МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра Математического и компьютерного моделирования

# Автоматизация рекурсивного анализа позиции игры в шашки на небольшое число ходов

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы

направление 01.03.02 — Прикладная математика и информатика

механико-математического факультета

Дробышева Никиты Михайловича

Научный руководитель
доцент, к.т.н. С.П. Шевырев

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2020

**Введение.** Искусственный интеллект — это способность цифрового компьютера или управляемого компьютером робота выполнять задачи, обычно связанные с разумными существами. Термин часто применяется к проекту развития систем, наделенных интеллектуальными процессами, характерными для человека, такими как способность рассуждать, обобщать или учиться на прошлом опыте. Кроме того, определение понятия ИИ (искусственный интеллект) сводится к описанию комплекса родственных технологий и процессов, таких как, например, машинное обучение, виртуальные агенты и экспертные системы. Говоря простыми словами, ИИ — это грубое отображение нейронов в мозге.

Исходя из сказанного выше, нетрудно понять, насколько важную роль играет данная технология в современном мире и какие она несет за собой перспективы.

Но зададимся вопросом, а можно ли создать аналог ИИ или нейронной сети?

Для того чтобы разобраться в этом вопросе, для данной выпускной квалификационной работы бакалавра были поставлены следующие цели:

1. Изучение такой технологии как Искуственный Интеллект;
2. Изучение нейронных сетей;
3. Углубленное изучение рекурсивного анализа, как аналога нейронным сетям;
4. Написание програмного продукта для игры в "Шашки"

**Структура** работы включает в себя 4 раздела:

1. Первый раздел рассказывает об истории развития, перспективности и актуальности такой технологии как Искуственный Интеллект;
2. Во втором разделе работа знакомит непосредственно с таким понятием как Искуственные Нейронные Сети(ИНС), подробно рассматриваются следующие вопросы: Как же устроенна ИНС? Каков прнцип её работы? Процесс обучения системы. А так же подробно были рассмотренны виды нейронных сетей;
3. Третий раздел описывает рекурсивный анализ.
4. В четвертом разделе рассписан непосредственно сам процесс разработки, а так же какой подход к созданию кода был выбран.

**Основное содержание работы**

**Искусственный нейрон** — это математическая функция, задуманная как модель биологических нейронов, нейронной сети. Искусственные нейроны — элементарные единицы в искусственных нейросетях. Искусственный нейрон получает один или несколько входов и суммирует их, чтобы произвести выход или активацию, представляющую потенциал действия нейрона, который передается вдоль его аксона. Обычно каждый вход анализируется отдельно, и сумма передается через нелинейную функцию, известную как функция активации, или передаточная функция.

**Символьный подход к ИИ** — совокупность всех методов исследования искусственного интеллекта, основанных на высокоуровневых символических (читаемых человеком) представлениях о задачах, логике и поиске. Символьный подход широко применялся в исследованиях ИИ в 1950–80-х годах. Одной из популярных форм символьного подхода являются экспертные системы, использующие сочетание определенных правил производства. Производственные правила связывают символы в логические связи, которые подобны алгоритму If-Then. Экспертная система обрабатывает правила, чтобы сделать выводы и определить, какая дополнительная информация ей нужна, то есть какие вопросы задавать, используя удобочитаемые символы.

**Логический подходю**Термин «логический подход» предполагает апеллирование к логике, размышлениям, решению задач с помощью логических шагов. Логики еще в XIX веке разработали точные обозначения для всех видов объектов в мире и отношений между ними. К 1965 году существовали программы, которые могли решить любую логическую задачу (пик популярности данного подхода пришелся на конец 1950–70-х годов). Сторонники логического подхода в рамках логического искусственного интеллекта надеялись выстроить на таких программах (в частности, записанных на языке Prolog) интеллектуальные системы. Однако у такого подхода два ограничения. Во-первых, нелегко взять неформальное знание и изложить его в формальных терминах, которые требуются для обработки ИИ. Во-вторых, есть большая разница между решением проблемы в теории и ее решением на практике. Даже проблемы с несколькими сотнями фактов могут исчерпать вычисли-

тельные ресурсы любого компьютера, если у него нет каких-либо указаний относительно того, какие рассуждения надо использовать в первую очередь.

**Агентно-ориентированный подход.**Агент — это то, что действует (от лат. agere, «делать»). Конечно, все компьютерные программы что-то делают, но ожидается, что компьютерные агенты будут делать больше: работать автономно, воспринимать сигналы окружающей среды (с помощью специальных датчиков), адаптироваться к изменениям, создавать цели и выполнять их. Рациональный агент — это тот, кто действует так, чтобы достичь наилучшего ожидаемого результата.

**Влияние искусственного интеллекта.** Внедрение ИИ неразрывно связано с научно-техническим прогрессом, и сферы применения расширяются с каждым годом. Мы сталкиваемся с этим каждый день в жизни, когда крупная розничная сеть в интернете рекомендует нам какой-то товар или, только открыв компьютер, мы видим рекламу фильма, который как раз хотели посмотреть. Эти рекомендации основаны на алгоритмах, анализирующих то, что купил или смотрел потребитель. За этими алгоритмами стоит искусственный интеллект.

**Искуственные нейронные сети** – математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети Маккалока и Питтса. Впоследствии, после разработки алгоритмов обучения, получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть

— способ решения проблемы эффективного параллелизма. А с точки зрения искусственного интеллекта, ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

**Структура нейронной сети**. Для построения искусственной нейронной сети используется структура человеческого мозга. Как и биологическая нейронная сеть, искусственная состоит из нейронов, взаимодействующих между собой, однако представляет собой упрощенную модель. Так, например, искусственный нейрон, из которых состоит ИНС, имеет намного более простую структуру: у него есть несколько входов, на которых он принимает различные сигналы, преобразует их и передает другим нейронам. Другими словами, искусственный нейрон — это такая функция $\mathbb{R}^n \Rightarrow \mathbb{R}$, которая преобразует несколько входных параметров в один выходной.

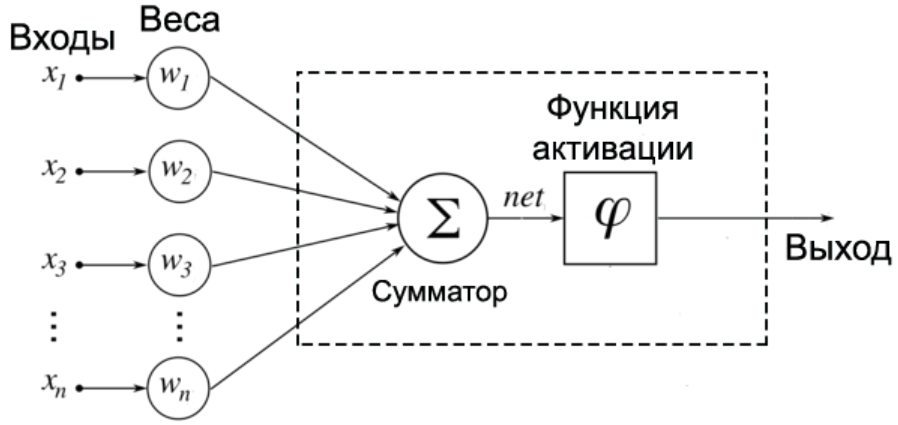


Рисунок 1 — Схема искусственного нейрона

Функции активации нейрона:

1. Функция единичного скачка. Если $net > treshold, \phi(net) = 1$, а иначе 0;
2. Сигмоидальная функция. $\phi(net) = \frac{1}{1+exp(-a*net)}$, где параметр $a$ характеризует степень крутизны функции;
3. Гиперболический тангенс. $\phi(net) = tanh(\frac{net}{a})$, где параметр $a$ характеризует степень крутизны графика функции;

**Рекуррентные нейронные сети** – сети с циклами, которые хорошо подходят для обработки последовательностей.

Обучение RNN аналогично обучению обычной нейронной сети. Мы также используем алгоритм обратного распространения ошибки, но с небольшим изменением. Поскольку одни и те же параметры используются на всех временных этапах в сети, градиент на каждом выходе зависит не только от расчетов текущего шага, но и от предыдущих временных шагов. Например, чтобы вычислить градиент для четвертого элемента последовательности, нам нужно было бы «распространить ошибку» на 3 шага и суммировать градиенты. Этот алгоритм называется «алгоритмом обратного распространения ошибки сквозь время».

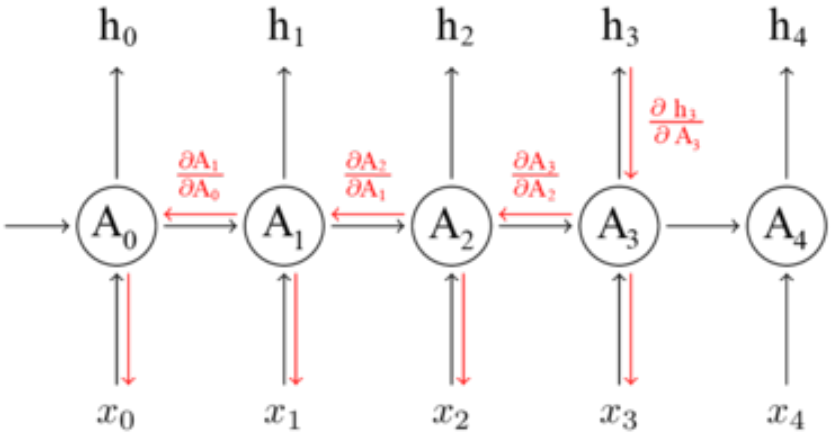


Рисунок 2 — Алгоритм обратного распространения ошибки сквозь время

**Рекурсивные нейронные сети** – тип глубокой нейронной сети, сформированный при применении одних и тех же наборов весов рекурсивно через структуру в виде дерева, чтобы сделать скалярное или структурированное предсказание над входными данными переменного размера через обход дерева в топологическом порядке.

При обучении последовательных структур и деревьев в задачах обработки естественного языка, фразы и предложения моделируются через векторное представление слов. Базовая структура сети является бинарным деревом, состоящим из родительского компонента (корня), и дочерних компонентов (листьев). Каждый компонент - набор нейронов, размер которого зависит от сложности входных данных. Входная последовательность данных подаётся на листья, а корень использует классификатор для определения класса и ме-

ры (score) Рекурсивная нейронная сеть использует следующую формулу для вычисления родительского вектора:

$$p_1 = f(W[b; c] + bias)$$

где,

1. $b, c$ - дочерние векторы;
2. $W$ - обученная матрица весов, $W \in R^{d*2d}$;
3. $f$ - нелинейная функция активации типа гиперболического тангенса;
4. $bias$ - смещение, оно может быть добавлено в качестве дополнительного столбца к $W$, если к конкатенации входных векторов добавляется 1.

Родительские векторы должны иметь одинаковую размерность, чтобы быть рекурсивно совместимыми и использоваться в качестве входных данных для следующей композиции.

Последующие шаги получают на вход меру предыдущего корня и следующее слово последовательности, таким образом пока в сети не будет сформировано дерево со всеми словами в последовательности.

Деревья могут иметь разную структуру, выбор лучшей подструктуры дерева для сети основывается на их мере. Мера дерева - сумма мер на каждом узле:

$$s(x, y) = \sum_{n \in nodes(y)} s_n$$

После выбора структуры, сеть классифицирует части последовательности. Вероятность принадлежности к классу вектора p вычисляется классификатором с помощью функции Softmax:

$$label_p = softmax(W^{label} p)$$

Здесь $W^{label}$ – матрица классификаций. Основной задачей и разницей между моделями будет вычисление скрытых векторов $p_i \in R^d$ снизу вверх.

**Алгоритм обратного распространения ошибки**. В рекурсивных нейронных сетях используется алгоритм обратного распространения ошибки

(backpropagation) с некоторыми отличиями, вытекающими из древовидной структуры и рекурсии:

1. Сумма производных матрицы W от всех узлов. Можно предположить, что она разная на каждом узле, однако если взять отдельные производные от каждого вхождения, то получится то же самое;
2. Разделение производных в каждом узле. Во время прямого распространения, родительский вектор считается через дочерние узлы по формуле выше. Следовательно, ошибки должны быть вычислены относительно каждого из них, причём ошибка каждого дочернего узла является n-мерной;

**Обучение нейронной сети** – поиск такого набора весовых коэффициентов, при котором входной сигнал после прохода по сети преобразуется в нужный нам выходной.

Это определение «обучения нейронной сети» соответствует и биологическим нейросетям. Наш мозг состоит из огромного количества связанных друг с другом нейросетей, каждая из которых в отдельности состоит из нейронов одного типа (с одинаковой функцией активации). Наш мозг обучается благодаря изменению синапсов — элементов, которые усиливают или ослабляют входной сигнал.

Если обучать сеть, используя только один входной сигнал, то сеть просто «запомнит правильный ответ», а как только мы подадим немного измененный сигнал, вместо правильного ответа получим бессмыслицу. Мы ждем от сети способности обобщать какие-то признаки и решать задачу на различных входных данных. Именно с этой целью и создаются обучающие выборки.

**Сущность рекурсии.** Рекурсивный подход подразумевает разделение сложной задачи, на один простой шаг к её решению и оставшуюся часть, которая становится упрощённой версией той же задачи. Затем этот процесс повторяется. Каждый раз вы совершаете один шаг, до тех пор, пока задача упростится до одного простейшего решения (его называют «базовым случаем»).

**Примеры рекурсивных алгоритмов.** Рекурсивный алгоритм всегда разбивает задачу на части, которые по своей структуре являются такими же как исходная задача, но более простыми. Для решения подзадач функция

вызывается рекурсивно, а их результаты каким-либо образом объединяются. Разделение задачи происходит лишь тогда, когда ее не удается решить сразу (она является слишком сложной).

Например, задачу обработки массива нередко можно свести к обработке его частей. Деление на части выполняется до тех пор, пока они не станут элементарными, т.е. достаточно простыми чтобы получить результат без дальнейшего упрощения.

1. Поиск элемента массива;
2. Двоичный поиск в массиве;
3. Вычисление чисел Фибоначчи;
4. Быстрая сортировка.

**Деревья.** Определение: Деревом будем называть конечное множество $T$, состоящее из одного или более узлов, таких что:

1. Имеется один специальный узел, называемый корнем данного дерева;
2. Остальные узлы (исключая корень) содержатся в $m \geq 0$ попарно непересекающихся подмножествах $T_1, T_2, ..., T_m$, каждое из которых в свою очередь является деревом. Деревья $T_1, T_2, ..., T_m$ называются поддеревьями данного дерева.

Это определение является рекурсивным. Если коротко, то дерево это множество, состоящее из корня и присоединенных к нему поддеревьев, которые тоже являются деревьями. Дерево определяется через само себя. Однако данное определение осмысленно, так как рекурсия конечна. Каждое поддерево содержит меньше узлов, чем содержащее его дерево.

**Minimax.** Минимакс-это правило принятия решений, используемое в искусственном интеллекте, теории принятия решений, теории игр, статистике и философии для минимизации возможных потерь при наихудшем сценарии (максимальных потерях). Когда речь идет о выигрыше, его называют "Максимин"—максимизировать минимальный выигрыш. Первоначально сформулированная для теории игр с нулевой суммой для двух игроков, охватывающая как случаи, когда игроки делают альтернативные ходы, так и те, когда они делают одновременные ходы, она также была распространена на более сложные игры и на общее принятие решений в присутствии неопределенности.

**Tkinter** – то кроссплатформенная библиотека для разработки графического интерфейса на языке Python (начиная с Python 3.0 переименована в tkinter). Tkinter расшифровывается как Tk interface, и является интерфейсом к Tcl/Tk. Tkinter входит в стандартный дистрибутив Python.

Библиотека предназначена для организации диалогов в программе с помощью оконного графического интерфейса (GUI).

**Разработка программного кода**

Сначала была решена задача по реализации шахматного поля:

```
def output(x_poz_1,y_poz_1,x_poz_2,y_poz_2):#рисуем игровое поле
        global pawn
        global pole
        global kr_ram,zel_ram
        k=100
        x=0
        board.delete('all')
        kr_ram=board.create_rectangle(-5, -5, -5, -5,outline="red",width=5)
        zel_ram=board.create_rectangle(-5, -5, -5, -5,outline="green",width=5)

        while x<8*k:#рисуем доску
                y=1*k
                while y<8*k:
                        board.create_rectangle(x, y, x+k, y+k,fill="black")
                        y+=2*k
                x+=2*k
        x=1*k
        while x<8*k:#рисуем доску
                y=0
                while y<8*k:
                        board.create_rectangle(x, y, x+k, y+k,fill="black")
                        y+=2*k
                x+=2*k
}
```

После этого зададим алгоритм передвишения шашки с учетом соблюдения правил игры, т.е. передвижение только по черным клеткам.

Далее была осуществленна логика хода компьютера:

```
def computer_run():#!!!
        global f_hi
        global n2_list
        validation_hk(1,(),[])
```

```
        if n2_list:#проверяем наличие доступных ходов
                kh=len(n2_list)#количество ходов
                th=random.randint(0,kh-1)#случайный ход
                dh=len(n2_list[th])#длина хода
                for h in n2_list:#!!!для отладки!!!
                        h=h#!!!для отладки!!!
                for i in range(dh-1):
                        #выполняем ход
                        list=run(1,n2_list[th][i][0],
                        n2_list[th][i][1],n2_list[th][1+i][0],n2_list[th][1+i][1])
                n2_list=[]#очищаем список ходов
                f_hi=True#ход игрока доступен
```

Логика хода компьютера заключается в следующем: создается список для нейронной сети, которая записывает ходы и в итоге, оставляет лучший вариант хода, т.е. обучается. В дальнейшем она пользуется этим списком для следующих игровых сессий.

Также в программе осуществленно превращение пешки в "дамку"с соблюдением игровых правил для неё.

```
def view_k1p(list,x,y):
        if pole[y][x]==3:#пешка
                for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):
                        if 0<=y+iy+iy<=7 and 0<=x+ix+ix<=7:
                                if pole[y+iy][x+ix]==1 or pole[y+iy][x+ix]==2:
                                        if pole[y+iy+iy][x+ix+ix]==0:
                                                list.append(((x,y),(x+ix+ix,y+iy+iy)))#запись
                                                хода в конец списка
        if pole[y][x]==4:#пешка с короной
                for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):
                osh=0#определение правильности хода
                for i in  range(1,8):
                        if 0<=y+iy*i<=7 and 0<=x+ix*i<=7:
                                if osh==1:
                                        list.append(((x,y),(x+ix*i,y+iy*i)))#запись хода
                                        в конец списка
                                if pole[y+iy*i][x+ix*i]==1 or pole[y+iy*i][x+ix*i]==2:
                                        osh+=1
                                if pole[y+iy*i][x+ix*i]==3 or pole[y+iy*i][x+ix*i]==4
                                        or osh==2:
                                        if osh>0:list.pop()#удаление хода из списка
                                        break
        return list
```

**Заключение.** В ходе данной дипломной работы, углубленно узучены такие сферы технологий, как: нейронные сети и искуственный интеллект. Так же был изучен метод рекурсивного анализа и такой метод, как Minimax. После реализации программы для игры в Шашки, были сделаны следующие выводы:

1. Метод рекурсивного анализа, для реализации своего рода ИИ, не запрашивает большое количество ресурсов;
2. Такой метод реализации отлично подходит для написания обучающих программ, которыми будут пользоваться новички,