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

Филин Дмитрий1, Панов Александр1,2

1НИУ Высшая школа экономики, Москва, Россия

2ФИЦ ИУ РАН, Москва Россия

d.filin64@gmail.com

# Абстракт

В работе представлена модель обучения с подкреплением, использующая модифицированную нейросетевую архитектуру с пространственно- временными связями, также известную как Temporal Hebbian Self-Organizing Map (THSOM). Проведен ряд экспериментов с данной моделью на задаче нахождения пути в лабиринте. Модель показала устойчивое обучение, строя путь, близкий к оптимальному. В настоящей работе продемонстрировано поведение агента в лабиринтах различной сложности, а также показано, как различные параметры обучения влияет на длину формируемых путей.

# Введение

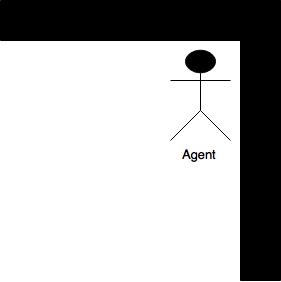
В настоящее время задача повышения уровня автономности робототехнических систем за счет интеграции в их системы управления расширенных подсистем представления знании и обучения, становится важным направлением в искусственном интеллекте и в когнитивных архитектурах [6-8]. Глубокое обучение с подкреплением 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».

Изначально алгоритм THSOM служит для генерации марковских последовательностей (восстановления вероятностей переходов между состояниями системы) по потоку входных данных. Поэтому интуитивно кажется, что такой подход должен успешно укладываться в концепцию Q-обучения. Благодаря кластеризации входных данных, мы дополнительно получаем снижение размерности пространства описаний состояния внешней среды при минимальной потере информации. В противоположность традиционному подходу [9], где Q-таблица хранит все состояния среды, мы утверждаем, что достаточно всего несколько состояний, чтобы полностью описать движение по определенному классу лабиринтов. Воспользовавшись разработанной в DeepMind архитектурой Q-сети [1], мы определил наблюдаемое агентом состояние среды, как изображение некоторого участка карты лабиринта при виде «сверху». При перемещении агент «видит» поле MxM вокруг себя. Стоит отметить, что изначально агент не знает ничего о среде в которой он двигается, кластеризация по паттернам происходит уже непосредственно в процессе исследования агентом пространства, за счет чего достигается независимость от карты самого лабиринта. Наконец, главным отличием от классического Q-обучения является то, что вся информация о передвижении хранится в связях между нейронами без использования дополнительных Q-таблиц, то есть моделируются связи между сенсорными и моторными отделами коры головного мозга [10].



*Рис.1 Примеры паттернов лабиринта*



*Рис. 2 Агент в одном из состояний среды*

# Формальная постановка задачи

Рассмотрим стохастическую среду в которой агенту доступен следующий набор действий:

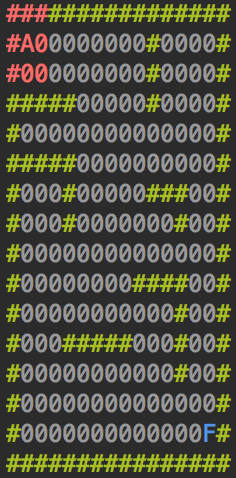
которые соответствуют следующим изменениям положения агента в пространстве :

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 Рис.3.



*Рис 3. Эмулированный лабиринт. Красным отмечена область, доступная для видимости агенту*

# Нейросетевая архитектура

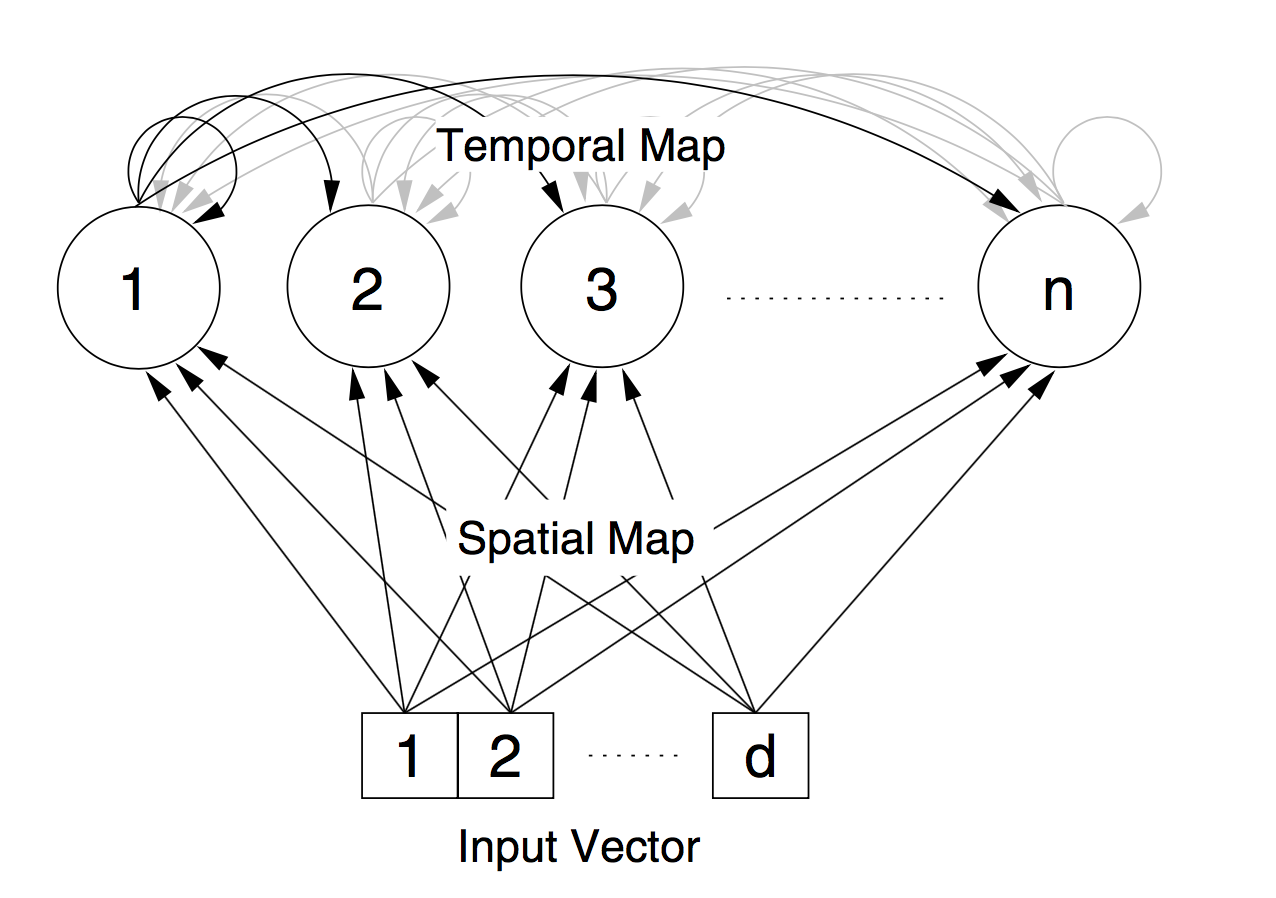
Как уже было сказано, в основе модели лежит нейронная сеть THSOM (Рис. 4), которая обладает двумя видами межнейронных связей – пространственными и временными. Основная идея заключается в том, что каждый входной вектор является аттрактором для нейронов, к которому часть из них итеративно стягивается в процессе обучения, таким образом образуются кластеры похожих паттернов. Данный процесс контролируется стандартным для алгоритмов обучения без учителя способом (например, алгоритм самоорганизующихся карт Кохонена). Радиус рассчитывается как:

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:

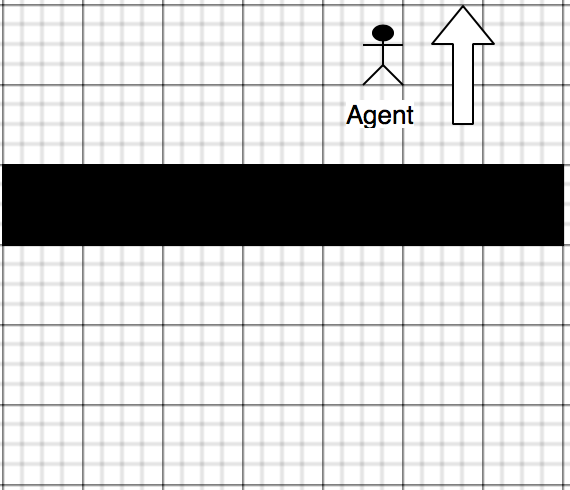
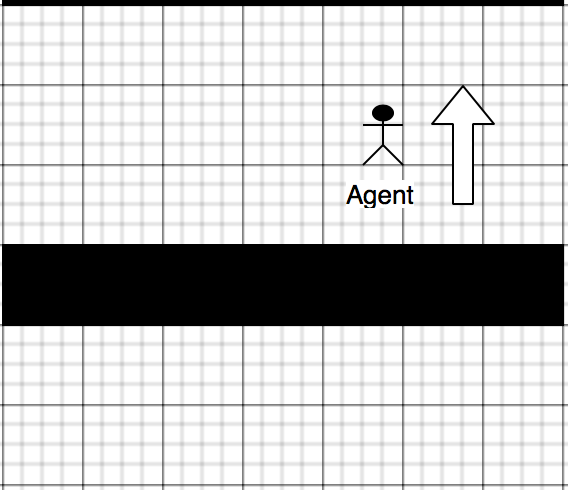
где нейрон, – входной вектор, расстояние между векторами.

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. 4) можно считать одинаковыми, и им соответствуют одни и те же доступные действия агента. Однако на Рис. 5 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:

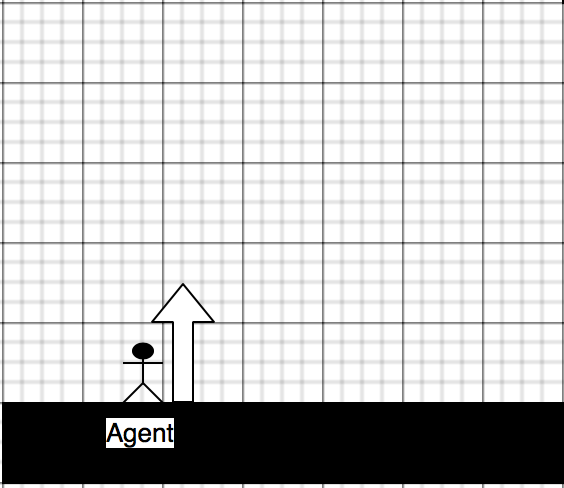
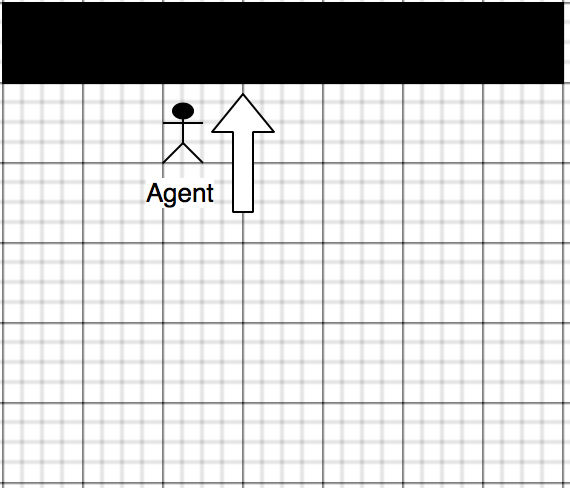
где– минимальная разница между паттернами при наложении их друг на друга (в единицах), – количество сдвигов для достижения этой разницы,



*Рис 4. Архитектура THSOM*



*Рис 5. Примеры похожих паттернов*



*Рис. 6 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:

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:

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 рандомно, накапливая опыт. В общем случае вероятность случайного действия вычисляется как:

Алгоритм обучения агента:

1. Проинициализировать временные и пространственные веса малыми случайными значениями
2. for t = 1, T do
   1. Считать входной сигнал с сенсоров (получить текущий блок MxM)
   2. Определить
   3. Обновить пространственные веса
   4. if t != 1 обновить временные веса
   5. Запомнить текущее состояние как
   6. В соответствии с выбрать
   7. Выполнить выбранное действие, запомнить
   8. Если не достигли финиша, перейти к шагу 2
3. end for

# Эксперименты

Мы протестировали наш алгоритм на различных последовательностях лабиринтов 16x16, отличающихся по сложности прохождения. Для начала было решено прогнать агента 5 раз на одном и том же лабиринте (Рис. 3), чтобы понять насколько хорошо проходит обучение. Количество шагов указано в таблице

*Таблица 1*

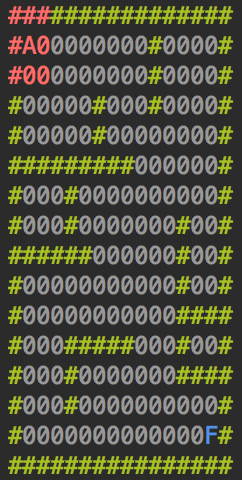
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итерация | 1 | 2 | 3 | 4 | 5 |
| Шаги | 119 | 84 | 49 | 33 | 42 |

Как видно, агент обучается и обучается довольно быстро. Учитывая начальное расположение агента относительно конечной точки, результаты можно считать сравнимыми с человеческими.

Однако на более сложных лабиринтах агент затричивает уже на порядок больше шагов. Например если лабиринт содержит тупики, агенту нужно затратить дополнительное время на то, чтобы выходить из трудных ситуаций, что сказывается на общем времени работы. В подобном случае как на рисунке 7 статистика получается следующей:

*Таблица 2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итерация | 1 | 2 | 3 | 4 | 5 |
| Шаги | 784 | 702 | 522 | 284 | 238 |

**

*Рис. 7 Пример более сложного лабиринта с тупиками*

Далее мы протестировали алгоритм на последовательности из пяти различных лабиринтах и усреднили результаты их прохождения:

*Таблица 3*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Лабиринты | 1 | 2 | 3 | 4 | 5 |
| Шаги | 529 | 758 | 329 | 557 | 205 |

Стоит отметить, что все лабиринты были различными относительно друг друга по уровню сложности паттернов и расположению начальных и конечных точек. В общем случае в ходе экспериментов были выяснены следующие обстоятельства, влияющие на время работы алгоритма:

1. 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 лабиринтах со сложной структурой паттернов, там, где отдельно взятые клетки влияют на движения агента (например, узкие трубки, нестандартные изгибы стен).
2. 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.
3. 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.

# 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. В общем случае, агент не только всегда находит путь, но делает это приемлемо быстро. В дальнейшем планируется более тщательно проанализировать каждый из параметров модели, чтобы добиться наилучшего по времени результата. Также мы работаем над способом, который поможет агенту быстрее разбираться с трудными ситуациями в виде тупиков и обманных путей и не застревать в них надолго.

# Литература

[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.