**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«Национальный исследовательский университет** **«Высшая школа экономики»**

**Факультет компьютерных наук**

**Основная образовательная программа**

**Прикладная математика и информатика**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**на тему**

**Использование нейросетевой архитектуры с пространственно-временными связями для построения пути в лабиринте**

**Выполнил студент группы 131 4 курса,**

**Филин Дмитрий Андреевич**

**Научный руководитель:**

**Панов Александр Игоревич**

**Москва 2016**

Оглавление

[Аннотация 3](#_Toc482603768)

[Abstract 4](#_Toc482603769)

[Список ключевых слов 5](#_Toc482603770)

[Введение 6](#_Toc482603771)

[1 Нейросетевая архитектура 10](#_Toc482603772)

[Концепция Обучения с Подкреплением 10](#_Toc482603773)

[Temporal Hebbian Self-Organizing Map 11](#_Toc482603774)

[Формальная постановка задачи 14](#_Toc482603775)

[2 Эксперименты 16](#_Toc482603776)

[Лабиринты 16x16 16](#_Toc482603777)

[Лабиринты 64x64 18](#_Toc482603778)

[Процесс обучения агента на различных лабиринтах 19](#_Toc482603779)

[Анализ результатов и влияние параметров 20](#_Toc482603780)

[Заключение 23](#_Toc482603781)

[Библиографический список 24](#_Toc482603782)

# Аннотация

В работе представлена модель обучения с подкреплением, использующая модифицированную нейросетевую архитектуру с пространственно-временными связями, также известную как Temporal Hebbian Self-Organizing Map (THSOM). Мы утверждаем, что такие классы сред как лабиринты могут быть подвержены кластеризации состояний, что позволяет использовать сравнительно небольшие нейронные сети, в следствие чего время обучения и вычислительных затрат может быть значимо снижено. Был проведен ряд экспериментов с данной моделью на задаче нахождения путей в лабиринтах различной сложности. В большинстве случаев модель показала устойчивое обучение, строя путь, близкий к оптимальному. В данной работе подробно рассмотрены как и какие параметры модели влияют на скорость и сходимость процесса обучения агента, а также на время прохождения лабиринта. В Главе 3 также приведена критическая составляющая модели, рассмотрены некоторые нестандартные для агента ситуации и даны рекомендации по их преодолению. В общем и целом по полученным результатам можно судить о том, что рассмотренный в этой работе эксперимент прошел успешно и нам удалось разработать применимый в реальных условиях алгоритм поведения агента.

Abstract

The paper reviews a Reinforcement Learning model, which uses a modified neural network architecture with spatio-temporal weights also well-known as Temporal Hebbian Self-Organizing Map (THSOM). We claim that the states clustering can be successfully applied to such kind of environment as mazes what leads to usage of relatively small neural networks, as a result of which the training time and computational costs can be significantly reduced. We conducted many experiments where we were trying the model with the solving labyrinths of different complexity. In the most cases the model demonstrated sustainable learning, building close to optimal path. In this work we are considering in details how different parameters of the algorithm affect the speed and convergence of the agent's learning process, as well as the time of maze solving. At the chapter 3 we provide a critics of the model, consider some non-trivial for an agent cases and give advice on how to overcome them. Overall, based on the obtained results, it can be judged that the experiment considered in this work was successful and we succeeded in developing an algorithm of the agent's behavior that is applicable in real conditions.

# Список ключевых слов

Neural network, Machine Learning, Self-Organizing Map, Robotics, Learning with Reinforcement.

# ВВЕДЕНИЕ

В настоящее время задача повышения уровня автономности робототехнических систем за счет интеграции в их системы управления расширенных подсистем представления знании и обучения, становится важным направлением в искусственном интеллекте и в когнитивных архитектурах [6-8]. Глубокое обучение с подкреплением продемонстрировало впечатляющие результаты на так называемых «сырых данных», т.е. необработанных изображениях, получаемых с сенсоров обучающейся системы [1]. В настоящее время системы, разрабатываемые для простых экспериментов в игровых симуляционных средах, начинают применяться в реальных робототехнических задачах [3,4]. Идея использования в качестве обучающих данных информацию, получаемую с таких сенсоров, как зрительные и звуковые, дает возможность построить представление среды, с которой взаимодействует агент. Интеграция нейронных сетей с традиционным Q-обучением позволяет сопоставлять наблюдаемые состояния среды с получаемым от нее вознаграждением за те или иные действия. Автоматическая генерация признаков для более качественного описания состояний среды с помощью нейронных сетей позволяет применять Q-обучение на реальных «сырых данных», получаемых с сенсоров [5].

В настоящей работе представлена модель, комбинирующая одну из нейросетевых архитектур, базирующуюся на THSOM [2], с Q-обучением в задаче обучения построению пути на местности. Мотивацией послужила идея о том, что в целом все лабиринты, точнее их изображения, состоят из некоторых паттернов (см. Рис. 1), которые можно разделить на несколько категорий. Если бы агент понимал в каком состоянии среды он находится по наблюдаемому паттерну, то он мог бы определить наиболее подходящее для этого состояния действие. Так, например, в случае, изображенном на Рис. 2, агенту доступны действия «движение влево» и «движение вниз».

Главной целью данной работы является разработка алгоритма передвижения агента в структурированной (то есть заданной по определенным правилам, с определенным набором условий) среде, превосходством которого бы являлась низкая вычислительная сложность по сравнению с аналогичными средствами решения данной задачи. Так, самым распространенным способом нахождения пути в лабиринте является представления его в виде связанного графа и запуска алгоритмов, основанных на поиске в ширину, таких как алгоритм Ли или алгоритм A\* (A star). Однако, во-первых, асимптотика данных алгоритмов составляет , что при больших размерах лабиринтов является существенной величиной (так как само - количество вершин в графе составляет , где – размеры лабиринта; а число ребер приблизительно составляет в зависимости от конкретной структуры среды). Во-вторых при использовании подобных алгоритмов, предполагается, что мы владеем полной информацией о графе, тогда как в рассматриваемой нами задаче агент видит лишь часть лабиринта и не обладает знанием о всей его структуре. В-третьих, такой подход даст агенту лишь решение на конкретной карте, однако мы ставим задачу обучить агента так, чтобы на каждом последующем лабиринте время прохождения уменьшалось. Альтернативным решением является применение глубинных нейронных сетей, которые применяются в прохождении игр в эмуляционной среде Atari[1]. Преимуществом нашей модели перед данным подходом является отсутствие многослойности сети и как следствие высокая скорость вычислений и обучения. Далее будут приведены результаты экспериментов и данные о времени работы агента, где будет продемонстрировано, что алгоритм выполняется за приемлемое время на среднем процессоре, тогда как использование технологии Deep Reinforcement Learning требует наличие дорогостоящих и энергозатратных GPU.

Основными задачами которые ставились в данной работе являются:

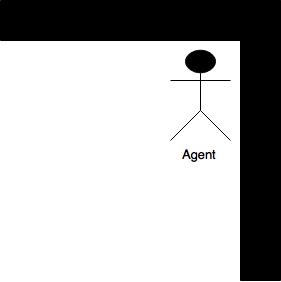
1. Проектирование модели, совмещающей обучение с подкреплением и нейросетевую архитектуру THSOM
2. Программная реализация среды и алгоритма
3. Анализ результатов, а именно: тестирование модели на лабиринтах различных размеров, содержащих различное количество препятствий; оценка влияния параметров модели на скорость прохождения агентом лабиринтов; оценка степени обучаемости агента.

Область применения данной модели составляет большой класс задач робототехники и видеоигр. Так как лабиринты являются классической средой передвижения объектов в таких продуктах как, например, Tanks 2D и Doom, внедрение предложенной модели в виде элементарного искусственного интеллекта ботов, требующего низких вычислительных затрат (говоря о таких играх мы ориентируемся на маломощные видео приставки) являлось бы удачным решением. Помимо видеоигр лабиринты также активно применяются в робототехнике, свидетельством чего являются так называемые Maze Challenges (соревнования по прохождению лабиринтов роботами на скорость).

В Главе 1 дается подробное описание теоретической составляющей модели. Объясняется архитектура THSOM, а также интеграция данной нейросети с RL-подходом. Также мы приводим описание непосредственно самой модели, ее параметров и демонстрируем разработанный алгоритм. В Главе 2 приводятся эксперименты и полученные результаты, делаются выводы об успешности применимости модели к тем или иным средам.



**Рис. 1. Примеры паттернов лабиринта**



**Рис. 2. Агент в одном из состояний среды**

## 1 НЕЙРОСЕТЕВАЯ АРХИТЕКТУРА

## 1.1 Концепция Обучения с Подкреплением

Обучение с подкреплением (Рис. 1.1) представляет один из видов машинного обучения, где роль учителя играет среда, с которой взаимодействует агент. В ходе эксперимента агент совершает определенный для него набор действий, каждое из которых сопровождается определенным откликом среды. Задача агента понять, как нужно действовать в тех или иных состояниях среды, чтобы максимизировать долгосрочный выигрыш. Рассматривая обучение с подкреплением с математической точки зрения, можно представить данный подход в виде Марковского процесса с конечным числом состояний, вероятности перехода между которыми определяют то, насколько агенту выгодно выполнять данное действие в текущем состоянии. Стоит отметить, что обучение с подкреплением не решает задачу нахождения глобального экстремума, то есть в большинстве случаев не дает оптимального решения задачи. Агент в каждом моменте находит лишь локальный экстремум, получая на выходе некий субоптимальный алгоритм действий. Концепция обучения с подкреплением заключается в том, чтобы найти компромисс между уже полученным опытом и исследованием агентом новых возможностей. Так как данный подход является естественным процессом при обучении человека, он активно применяется в робототехнике и смежных областях.



**Рис. 1.1. Концепция обучения с подкреплением**

## 1.1 Temporal Hebbian Self-Organizing Map

Как уже было сказано, в основе модели лежит нейронная сеть THSOM (Рис. 1.2), которая обладает двумя видами межнейронных связей – пространственными и временными. Основная идея заключается в том, что каждый входной вектор является аттрактором для нейронов, к которому часть из них итеративно стягивается в процессе обучения, таким образом образуются кластеры похожих паттернов. Данный процесс контролируется стандартным для алгоритмов обучения без учителя способом (например, алгоритм самоорганизующихся карт Кохонена). Радиус стяжения вокруг наиболее близкого к входному вектору нейрона (Best Matching Unit) рассчитывается как:

|  |  |
| --- | --- |
|  | (1.1) |

но в нашем эксперименте, он брался нулевым, чтобы стягивался только один нейрон. Так достигалось наименьшее количество необходимых состояний, при этом значительного влияния на работоспособность алгоритма это не оказало. Сила связей нейронов изменялась по формуле:

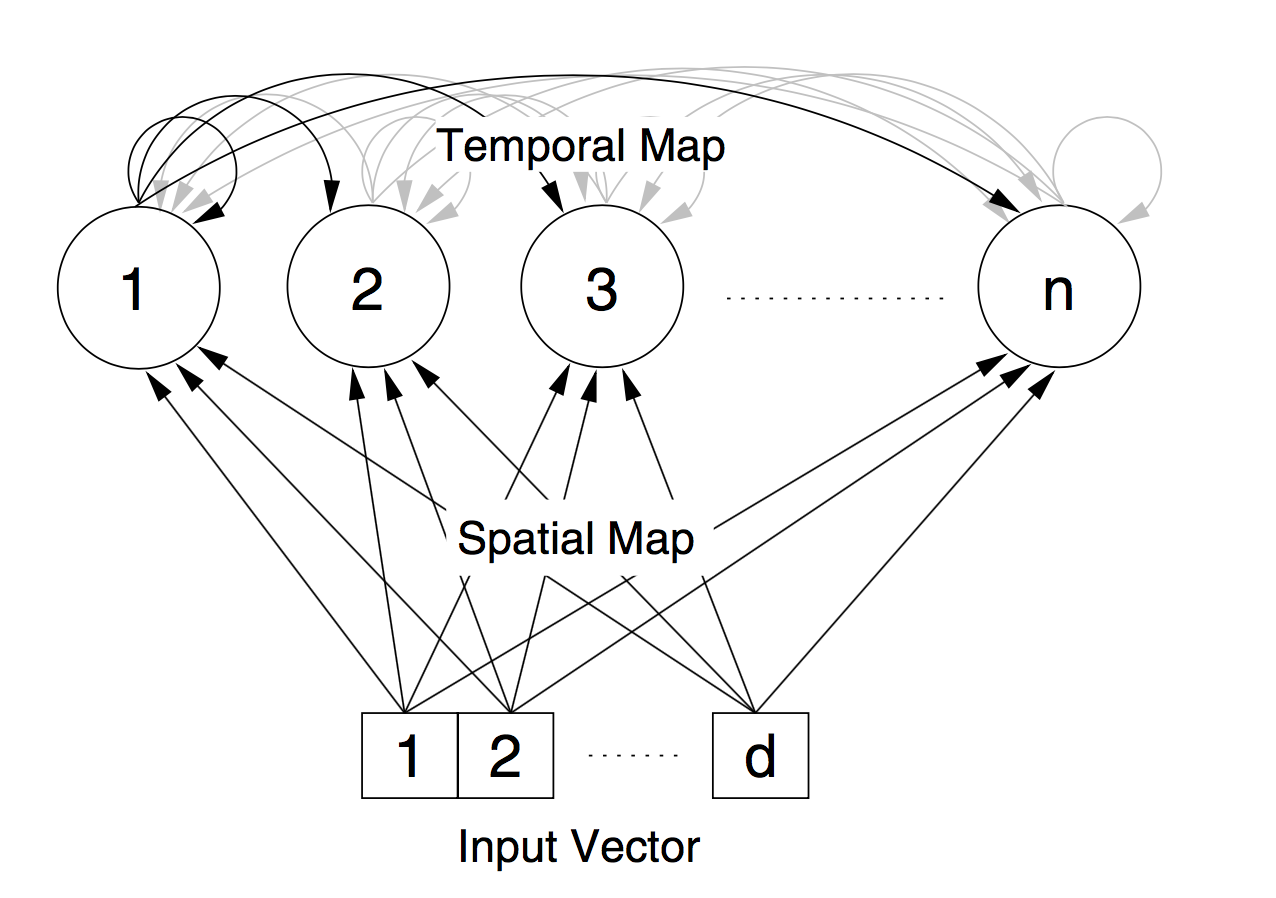
|  |  |
| --- | --- |
|  | (1.2) |

где нейрон, – входной вектор, расстояние между векторами.

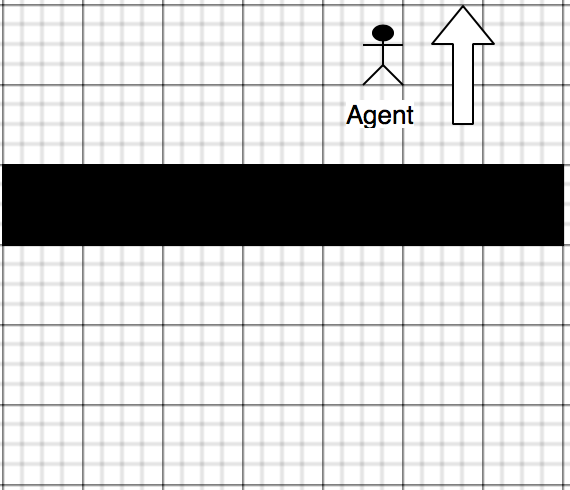
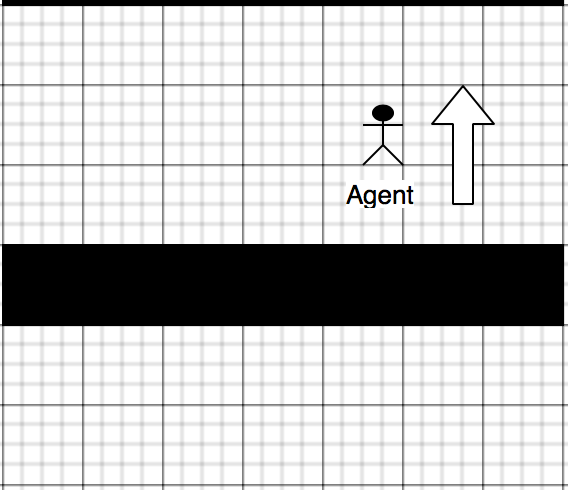
Отдельного внимания здесь заслуживает именно расчет расстояния, так как в нашей задаче мерой расстояния является схожесть паттернов лабиринта. Стандартные метрики не подходят, так как они основаны на разности соответствующих компонент вектора, тогда как в случае ниже паттерны (Рис. 1.3) можно считать одинаковыми, и им соответствуют одни и те же доступные действия агента. Однако на Рисунке 1.4. паттерны хоть и похожи внешне, но отличаются набором доступных для агента действий. То есть, если для агента не будет важна позиция стены относительно него самого, а только форма, то, научившись ходить вверх от горизонтального препятствия, он будет делать это всегда и для него не будет разницы с какой стена стороны. В связи с этим была введена метрика, которая учитывает, как разницу между формой паттернов, так и между их расположением в области видимости. Тогда расстояние считается как:

|  |  |
| --- | --- |
|  | (1.3) |

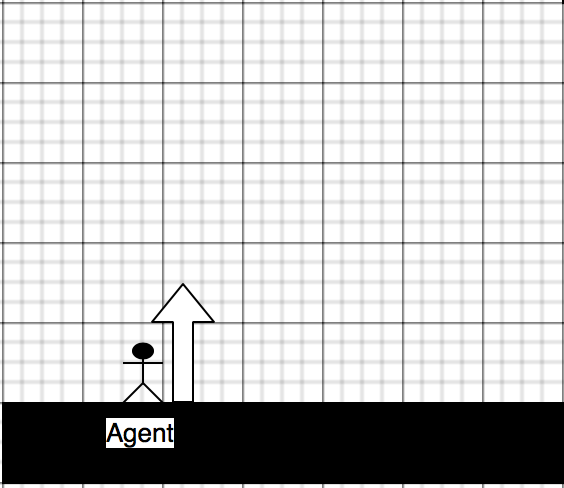
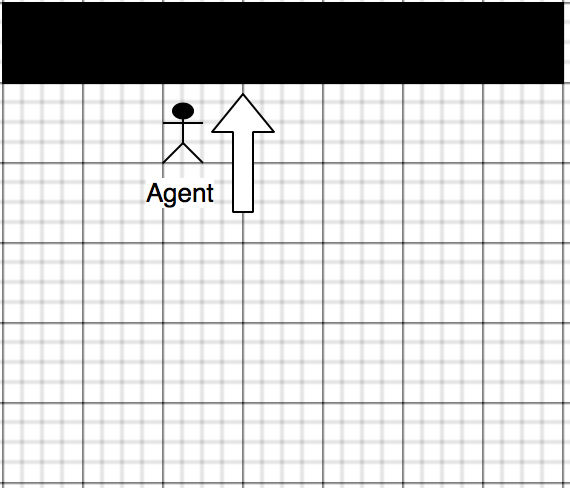
где– минимальная разница между паттернами при наложении их друг на друга (в единицах), – количество сдвигов для достижения этой разницы,



**Рис. 1.2. Архитектура THSOM**



**Рис. 1.3. Примеры похожих паттернов**



**Рис. 1.4. Паттерны имеют одинаковую структуру, однако в одном из случаев движение вверх является недопустимым**

Временная составляющая модели состоит из 4 связей между каждой парой нейронов, которые обуславливают распределение вероятностей в пространстве действий. То есть, чем сильнее связь, тем вероятнее действие, которое нужно выполнить для перехода от одного состояния к другому. Расчет временных весов происходит по формуле:

|  |  |
| --- | --- |
|  | (1.4) |

где – состояния, – действие, – фидбэк от среды. Он состоит из постоянной награды/наказания за каждый шаг + награды за приближение к цели. Последняя компонента добавлена для того, чтобы агент был мотивирован идти именно к финишу. Выбор действия в момент времени соответствует самой сильной исходящей связи из текущего активного нейрона:

|  |  |
| --- | --- |
|  | (1.5) |

Помимо выбора оптимальных действий в модели также используется стратегия которая на начальных этапах играет крайне важную роль. Когда процесс только начинается, и игрок еще ничего не знает о паттернах, он действует какое-то время абсолютно случайно, накапливая опыт. В общем случае вероятность случайного действия вычисляется как:

|  |  |
| --- | --- |
|  | (1.6) |

где – пороговая константа. Обычно берется небольшое значение .

Рассмотрим стохастическую среду в которой агенту доступен следующий набор действий:

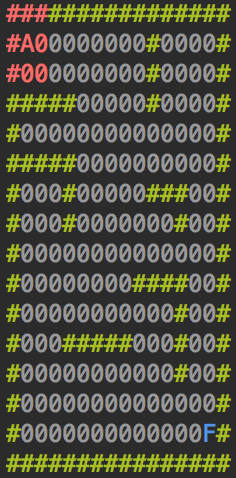
которые соответствуют следующим изменениям положения агента в пространстве :

Каждая ячейка пространства может быть в одном из трех соcтояний:

что соответствует пустой клетке, стенке или конечной точке. При попадании в каждое из этих состояний агенту назначается награда:

Также существует награда за приближение к цели , о чем будет сказано позже.

Задача – добраться до конечной цели за наименьшее число шагов. Пример состояния среды изображен на Рисунке 1.5.



**Рис. 1.5. Эмулированный лабиринт. Красным отмечена область, доступная для видимости агенту**

Алгоритм обучения агента:

1. Проинициализировать временные и пространственные веса малыми случайными значениями
2. Сгенерировать нейронов как рандомные вектора с равномерным распределением на отрезке размера (соответствующие размерам области видимости)
3. Задать константы , соответствующие очкам за действия агента
4. Установить порог -*greedy policy ,* а также динамику уменьшения данной величины – константу (см. Формулу 1.6)
5. for t = 1, T do
   1. Считать входной сигнал с сенсоров (получить текущий блокx)
   2. Определить как самый близкий ко входу нейрон
   3. Обновить пространственные веса (Формула 1.2)
   4. if t != 1 обновить временные веса (Формула 1.4)
   5. Запомнить текущее состояние как
   6. В соответствии с выбрать (Формула 1.6)
   7. Выполнить выбранное действие, запомнить
   8. Если не достигли финиша, перейти к шагу 2
6. end for

2 ЭКСПЕРИМЕНТЫ

В целях подробного анализа работоспособности алгоритма было решено провести эксперименты на лабиринтах двух различных размеров и содержащих препятствия различной сложности. Изначально будет показан процесс в рамках каждого класса лабиринтов, а затем будет приведен пример, когда агент последовательно передвигается в различных средах. Также будет продемонстрировано влияние ложных путей и тупиков на поведение агента.

## 2.1 Лабиринты 16x16

Запустим агента 20 раз на данном лабиринте, не содержащем тупиков.  Размер сети возьмем равным 10 нейронам. Зависимость шагов агента до финиша от номера итерации тестирования приведена на графиках ниже:

|  |  |
| --- | --- |
|  |  |
| **Рис. 2.1. Простой лабиринт с областью видимости 3x3** | |

|  |  |
| --- | --- |
|  |  |
| **Рис. 2.1. Лабиринт с тупиками c областью видимости 3x3** | |
|  |  |
| **Рис. 2.2. Лабиринт с тупиками c областью видимости 5x5** | |

Общее время работы алгоритма составило 8.83, 18.42 и 24.36 секунд соответственно.

На первом графике изображен простой лабиринт, не представляющий для агента трудностей, так как во все состояниях он способен однозначно определять действие, максимизирующее его выигрыш. Из графика видно устойчивое обучение агента, а также стоит отметить, что лучший результат в 25 шагов отличается всего на 3 от оптимального пути (оптимальный путь составляет 22 шага). Однако нельзя не сказать о таких ситуациях, где модель ведет себя наихудшим образом – когда агент попадает в тупик (см. Рис. 2.1). Причина такого поведения будет объяснена ниже, однако мы можем видеть, что количество шагов, требуемое для достижения финиша возрастает почти в 10 раз. Данную ситуацию можно улучшить, увеличив область обзора у агента, как показано на Рисунке 2.2. В данном случае из графика видно, что в среднем агенту требуется немного больше шагов, чем в первом случае, однако он перестает заходить в тупики.

## 2.2 Лабиринты 64x64

Теперь возьмем лабиринты большего размера, выставим большую область видимости 6x6 и посмотрим как вырастет время прохождения в сравнении с предыдущим кейсом. Также увеличим количество нейронов в сети до 20.

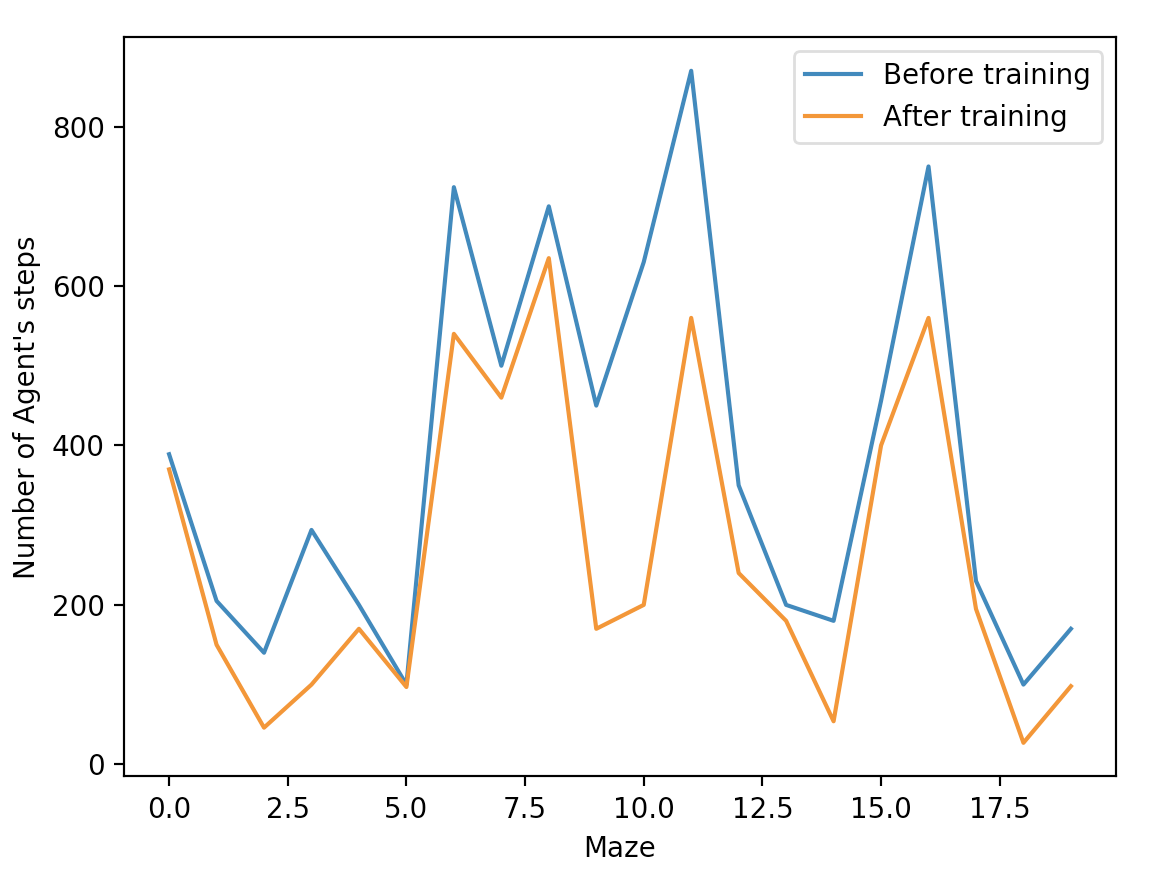
|  |  |
| --- | --- |
|  |  |
| **Рис. 2.3. Лабиринт 64x64** | |

Общее время работы алгоритма составило 5.1 минуты.

Из графика видно, что среднее время прохождения лабиринта сильно выросло, однако тенденция к обучению сохранилась, хотя уже и не выражена так явно. Это связано с тем, что чем больше размер лабиринта, а значит и препятствий на пути у агента, тем более извилистей становится его путь. Поэтому в некоторых случаях одному и тому же состоянию могут соответствовать разные оптимальные действия. Чтобы понять это агенту необходимо затратить определенное время. К тому же при увеличении размеров лабиринта возрастает количество путей, которые ведут к финишу, так как агент кроме своего опыта еще и исследует карту, он может ходить разными путями, что приводит к большой дисперсии требуемых шагов.

## 2.3 Процесс обучения агента на различных лабиринтах

До настоящего момента мы показывали, что агент способен обучаться только на одном типе лабиринтов. В данном эксперименте агенту на каждой итерации будет предложен лабиринт, по которому он не проходил ранее. Задача – продемонстрировать, что даже обучаясь на выборке совершенно разнородных сред, агент будет показывать результаты лучше, чем если бы он передвигался по лабиринтам не обладая приобретенным опытом. Все лабиринты в данном эксперименте обладают размером 32x32 и различным рисунком препятствий. Данные о скорости прохождения лабиринтов показаны на графике ниже:



**Рис. 2.4. Показатели шагов агента на различных лабиринтах до и во время обучения**

Из данного графика видно, что *общая* тенденция к обучению пропала. Такое поведение связано с тем, что каждый лабиринт в выборке отличается расположением стартовых и конечных точек, узором препятствий, а также их сложностью. Следовательно сравнивать необходимо лишь относительные показатели. В этом случае очевидно наблюдается влияние приобретенного опыта, агент тратит меньше времени на кластеризацию, так как уже приблизительно знает какие-то из паттернов (состояний) лабиринта.

## 2.4 Анализ результатов и влияние параметров

Как уже было сказано параметры модели играют существенную роль в процессе обучения агента. Правильно подобранные числовые характеристики могут повысить скорость прохождение лабиринта до нескольких раз. Рассмотрим более детально основные компоненты модели и среды:

1. Начальное пространственное распределение нейронов. То есть то, насколько плотно будут находится нейроны в пространстве относительно друг друга, насколько они будут схожи. А также насколько будут велики их нормы. Норма определяет насколько состояние соответствующее нейрону будет заполнено препятствующими движению элементами. Если лабиринт имеет небольшую плотность стен, то лучше генерировать нейроны с низкой нормой и наоборот. Также процесс кластеризации может быть значительно ускорен, если часть паттернов лабиринта известно заранее. Тогда можно сгенерировать долю нейронов по уже известному шаблону, что сократит время исследования и количество неопределенных состояний.
2. Константы, контролирующие скорость обучения, кластеризации, а также количество нейронов и тп. Например, чем ниже интенсивность кластеризации, то есть чем ниже сила пространственных весов, тем дольше, но качественнее работает алгоритм. Низкие веса стоит рассматривать в случае лабиринтов со сложной структурой паттернов, там, где отдельно взятые клетки влияют на движения агента (например, узкие трубки, нестандартные изгибы стен). Также существенную роль играют очки за правильные и неправильные действия. При высоких значениях данных параметров модель подвержена переобучению. Данную ситуацию стоит рассматривать только на очень простых лабиринтах, где путь для агента должен быть очевиден. Как же было сказано, за движение в сторону финиша агенту дается дополнительный бонус. Такое поведение модели стимулирует его идти по наиболее короткому пути. Однако сильная разница между данным бонусом и просто наградой за правильное действие может привести к тому, что агент не сможет изменить свой маршрут, если требуется обойти какое-то препятствие или выбрать другой путь следования.
3. Расположение начальной и конечной точек. Так как агенту дается награда за приближение к финишу, у него возникают собственные приоритеты в движениях при прохождении препятствий. Поэтому, если агент обучается вначале на выборке лабиринтов, где финиш находится справа снизу, а затем ему дается возможность пройтись по лабиринту, конечная точка в котором расположена слева сверху, это может значительно повлиять на время прохождения. Однако даже в этом случае агент будет проходить такой лабиринт быстрее, чем если бы он увидел его впервые, что мы и наблюдали на графике выше (Рис. 2.4).
4. Начальное расположение агента на карте. То насколько разнообразным будет опыт агента во время исследования, следуя правилу . Чем разнообразнее будут препятствия, которые агент встретит за этот период, то есть, чем больше он узнает, тем точнее он будет двигаться, опираясь исключительно на собственный опыт. Поэтому более правильным выбором является установка агента в достаточно запутанных местах со сложными контурами препятствий.
5. Область видимости, количество ложных путей и сложность тупиков. Как было показано в эксперименте такие препятствия могут сильно затормозить процесс прохождения лабиринта. В ходе исследований было обнаружено, что избежать этого можно увеличив область видимости агента. Иными словами, если агент полностью видит тупик, он классифицирует его как одно из ранее изученных препятствий и не станет заходить в него. Однако в случае ложного пути такого решения не существует и выбраться агенту поможет только рандомизация действий.

# ЗАКЛЮЧЕНИЕ

В настоящей работе представлена оригинальная нейросетевая архитектура интеллектуального агента, способного обучаться строить пути в различных лабиринтах. Архитектура основана на известной модели THSOM, с модификациями для использования в задаче обучения с подкреплением. По результатам проведенных экспериментов можно сделать вывод о сходимости процесса обучения.

Модель хорошо показала себя на небольших лабиринтах с отсутствием нетривиальных путей. Было также показано влияние тупиков на время прохождения лабиринтов, а также способ ускорения данного процесса. Из достоинств алгоритма стоит выделить вычислительную скорость, которая на небольших лабиринтах достигает несколько секунд, а также малый размер нейронной сети, составляющий лишь нескольких десятков нейронов для достаточно сложных лабиринтов.

Было также показано, что агент способен обучаться в различных средах, показывая прирост скорости прохождения в абсолютно новых лабиринтах. Данный факт позволяет модели быть применимой в реальных условиях, в частности когда требуется быстрый и требующий минимальных вычислительных затрат подход.

# Библиографический список

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning", 2013.

[2] Koutník and M. Šnorek, "Temporal Hebbian Self-Organizing Map for Sequences", 2008

[3] Gupta S. et al. Cognitive Mapping and Planning for Visual Navigation // ArXiv: 1702.03920.

[4] Schrodt F. et al. Mario Becomes Cognitive // Top. Cogn. Sci. 2017. P. 1–31.

[5] Paxton C. et al. Combining Neural Networks and Tree Search for Task and Motion Planning in Challenging Environments // ArXiv: 1703.07887. 2017.

[6] Hélie S., Sun R. Autonomous learning in psychologically-oriented cognitive architectures: A survey // New Ideas Psychol. 2014. Vol. 34, № 1. P. 37–55.

[7] Panov A.I. Behavior Planning of Intelligent Agent with Sign World Model // Biol. Inspired Cogn. Archit. 2017. Vol. 19. P. 21–31.

[8] Emel’yanov S. et al. Multilayer cognitive architecture for UAV control // Cogn. Syst. Res. 2016. Vol. 39. P. 58–72.

[9] Kaelbling L.P., Littman M.L., Moore A.W. Reinforcement learning: A survey // J. Artif. Intell. Res. 1996. Vol. 4. P. 237–285.

[10] Chalita M.A., Lis D., Caverzasi A. Reinforcement learning in a bio-connectionist model based in the thalamo-cortical neural circuit // Biol. Inspired Cogn. Archit. 2016. P. 45–63.