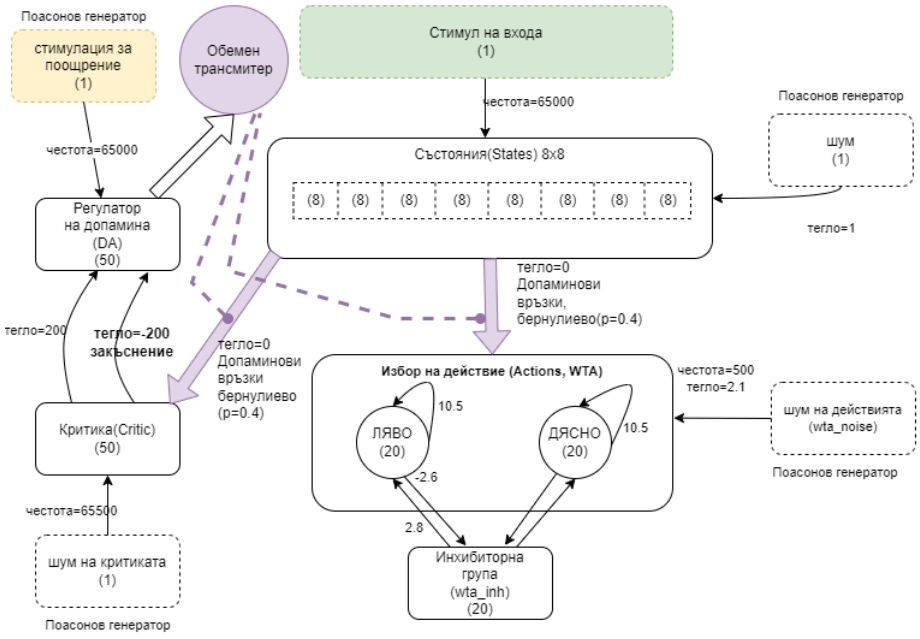
индекс	Вид наблюдение	Допустими ст-ти	Отрязване на стойностите (clipping)	Скалиране и разделяне на 2 стойности	Допълнително преобразуване
0	Позиция на количката	[-4.8; +4.8]	[-1.5; +1.5]	[0.2;1.5] [0.2;1.5]	$1.5(e^x - 1),$
1	Скорост на количката	[-∞;+∞]	[-1.5; +1.5]	[0.2;1.5] [0.2;1.5]	$1.5(e^x - 1)$
2	Ъгъл на балансираното рамо	[-0.418; +0.418]	-0.13;+0.13	[0.2;1.5] [0.2;1.5]	$1.5(e^x - 1)$

3	Ъглова скорост на балансираното рамо	[-∞;+∞]	-1.5; +1.5	[0.2;1.5] [0.2;1.5]	$1.5(e^x - 1)$

Таблица 6.2.2.3 Входни параметри от средата на CartPole и преобразуване до размерност 8

За всяка от тези 8 стойности на вече преобразувания входен вектор се създава отделна група от по 8 неврона, разположени таблично и се нарича „States“. Тази група представлява възможните състояния на средата и се състои от 8*8=64 неврона. При преместване на агента на различна позиция се активират по-малко или повече определените групи неврони отговаряща за всяка една компонента на 8-размерния вектор. Активацията става посредством генератор на поасонов шум с определена честота, наречен „стимул“. Така на всяка стъпка от всеки епизод, всяка компонента от „States” бива захранена с различно тегло от връзката си със стимула.

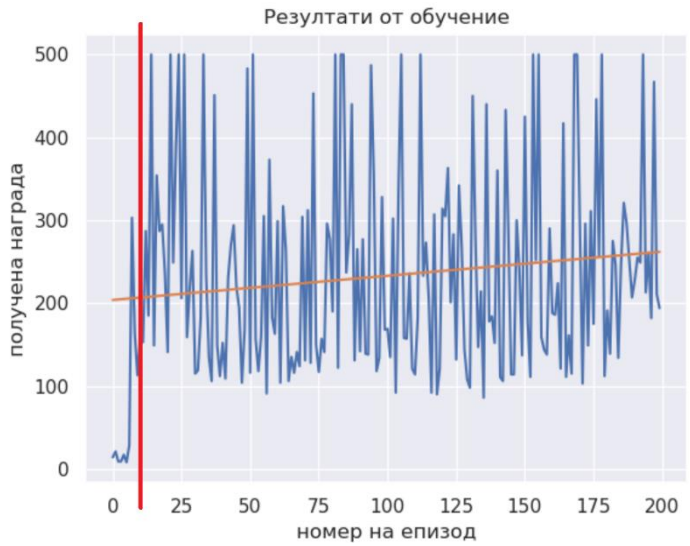
„States“ са свързани към WTA схема, с 2 възможни състояния като свързването е бернулиево по двойки с вероятност 0.4 (в NEST – „pairwise_bernoulli“, p=0.4). Всяка група от WTA се състои от 20 неврона и отговаря съответно на действията на агента, наречени „Actions”, от 0 до 1 включително, а именно: наляво-0, надясно-1. Връзките между „States“ и „Actions“ са с пластични допаминови синапси, първоначално с тегла 0.0, Схема на свързване е дадена на следващата фигура.



Фиг.6.2.2.1 Схема на свързване на невронните групи за вариант 2

„States“ са свързани с друга невронна група, наречена „Critic“ от 50 неврона също с допаминови връзки. Тази група представя „v*“ функцията от уравнението на Белман. „Critic“ е свързана с друга група от 50 неврона, отговарящи за нивото на допамина, условно наречена „DA“. Наградата от средата се формира като сигнал от поасонов шумогенератор с определена честота пропорционална на наградата. Този вход е наречен „Reward Stimulus“. В синаптичните връзки между „States“ и „Actions“ е заложено решението π*, защото в процеса на обучение на „Critic“, успешните ходове на агента са предпоставка за усилваща връзка от дадено състояние на средата, към определена посока, например „Ляво“. „Reward Stimulus“ постъпва през групата „DA“. Така „DA“ отчита очакваната награда, а не абсолютната награда, което съответства на TD грешката $\delta_{t-1} = R_t + \gamma V(S_t) - V(S_{t-1})$ от (5.7). Допаминовите синапси в симулатора NEST получават нивото на допамина, генерирано от DA посредством устройството volume transmitter, описан в 5.6. Тъй като средата CartPole дава награда на всеки ход от всеки експеримент, има обучение винаги, когато наградата се различава от очакваната награда. Тук е редно да се спомене, 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 брой спайкове).

След пускане на скрипт вариант 2 наблюдаваме следните резултати.



Фиг. 6.2.2.2 Резултати от обучение на вариант 2 при γ=0.95 за 200 епизода

Вариантът изглежда дава очакваните резултати за решение още на първите 10 стъпки. На първо място огромно значение имат входните променливи и тяхното мащабиране. В този случай са избрани стойностите 0.2-1.5 както е видно от Таблица 6.2.2.3 колона 5. В този интервал, получен след множество експерименти, стойностите дават оптимален резултат и могат да възбудят невронните групи на състоянията, така, че да са балансирани спайковете към следващата свързана невронна група.

Втората важна промяна е скоростта на положителното и отрицателното обучение. Под положително обучение се има предвид параметъра на допаминовите връзки A_plus=0.004. Под отрицателно обучение се има предвид параметъра A_minus=0.0, т.е. няма отрицателно обучение. Отрицателното обучение е премахнато. То отслабва тези невронни групи с допаминови връзки, за които $t_{pre} > t_{post}$ (5.6.2). Но в конкретния случай невроните от WTA са единствено възбуждащи се от States. Такива невронни двойки, за които времето на спайк от групата State изпреварва времето за спайк на неврон от групата Actions е резултат на страничен ефект (например редуващи се спайкове) и не е желан ефект. Поради тази причина отрицателното обучение внася само шумове, макар и да регулира по някакъв начин теглата на допаминовите връзки да не нарастват прекалено много.

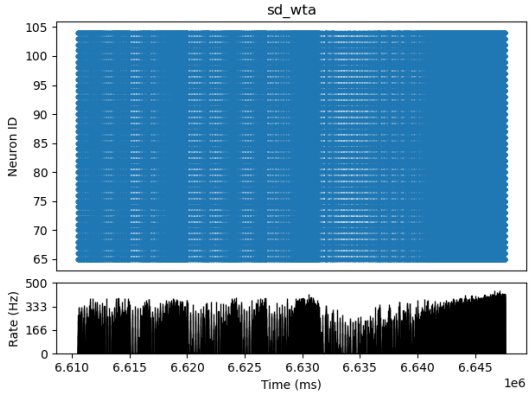
Следващата важна промяна е въвеждането на висок коефициент на скалиране на наградата REWARD_SCALING = 6.25. Целта е в началните епизоди агента да бъде научен да държи баланс, защото когато започне да пада балансираното рамо, средата пак дава награда и агента ще заучи падането като нещо полезно, което е невярно.

Следващата важна промяна е въвеждане на дълъг покой между епизодите и раздалечаване във времето на спайковете между отделните епизоди, защото допаминовите връзки могат да се влияят и от по-далечни спайкове във времето в зависимост от параметър за следа (tau_c – eligibility trace). Това налага въвеждане на следната промяна:

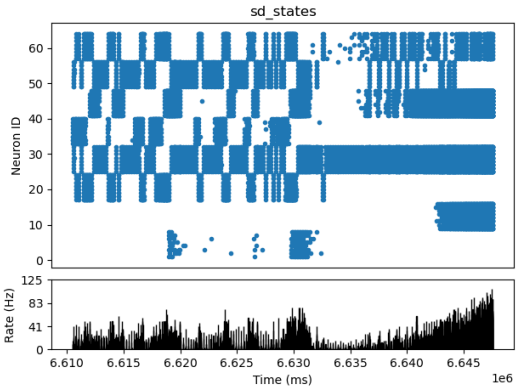
```
        if done:
            # Long rest time to separate episodes
            nest.Simulate(10 * REST_TIME)
            current_time += 10 * REST_TIME
```

Таблица 6.2.2.4 Въвеждане на разделяне на епизодите във времето с многократно време за почивка.

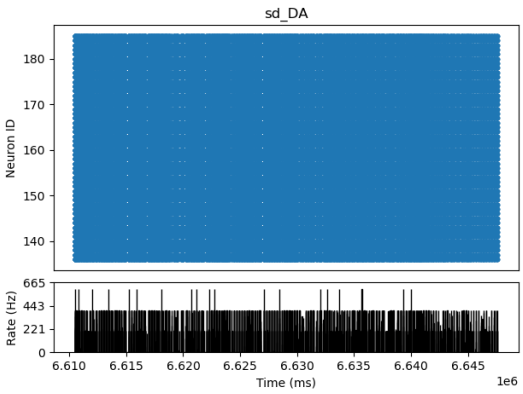
По този начин става невъзможно спайк от преден епизод да влияе на спайк от следващ епизод. Фигурата по-долу показва спайк диаграмите на вариант 2 за последният епизод, заедно с кумулативната награда във времето за 250 епизода.



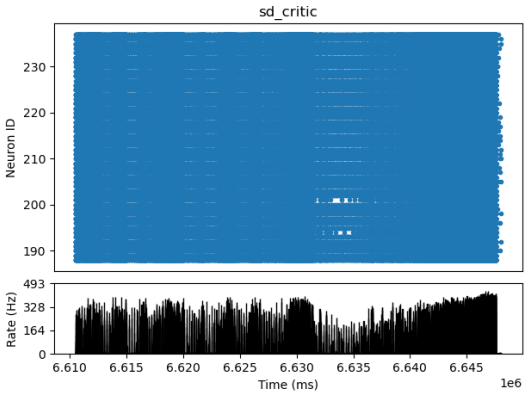
(А)



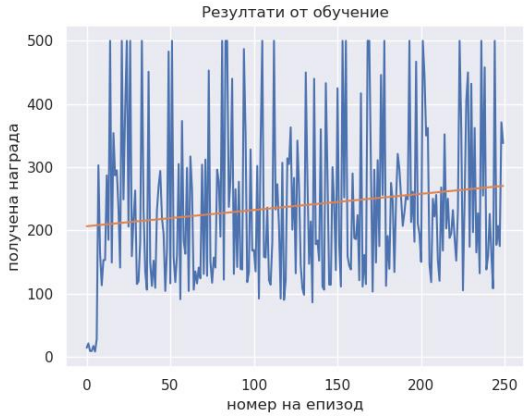
(Б)



(В)



(Г)



(Д)

Фиг. 6.2.2.3 Резултати от обучение, вариант 2 за 250 епизода, А,Б,В,Г – Спайкове и хистограма на различни невронни групи за последния епизод, Д – резултат от обучение

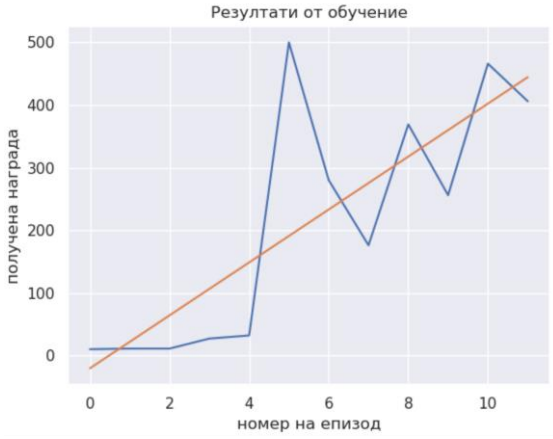
Вижда се, че още в ранните епизоди (Фиг.6.2.2.4-Д) средните точки от епизод надминават 195 точки. Можем да считаме, че задачата е решена с тези подобрения във вариант 2. Решението надминава всички решения както с класически невронни мрежи , така и с SNN, изброени в обзорната част в началото, освен решението на Jian Xu [11] (което от своя страна не е машинно самообучение, а човешко наблюдение).

Тренирането на подобна задача на DELL XPS, Intel I9 32GB RAM с виртуална машина VirtualBox и Ubuntu отнема около 1 час. Невронните групи са намалени до минимален брой, за да се ускори симулацията.

Трябва да се има предвид, че всяко стартиране на скрипта няма да обучи агента по същия начин, поради наличието на случаен момент при свързването на мрежата. Случайността

идва от бернулиевите връзки по двойки с вероятност 0.4, както и от поасоновите шумогенератори и и средата, която стартира по случаен начин всеки път.

За пълнота на експеримента е увеличен параметърът γ до γ =0.999 и резултатът е сравним с резултатите от γ=0.95 но само до достигане на средно 195 точки от последните 100 епизода. Логиката на увеличаване на този параметър е, че наградата в бъдеще е толкова ценна колкото наградата сега.



Фиг. 6.2.2.3 Резултати от обучение на вариант 2 при γ=0.999 до достигане на решение при епизод 11

6.3 Параметри на постановката и анализ на резултатите

За допаминовите синапси са зададени следните параметри:

```
tau_c = 50.0  # Time constant of eligibility trace
tau_n = 20.0  # Time constant of dopaminergic trace
tau_plus = 20.

# Connect states to actions
nest.CopyModel('stdp_dopamine_synapse', 'dopa_synapse', {
    'vt': vol_trans.get('global_id'), 'A_plus': 0.004, 'A_minus': 0.0,
"tau_plus": tau_plus,
    'Wmin': -1000., 'Wmax': 1000., 'b': 0., 'tau_n': tau_n, 'tau_c':
tau_c})
```

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

На скоростта на обучение може да се повлияе и като се скалира поощрението от средата. Към момента това става с емпиричната формула:

```
REWARD_SCALING = 6.25...
nest.SetStatus(nest.GetConnections(reward_stimulus,          DA_neurons),
{'weight': float(reward) * REWARD_SCALING})
```

Таблица 6.3.2 Скалиране на наградата и влияние на скоростта на обучение

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

Синапсите имат механизъм за следа (eligibility trace) в обучението. Тя може да спомогне да проследи през какви стъпки е минал агента стигайки до конкретно решение. Това се регулира с константата „tau_c“ в модела „stdp_dopamine_synapse“. Този параметър е избран да е доста по-малък в сравнението със стойността му по подразбиране (1000.0) за да не се смесва обучение при падащо рамо с преден момент, когато рамото е било балансирано.

Друг полезен механизъм е регулиране на базовата концентрация на допамин „b” от таблица 5.6.2. 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. Този параметър няма как да се използва в тази задача.

Реализирането на ε-greedy политика π от алгоритъма става посредством поасонов шумогенератор „wta_noise“ с честота „WTA_NOISE_RATE“ показан на Фиг.6.2.2.1 долу вдясно. Давайки по-голяма честота, се засилва силата на случайния сигнал за вземането на решения в „Actions”. Давайки по-малка честота ще има по-малко шум и съответно няма да се изпробват нови комбинации от действия а ще се следва само наученото, което може и да не е оптимално. Този шум действа само в началото на обучението, когато теглата от 0 нарастват, но все още са с малки стойности. При нарастване на теглото на допаминовите връзки към стойности близки до Wmax=1000.0 вече генераторът на шум не влияе особено на сигнала.

Как влияе времеделенето при експерименталната част? Времеделенето при вариант 1 и вариант 2 са еднакви като значение но с различни стойности. 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 на повечето синапси. Ако е много малко, тогава обучението ще е много дълго. То допринася към скоростта на обучение.

Хиперпараметрите на обучението на подобна система са много на брой, над 20 в случая и изпробването на всички комбинации не е възможно, а и не е цел на настоящия труд. Достатъчно беше да се покаже поне един работещ вариант и през какви стъпки, със съответните обяснения, се преминава.

7. Заключение

Проведени бяха по няколко експеримента върху два варианта на мрежа от тип Spiking Neural Network (SNN) и реинфорсмънт обучение. Бяха показани редица промени по тези варианти с рационални обяснения. Решение бе достигнато с втория вариант, като такова решение превъзхожда други решения с конвенционални невронни мрежи.

7.1 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.

Друг вид подобрение е да се избегне експерименталното нагаждане на стойностите от входния сигнал и да се направи, така че мрежата сама да скалира входните токове към годни за употреба стойности в подходящи горни и долни граници, захранващи мрежата с балансирани спайкове по различните звена.

Друг вариант за подобрение е да се експериментира с хиперпараметрите от вариант 1 и вариант 2 и търсене на подобрение.

Може да се експереминентира и с различна по структура мрежа с включване на Go и NoGo гейтове от базалните ганглии, които умишлено съм пропуснал поради високата сложност и липса на достатъчно източници и литература с работещи решения.

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://compcogneuro.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] CartPole, OpenGym, https://www.gymlibrary.dev/environments/classic_control/cart_pole/

[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.

[8] Doya K (2000), Complementary roles of basal ganglia and cerebellum in learning and motor control, Current Opinion in Neurobiology 2000, 10:732–739, Elsevier Science Ltd

[9] Koprinkova-Hristova P,(2020), Brain-Inspired Spike Timing Model of Dynamic Visual Information Perception and Decision Making with STDP and Reinforcement Learning, Springer Nature Switzerland AG

[10] Florian R (2007), Reinforcement Learning Through Modulation of Spike-Timing-Dependent Synaptic Plasticity

[11] Xu J, How to Beat the CartPole Game in 5 Lines, https://towardsdatascience.com/how-to-beat-the-cartpole-game-in-5-lines-5ab4e738c93f

[12] Surma G, Cartpole - Introduction to Reinforcement Learning (DQN - Deep Q-Learning) https://gsurma.medium.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288#c876

[13] Kale S, Solving CartPole-V1, https://medium.com/@siddharthkale/solving-cartpole-v1-4be909b7c2c6

[14] Barto A, Sutton R, (1983), Neuronlike adaptive elements that can solve difficult learning control problems, IEEE, http://incompleteideas.net/papers/barto-sutton-anderson-83.pdf

[15] Kenji D, (2000), Reinforcement Learning In Continuous Time and Space, ATR Human Information Processing Research Laboratories, Japan

[16] Qiu H, Garratt M, Howard D, Anavatti S,(2019) Evolving Spiking Neural Networks for Nonlinear Control Problems, arXiv:1903.01180v1

[17] Mahmoud Akl, Yulia Sandamirskaya, Deniz Ergene, Florian Walter, and Alois Knoll. 2022. Fine-tuning Deep Reinforcement Learning Policies with r-STDP

[18] Akl M, Ergene D, Walter F and Knoll A (2023) Toward robust and scalable deep spiking reinforcement learning. Front. Neurorobot. 16:1075647. doi: 10.3389/fnbot.2022.1075647

[19] Bing Z., Meschede C., Huang K., Chen G., Rohrbein F., Akl M. Knoll A., (2019), End to End Learning of Spiking Neural Network based on R-STDP for a Lane Keeping Vehicle, ResearchGate

[20] Shim M., Li P., (2017) Biologically inspired reinforcement learning for mobile robot collision avoidance, IEEE

[21] Zhao F, Zeng Y and Xu B (2018) A Brain-Inspired Decision-Making Spiking Neural Network and Its Application in Unmanned Aerial Vehicle. Front. Neurorobot. 12:56.doi: 10.3389/fnbot.2018.00056

[22] Schultz, W., P. Dayan, and P. R. Montague. 1997. “A Neural Substrate of Prediction and Reward.” Science 275 (5306): 1593–9. http://www.ncbi.nlm.nih.gov/pubmed/9054347.

[23] Gazzaniga M, Ivry R, Mangun G, (2002), Cognitive Neuroscience: The Biology of the Mind, W.W. Norton & Company, New York.

[24] Rescorla R, Wagner A, (1972), A Theory of Pavlovian Conditioning: Variation in the Effectiveness of Reinforcement and Non-Reinforcement. In Classical Conditioning II: Theory and Research, edited by A. H. Black and W. F. Prokasy, 64–99. New York: Appleton-Century-Crofts.

[25] Hebb D, (1949), The Organization of Behavior, Wiley: New York

[26] Markram H, Lubke J, Frotscher M, Sakmann B (1997) Regulation of synaptic efficacy by coincidence of postysnaptic AP and EPSP. Science 275: 213–215

[27] Bi G, Poo M (1998) Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. J Neurosci 18: 10464–10472.

Приложения

1. Сорс код (Source code)

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