**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра многопроцессорных систем и сетей**

**Реализация алгоритмов машинного обучения на основе компьютерной игры «2048»**

Отчёт по преддипломной практике

Качкова Дмитрия Ильича

студента 5 курса,

специальность «информатика»

Научный руководитель:

Росолько Алексей Владимирович

Минск, 2017

**РЕФЕРАТ**

Отчёт о преддипломной практике, 32 с., 6 источников, 7 таблиц, 4 рисунка.   
  
Ключевые слова: машинное обучение, алгоритмы, игра 2048, стратегия, генетический алгоритм, метод ближайших соседей.  
  
Объект исследования – основанные на машинном обучении алгоритмы игры в 2048.  
  
Цель работы – изучить различные алгоритмы машинного обучения и провести сравнение их эффективности на примере игры 2048, обобщить алгоритмы для целого класса игр.  
  
Результатом является теоретическая база, проработанная концепция использования алгоритмов оптимизации для получения эффективного алгоритма, реализованный гибкий сервер для игры в 2048, допускающий быструю замену компонентов, реализованы два алгоритма.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc475972865)

[ГЛАВА 1. ИГРА 2048 5](#_Toc475972866)

[1.1 Правила игры 5](#_Toc475972867)

[1.2 Модификации игры 6](#_Toc475972869)

[ГЛАВА 2. МАШИННОЕ ОБУЧЕНИЕ 7](#_Toc475972870)

[2.1 Определение машинного обучения 7](#_Toc475972871)

[2.2 Общая постановка задачи обучения по прецедентам 7](#_Toc475972872)

[2.3 Типология задач обучения по прецедентам 8](#_Toc475972873)

[2.3.1 Обучение с учителем 8](#_Toc475972874)

[2.3.2 Обучение без учителя 9](#_Toc475972875)

[2.3.3 Частичное обучение 10](#_Toc475972876)

[2.3.4 Трансдуктивное обучение 11](#_Toc475972877)

[2.3.5 Обучение с подкреплением 11](#_Toc475972878)

[2.3.6 Прочие подходы 12](#_Toc475972879)

[ГЛАВА 3. ПРОЕКТИРОВАНИЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ИГРЫ 2048 13](#_Toc475972880)

[3.1 Выбор алгоритмов 13](#_Toc475972881)

[3.1.1 Примитивные подходы 13](#_Toc475972882)

[3.1.2 Классификация позиций 15](#_Toc475972883)

[3.1.3 Построение стратегии 15](#_Toc475972884)

[3.2 Описание алгоритмов 16](#_Toc475972885)

[3.2.1 Метод ближайших соседей 16](#_Toc475972886)

[3.2.2 Генетический алгоритм 17](#_Toc475972887)

[ГЛАВА 4. РАЗРАБОТКА ОКРУЖЕНИЯ 19](#_Toc475972888)

[4.1 Общий обзор окружения 19](#_Toc475972889)

[4.2 Обзор разработанного игрового сервера 20](#_Toc475972890)

[4.3 Обзор обучающего компонента 23](#_Toc475972891)

[ГЛАВА 5. РЕАЛИЗАЦИЯ АВТОМАТИЧЕСКИХ ИГРОКОВ 25](#_Toc475972892)

[5.1 Алгоритм, основанный на генетическом алгоритме 25](#_Toc475972893)

[5.1.1 Описание алгоритма 25](#_Toc475972894)

[5.1.2 Достигнутые результаты 26](#_Toc475972895)

[5.2 Алгоритм, основанный на kNN 28](#_Toc475972896)

[5.2.1 Описание алгоритма 28](#_Toc475972897)

[5.2.2 Результаты 29](#_Toc475972898)

[5.3 Сравнение полученных результатов 29](#_Toc475972899)

[ЗАКЛЮЧЕНИЕ 31](#_Toc475972900)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc475972901)

# ВВЕДЕНИЕ

Машинное обучение (Machine Learning) — обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться, то есть улучшать свои показатели на основе сигналов о правильной или неправильной работе программы (обучение с учителем) либо обнаруживая закономерности во входных данных (обучение без учителя).

В рамках данной дисциплины предложено большое количество алгоритмов, различающихся положенными в их основу принципами. Естественным образом возникает задача определения наиболее эффективного алгоритма в рамках той или иной проблемы.

Одной из областей, в которой применимы алгоритмы машинного обучения, являются проблемы теория игр, среди которых можно выделить задачу построения оптимальной стратегии. Одним из примеров подобной игры, пригодной для проведения соответствующих исследований, является популярная игра 2048. В исходной постановке она рассматривается как головоломка, одна легко может быть переформулирована в виде игры для двух противников (роль второго в оригинальной версии игры выполняет генератор случайных чисел). В таком случае 2048 может рассматриваться как дискретная несимметричная последовательная игра с полной информацией.

Итак, целями данной работы являются:

* Изучение различных алгоритмов машинного обучения, таких как, например, генетический алгоритм, алгоритм пчелиной колонии, алгоритм имитации отжига, алгоритм k ближайших соседей;
* Создание играющих в игру 2048 обучающихся ботов, то есть автоматических игроков, следующих заложенным в них алгоритмам, основанных на принципах машинного обучения;
* Реализация удобного окружения для тестирования алгоритмов;
* Анализ полученных результатов и последующее сравнение эффективности рассмотренных алгоритмов в рамках рассматриваемой задачи.

# ГЛАВА 1. ИГРА 2048

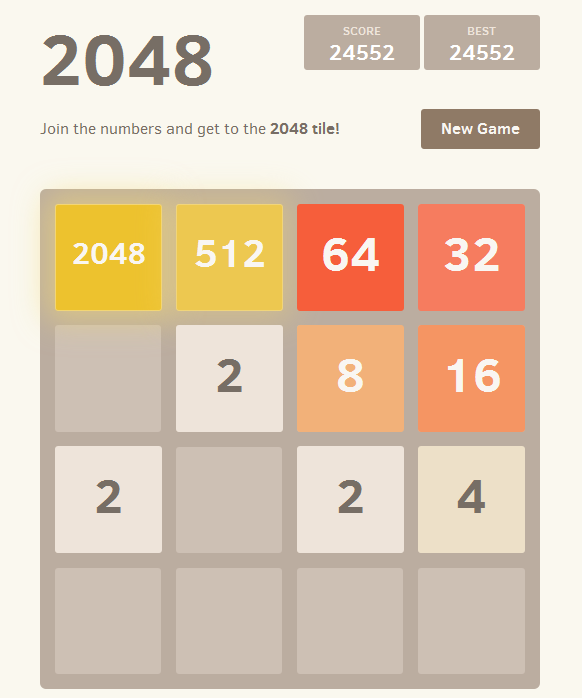
## 1.1 Правила игры

2048 – пошаговая логическая игра, действие которой происходит на игровом поле, имеющем форму квадрата 4x4. В каждой клетке этого поля может содержаться плитка (тайл), для которой указан её номинал – некоторая степень двойки. После каждого хода в одной из пустых клеток появляется новая плитка номинала «2» (с вероятностью 90 %) или «4» (с вероятностью 10 %). В начале партии (перед первым ходом) появляются два тайла. Ход заключается в том, что с помощью стрелок клавиатуры игрок скидывает все плитки в одну из сторон, при этом налетающие друг на друга плитки одного номинала объединяются в одну, номинал которой равен сумме объединившихся тайлов, то есть следующей степени двойки. Ход считается сделанным только в том случае, когда хотя бы одна плитка изменила своё положение. За каждую сложенную плитку игрок получает столько очков, каков номинал полученной плитки.

Рисунок 1. Демонстрация хода в игре 2048.

1. В позиции, представленной на первой картинке, первый игрок совершил ход «вверх».
2. В результате все тайлы поднялись вверх, при этом соседние плитки равных номиналов в этом направлении схлопнулись (две двойки в первом столбце превратились в четвёрку, а две восьмёрки во втором столбце – в шестнадцать).
3. На поле появилась новая плитка с номиналом, равным 2.

Цель игры — получить тайл с номиналом 2048, игра заканчивается поражением, если после очередного раунда невозможно совершить ход.

На рисунке 2 представлен скриншот оригинальной версии игры. Плитка 2048 уже собрана, однако имеется возможность продолжать игру.

## 

Рисунок 2. Скриншот оригинальной игры 2048

## 1.2 Модификации игры

Популярность игры 2048 и публикация её исходного кода в общем доступе на GitHub [1] способствовала разработке многочисленных модификаций данной игры. Указанные модификации могут отличаться от исходной игры размерами игрового поля, правилами перемещения и объединения плиток, наборами доступных плиток и условием победы. Среди них выделим следующие:

* игра на поле 3×3 или 5×5;
* так называемая Advanced 2048, где добавлены тайлы «×2» и «×4», которые объединяются с любой плиткой, удваивая (учетверяя) её номинал;
* Fibonacci, версия игры, в которой на тайлах указаны числа Фибоначчи, объединяются между собой те тайлы, на которых записаны последовательные числа Фибоначчи;
* игра, в которой новый тайл появляется не случайно, а намереваясь максимально ухудшить позицию играющему [2].

Последний вариант позволяет рассмотреть игру 2048 как соперничество двух игроков. Первый игрок делает очередной ход, указывая направление смещения всех тайлов на поле, и стремится получить наилучший результат. Роль второго игрока сводится к выставлению новых тайлов на поле, а задача – максимально помешать первому игроку. Оценивать результат можно по числу набранных очков, по максимальному номиналу тайла на поле или по числу совершённых ходов.

# ГЛАВА 2. МАШИННОЕ ОБУЧЕНИЕ

## 2.1 Определение машинного обучения

Машинное обучение (Machine Learning) — обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться, то есть улучшать свои показатели на основе сигналов о правильной или неправильной работе программы (обучение с учителем) либо обнаруживая закономерности во входных данных (обучение без учителя).

Различают два типа обучения. Обучение по прецедентам, или индуктивное обучение, основано на выявлении общих закономерностей по частным эмпирическим данным. Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.

Машинное обучение находится на стыке математической статистики, методов оптимизации и классических математических дисциплин, но имеет также и собственную специфику, связанную с проблемами вычислительной эффективности и переобучения. Многие методы индуктивного обучения разрабатывались как альтернатива классическим статистическим подходам. Многие методы тесно связаны с извлечением информации и интеллектуальным анализом данных (Data Mining).

Наиболее теоретические разделы машинного обучения объединены в отдельное направление, теорию вычислительного обучения (Computational Learning Theory, COLT).

Машинное обучение — не только математическая, но и практическая, инженерная дисциплина. Чистая теория, как правило, не приводит сразу к методам и алгоритмам, применимым на практике. Чтобы заставить их хорошо работать, приходится изобретать дополнительные эвристики, компенсирующие несоответствие сделанных в теории предположений условиям реальных задач. Практически ни одно исследование в машинном обучении не обходится без эксперимента на модельных или реальных данных, подтверждающего практическую работоспособность метода.

## 2.2 Общая постановка задачи обучения по прецедентам

Дано конечное множество прецедентов (объектов, ситуаций), по каждому из которых собраны (измерены) некоторые данные. Данные о прецеденте называют также его описанием. Совокупность всех имеющихся описаний прецедентов называется обучающей выборкой. Требуется по этим частным данным выявить общие зависимости, закономерности, взаимосвязи, присущие не только этой конкретной выборке, но вообще всем прецедентам, в том числе тем, которые ещё не наблюдались. Говорят также о восстановлении зависимостей по эмпирическим.

Наиболее распространённым способом описания прецедентов является признаковое описание. Фиксируется совокупность нескольких показателей, измеряемых у всех прецедентов. Если все показатели числовые, то признаковые описания представляют собой числовые векторы соответствующей размерности. Возможны и более сложные случаи, когда прецеденты описываются временными рядами или сигналами, изображениями, видеорядами, текстами, попарными отношениями сходства или интенсивности взаимодействия, и т. д.

Для решения задачи обучения по прецедентам в первую очередь фиксируется модель восстанавливаемой зависимости. Затем вводится функционал качества, значение которого показывает, насколько хорошо модель описывает наблюдаемые данные. Алгоритм обучения (learning algorithm) ищет такой набор параметров модели, при котором функционал качества на заданной обучающей выборке принимает оптимальное значение. Процесс настройки (fitting) модели по выборке данных в большинстве случаев сводится к применению численных методов оптимизации.

## 2.3 Типология задач обучения по прецедентам

Выделяют различные типы задач обучения по прецедентам, в основу классификации закладывая представление прецедентов [3].

### 2.3.1 Обучение с учителем

Обучение с учителем (supervised learning) — наиболее распространённый случай. Каждый прецедент представляет собой пару «объект, ответ». Требуется найти функциональную зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Функционал качества обычно определяется как средняя ошибка ответов, выданных алгоритмом, по всем объектам выборки.

В данной категории выделяют следующие задачи:

* задача классификации (classification) отличается тем, что множество допустимых ответов конечно. Их называют метками классов (class label). Класс — это множество всех объектов с данным значением метки;
* задача регрессии (regression) отличается тем, что допустимым ответом является действительное число или числовой вектор;
* задача ранжирования (learning to rank) отличается тем, что ответы надо получить сразу на множестве объектов, после чего отсортировать их по значениям ответов. Может сводиться к задачам классификации или регрессии. Часто применяется в информационном поиске и анализе текстов;
* задача прогнозирования (forecasting) отличается тем, что объектами являются отрезки временных рядов, обрывающиеся в тот момент, когда требуется сделать прогноз на будущее. Для решения задач прогнозирования часто удаётся приспособить методы регрессии или классификации, причём во втором случае речь идёт скорее о задачах принятия решений.

В данной работе будет разобран один из алгоритмов машинного обучения с учителем (алгоритм ближайших соседей), перед которым будет поставлена задача разбить все возможные игровые позиции на классы в зависимости от того, как ход следует совершить первому игроку.

### 2.3.2 Обучение без учителя

Обучение без учителя (unsupervised learning). В этом случае ответы не задаются, и требуется искать зависимости между объектами.

* Задача кластеризации (clustering) заключается в том, чтобы сгруппировать объекты в кластеры, используя данные о попарном сходстве объектов. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.
* Задача поиска ассоциативных правил (association rules learning). Исходные данные представляются в виде признаковых описаний. Требуется найти такие наборы признаков, и такие значения этих признаков, которые особенно часто (неслучайно часто) встречаются в признаковых описаниях объектов.
* Задача фильтрации выбросов (outliers detection) — обнаружение в обучающей выборке небольшого числа нетипичных объектов. В некоторых приложениях их поиск является самоцелью (например, обнаружение мошенничества). В других приложениях эти объекты являются следствием ошибок в данных или неточности модели, то есть шумом, мешающим настраивать модель, и должны быть удалены из выборки, см. также робастные методы и одноклассовая классификация.
* Задача построения доверительной области (quantile estimation) — области минимального объёма с достаточно гладкой границей, содержащей заданную долю выборки.
* Задача сокращения размерности (dimensionality reduction) заключается в том, чтобы по исходным признакам с помощью некоторых функций преобразования перейти к наименьшему числу новых признаков, не потеряв при этом никакой существенной информации об объектах выборки. В классе линейных преобразований наиболее известным примером является метод главных компонент.
* Задача заполнения пропущенных значений (missing values) — замена недостающих значений в матрице объекты–признаки их прогнозными значениями.

Для разработки искусственного интеллекта, играющего в 2048, возможно применить подход обучения без учителя, в частности, разработать самообучающуюся нейронную сеть. В данной работе этот подход не рассматривается.

### 2.3.3 Частичное обучение

Частичное обучение (semi-supervised learning) занимает промежуточное положение между обучением с учителем и без учителя. Каждый прецедент представляет собой пару «объект, ответ», но ответы известны только на части прецедентов.

Было замечено, что неразмеченные данные, будучи использованными совместно с небольшим количеством размеченных данных, могут обеспечить значительный прирост качества обучения. Под качеством обучения подразумевается некий функционал качества, например, среднеквадратичная ошибка. Сбор размеченных данных для задачи обучения зачастую требует, чтобы квалифицированный эксперт вручную классифицировал объекты обучения. Затраты, связанные с процессом разметки, могут сделать построение полностью размеченного набора прецедентов невозможным, в то время как сбор неразмеченных данных сравнительно недорог. В подобных ситуациях ценность частичного обучения сложно переоценить.

Примером частичного обучения может послужить сообучение: два или более обучаемых алгоритма используют один и тот же набор данных, но каждый при обучении использует различные — в идеале некоррелированные — наборы признаков объектов.

Альтернативный подход заключается в моделировании совместного распределения признаков и меток. В таком случае для неразмеченых данных метки могут трактоваться как пропущенные данные.

Пример прикладной задачи — автоматическая рубрикация большого количества текстов при условии, что некоторые из них уже отнесены к каким-то рубрикам.

### 2.3.4 Трансдуктивное обучение

Трансдуктивное обучение (transductive learning). Дана конечная обучающая выборка прецедентов. Требуется по этим частным данным сделать предсказания относительно других частных данных — тестовой выборки. В отличие от стандартной постановки, здесь не требуется выявлять общую закономерность, поскольку известно, что новых тестовых прецедентов не будет. С другой стороны, появляется возможность улучшить качество предсказаний за счёт анализа всей тестовой выборки целиком, например, путём её кластеризации. Во многих приложениях трансдуктивное обучение практически не отличается от частичного обучения.

Этот подход мало применим для разработки алгоритма игры в 2048, поскольку требует оценить сразу все возможные игровые позиции, что практически не реализуемо.

### 2.3.5 Обучение с подкреплением

Обучение с подкреплением, идея которого была почерпнута в смежной области психологии, является подразделом машинного обучения, изучающим, как агент должен действовать в окружении, чтобы максимизировать некоторый долговременный выигрыш. Алгоритмы с частичным обучением пытаются найти стратегию, приписывающую состояниям окружающей среды действия, которые должен предпринять агент в этих состояниях. В экономике и теории игр обучение с подкреплением рассматривается в качестве интерпретации того, как может установиться равновесие.

При обучении с подкреплением, в отличие от обучения с учителем, не предоставляются верные пары «входные данные-ответ», а принятие субоптимальных решений (дающих локальный экстремум) не ограничивается явно. Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний.

Откликом среды (а не специальной системы управления подкреплением, как это происходит в обучении с учителем) на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или ее модель. Также нужно иметь в виду, что некоторые правила подкрепления базируются на неявных учителях, например, в случае искусственной нейронной среды, на одновременной активности формальных нейронов, из-за чего их можно отнести к обучению без учителя.

Примеры прикладных задач: формирование инвестиционных стратегий, автоматическое управление технологическими процессами, самообучение роботов, и т.д.

В эту категорию можно отнести подход с обучением искусственного интеллекта игре в 2048, в основе которого лежит подбор с помощью генетического алгоритма оптимальных параметров, позволяющих ему более эффективно оценивать позицию. Этот подход подробно разобран в данной работе.

### 2.3.6 Прочие подходы

Динамическое обучение (online learning) может быть как обучением с учителем, так и без учителя. Специфика в том, что прецеденты поступают потоком. Требуется немедленно принимать решение по каждому прецеденту и одновременно доучивать модель зависимости с учётом новых прецедентов. Как и в задачах прогнозирования, здесь существенную роль играет фактор времени.

Активное обучение (active learning) отличается тем, что обучаемый имеет возможность самостоятельно назначать следующий прецедент, который станет известен.

Метаобучение (meta-learning или learning-to-learn) отличается тем, что прецедентами являются ранее решённые задачи обучения. Требуется определить, какие из используемых в них эвристик работают более эффективно. Конечная цель — обеспечить постоянное автоматическое совершенствование алгоритма обучения с течением времени.

Многозадачное обучение (multi-task learning). Набор взаимосвязанных или схожих задач обучения решается одновременно, с помощью различных алгоритмов обучения, имеющих схожее внутренне представление. Информация о сходстве задач между собой позволяет более эффективно совершенствовать алгоритм обучения и повышать качество решения основной задачи.

Индуктивный перенос (inductive transfer). Опыт решения отдельных частных задач обучения по прецедентам переносится на решение последующих частных задач обучения. Для формализации и сохранения этого опыта применяются реляционные или иерархические структуры представления знаний.

# ГЛАВА 3. ПРОЕКТИРОВАНИЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ИГРЫ 2048

## 3.1 Выбор алгоритмов

Рассмотрим описанные выше типы машинного обучения из соображения применимости к поставленной задаче.

Каждый бот может быть представлен как функция, которая каждой позиции ставит в соответствие ход, который, согласно положенному в его основу алгоритму, может быть рассмотрен, как оптимальный. Естественной оценкой эффективности данного бота является достигнутый им результат – число сделанных ходов, число набранных очков или максимально собранный тайл.

Сразу следует выделить два подхода к реализации алгоритмов машинного обучения, применимых к рассматриваемой игре:

* первый подразумевает разбиение множества всех позиций на классы, каждый из которых содержит те и только те позиции, в которых оптимальным является определённый ход. Например, для бота, исполняющего роль первого игрока, очевидно разбиение позиций на следующие классы: «оптимальный ход – вверх», «оптимальный ход – влево», «оптимальный ход – вниз», «оптимальный ход – вправо». Таким образом, при этом подходе прецедентом является отдельная позиция;
* второй подход подразумевает выделение некоторой стратегии, которой алгоритм придерживается на протяжении всей игры. При этом подходе прецедентом является целая игра.

### 3.1.1 Примитивные подходы

Среди алгоритмов, не являющихся алгоритмами машинного обучения, можно выделить несколько, относящихся ко второму подходу.

Наиболее примитивным является алгоритм случайного хода. Каждый ход определяется случайным образом и никоим образом не анализирует текущую позицию. Для данного бота были получены результаты, представленные в таблице 1 (здесь и далее в этой секции игра проходит по основным правилам, а действия второго игрока также случайны: в соответствии с правилами исходной игры, случайным в одной из пустых клеток появляется тайл, с вероятностью 90% он имеет номинал «2», с вероятностью 10% номинал «4»).

Таблица 1 – результаты, достигнутые с помощью случайных ходов

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Тайл** | 64 | 64 | 128 | 64 | 128 | 64 | 128 | 128 | 64 | 64 |
| **Ходы** | 87 | 108 | 152 | 73 | 142 | 86 | 126 | 142 | 88 | 101 |
| **№** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| **Тайл** | 128 | 256 | 256 | 64 | 128 | 128 | 128 | 128 | 32 | 128 |
| **Ходы** | 136 | 175 | 219 | 86 | 143 | 126 | 128 | 119 | 52 | 142 |

Графа «№» подразумевает номер очередной партии, графа «тайл» содержит информацию о максимальном собранном тайле, графа «Ходы» информирует о числе сделанных в партии ходов.

Таким образом, можно видеть, что алгоритм случайного хода в среднем делает чуть более 100 ходов и собирает как правило тайл 64 или 128.

Другим алгоритмом, не использующим подход машинного алгоритма, является строго заданная стратегия. В качестве примитивного примера можно привести бота, который всегда старается сделать ход «вверх»; если данный ход недоступен, делает ход «влево»; если и это невозможно, то «вправо»; наконец, если доступен только ход «вниз», совершает его. Для данной стратегии получены результаты, представленные в таблице 2:

Таблица 2 – результаты, достигнутые с помощью фиксированной стратегии

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Тайл** | 512 | 256 | 128 | 256 | 128 | 128 | 128 | 128 | 128 | 512 |
| **Ходы** | 511 | 269 | 164 | 294 | 186 | 169 | 246 | 152 | 194 | 407 |
| **№** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| **Тайл** | 256 | 128 | 512 | 256 | 128 | 128 | 128 | 256 | 256 | 256 |
| **Ходы** | 205 | 190 | 360 | 313 | 156 | 134 | 149 | 290 | 215 | 317 |

Очевидно преимущество данного алгоритма перед случайными ходами. Тем не менее, полученные результаты не являются оптимальными. Это объясняется в том числе тем, что предложенный алгоритм не анализирует текущую позицию. Введение дополнительных условий, определяющих выбор хода, могло бы усовершенствовать данный алгоритм и позволить достичь лучших результатов.

Однако основной недостаток фиксированной стратегии заключается в том, что для каждой конкретной позиции практически невозможно однозначно определить, какой ход является оптимальным, поскольку для этого требуется проанализировать полное дерево вариантов, которое может быть слишком велико.

Поэтому уместно отказаться от фиксированных стратегий и перейти к алгоритмам машинного обучения, которые позволят боту, проанализировав большое количество партий, выбирать те ходы, которые скорее всего окажутся наиболее оптимальными.

### 3.1.2 Классификация позиций

Рассмотрим первый из упомянутых подходов к реализации роботов. Задача разбиения множества всех возможных позиций на классы – типичная задача классификации. Она может быть решена, например, с применением метода ближайших соседей. Следует отметить, что применение данного метода к конкретной поставленной задаче может быть сопряжено с большими трудностями:

* данный алгоритм требует обучающей выборки, но нетривиальной задачей является построение множества позиций, для которых заведомо определён «ответ» – единственный оптимальный ход;
* сложность так же вызывает определение «близких» позиций – можно предложить ряд пар позиций, различающихся положением единственного тайла, в которых лучшими будут разные ходы.

### 3.1.3 Построение стратегии

Второй подход является более общим и допускает множество различных интерпретаций. Можно предложить, например, следующий алгоритм. На каждом ходу совершается просмотр всех допустимых ходов и определяется наилучшая из последующих позиций. Совершается тот ход, который приводит к выбранной наилучшей позиции. При этом при оценке позиции либо не учитывается новый тайл, либо рассматриваются все возможные варианты его появления и позиции выставляется усреднённая оценка по всем случаям. Также возможно применение более сложных алгоритмов, в частности, минимакс или минимакс с альфа-бета отсечением.

Возникает проблема построения функции оценивания позиции, для которой и требуется введение машинного обучения.

Для оценки позиции вводится ряд факторов, таких как, например:

* количество пустых клеток;
* количество тайлов различного номинала на доске;
* наличие тайла наибольшего номинала в углу поля;
* максимальная длина цепочки из соседствующих тайлов, в которой номинал каждого следующего меньше предыдущего;
* количество строк и столбцов поля, в которых тайлы идут строго по возрастанию номиналов;
* количество пар соседствующих тайлов одного номинала.

Значения всех введённых параметров суммируются с определёнными коэффициентами и полагается оценкой позиции. Подбор оптимальных коэффициентов является задачей обучения.

Переформулировав задачу, получаем, что на множестве задана функция , которая по набору заданных коэффициентов возвращается число – результат игры (максимальный тайл, число ходов, число очков либо некоторый усреднённый результат нескольких игр). Для корректного задания функции предположим, что второй игрок делает квазислучайные ходы – т.е. появление новых тайлов выглядит произвольным, но при этом повторение одних и тех же ходов приводит к полному повторению всей партии. С учётом этого допущения функция будет задана однозначно. Возникает задача найти точку максимума данной функции, и для решения этой задачи могут быть применены различные алгоритмы оптимизации, в частности генетический алгоритм. Также могут быть рассмотрены, например, алгоритм имитации отжига, алгоритм пчелиной колонии и т.д.

## 3.2 Описание алгоритмов

Рассмотрим более подробно те алгоритмы, которые были упомянуты в предыдущей секции.

### 3.2.1 Метод ближайших соседей

Метод k ближайших соседей (англ. k-nearest neighbors algorithm, k-NN) — метрический алгоритм для автоматической классификации объектов. Основным принципом метода ближайших соседей является то, что объект присваивается тому классу, который является наиболее распространённым среди соседей данного элемента.

Соседи берутся исходя из множества объектов, классы которых уже известны, и, исходя из ключевого для данного метода значения k высчитывается, какой класс наиболее многочислен среди них. Каждый объект имеет конечное количество атрибутов (размерностей).

Предполагается, что существует определенный набор объектов с уже имеющейся классификацией.

Выделяют несколько типов данного алгоритма:

Метод ближайшего соседа является, пожалуй, самым простым алгоритмом классификации. Классифицируемый объект относится к тому классу , которому принадлежит ближайший объект обучающей выборки .

Метод k ближайших соседей. Для повышения надёжности классификации объект относится к тому классу, которому принадлежит большинство из его соседей — k ближайших к нему объектов обучающей выборки .

Метод взвешенных ближайших соседей. Каждому соседу приписывается вес , как правило, убывающий с ростом ранга соседа i. Объект относится к тому классу, который набирает больший суммарный вес среди k ближайших соседей[4].

### 3.2.2 Генетический алгоритм

Генетический алгоритм — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе[5].

Задача формализуется таким образом, чтобы её решение могло быть закодировано в виде вектора («генотипа») генов, где каждый ген может быть битом, числом или неким другим объектом. В классических реализациях генетического алгоритма предполагается, что генотип имеет фиксированную длину. Однако существуют вариации алгоритма, свободные от этого ограничения.

Некоторым, обычно случайным, образом создаётся множество генотипов начальной популяции. Они оцениваются с использованием «функции приспособленности», в результате чего с каждым генотипом ассоциируется определённое значение («приспособленность»), которое определяет насколько хорошо фенотип, им описываемый, решает поставленную задачу. Этой функцией является построенная ранее функция .

Из полученного множества решений («поколения») с учётом значения «приспособленности» выбираются решения (обычно лучшие особи имеют большую вероятность быть выбранными), к которым применяются «генетические операторы» (в большинстве случаев «скрещивание» — crossover и «мутация» — mutation), результатом чего является получение новых решений. Для них также вычисляется значение приспособленности, и затем производится отбор («селекция») лучших решений в следующее поколение.

Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий остановки алгоритма. Таким критерием может быть нахождение глобального, либо субоптимального решения; исчерпание числа поколений, отпущенных на эволюцию; исчерпание времени, отпущенного на эволюцию.

Таким образом, можно выделить следующие этапы генетического алгоритма:

* задать целевую функцию (приспособленности) для особей популяции;
* создать начальную популяцию;
* (Начало цикла);

1. Размножение;
2. Мутирование;
3. Вычисление значение целевой функции для всех особей;
4. Формирование нового поколения;
5. Если выполняются условия остановки, то (конец цикла), иначе (начало цикла).

# ГЛАВА 4. РАЗРАБОТКА ОКРУЖЕНИЯ

## 4.1 Общий обзор окружения

Для последующего тестирования алгоритмов на языке Java был реализован фреймворк, позволяющий как обучать искусственных игроков, так и оценивать их результаты в играх между собой.

Разберём структуру проекта более подробно.

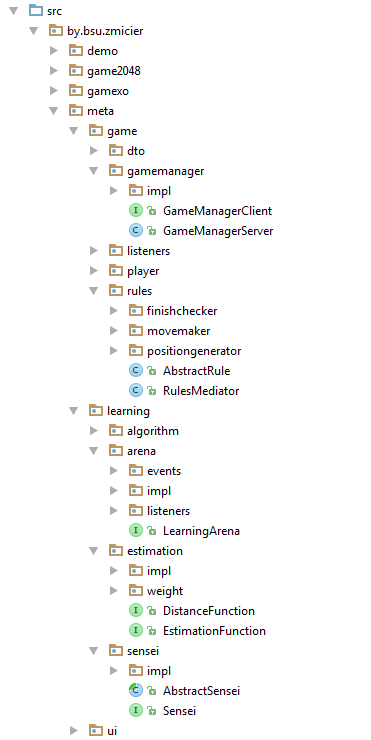


Рисунок 3. Структура проекта

## 4.2 Обзор разработанного игрового сервера

Хотя основное исследование проводилось на игре 2048, данный фреймворк гораздо гибче и может быть использован для любой игры с полной информацией, в которой два игрока совершают ходы по очереди, в частности, крестики-нолики, шахматы, го.

Ключевым пакетом является пакет by.bsu.zmicier.meta, в котором реализована основная логика абстрактной мета-игры для двух игроков.

Каждую игру можно представить как объединение набора правил, реализаций двух игроков и некоторой абстрактной позиции (MetaPosition), которую корректно рассматривать в более широком смысле — как текущее состояние игры. Так, например, при реализации шахмат имеет смысл хранить не только непосредственно позицию, но и такие параметры, как возможность в данный момент совершать рокировку или брать пешки на проходе.

Правила каждой игры задаются с помощью наследников класса AbstractRule. По умолчанию требуются следующие правила:

* PositionGenerator, предоставляющий начальную позицию для игры;
* FinishChecker, в чьи обязанности входит определять, является ли текущая позиция конечной;
* MoveMaker, ответственный за определение корректных ходов в данной позиции и вычисление нового состояния игры после совершения хода.

Кроме того существует возможность реализовывать собственные правила. Так, например, для игры 2048 и её вариаций можно добавить компоненты, отвечающие за логику объединения тайлов (объединяются только равные, как в 2048, или только соседние числа Фибоначчи, как в соответствующей игре), за подсчёт набранных игроком очков и за определение того, какие тайлы могут быть добавлены в позицию вторым игроком.

Для обеспечения низкой связности компонентов для доступа к правилам используется класс RulesMediator, представляющий собой реализацию паттерна проектирования «Посредник» («Mediator»). Таким образом, каждый компонент правил имеет доступ к остальным правилам, что позволяет реализовывать более сложную логику.

Каждый игрок представляет собой реализацию интерфейса Player, основная логика вынесена в метод move. На уровне мета-игры представлены несколько реализаций игроков, в частности:

* RandomPlayer совершает случайный ход из списка возможных;
* QuasiRandomPlayer – имитация случайного игрока, однако не содержащая элемента случайности. Используется для построения оценки успешности выбранной стратегии , описанной в разделе 3.1.3;
* HumanPlayer обеспечивает возможность сыграть в игру человеку.

RulesMediator и два игрока являются членами класса GameManagerServer, ответственным непосредственно за проведение игры. Для обеспечения безопасности игры, также существует интерфейс GameManagerClient, реализации которого передаются в качестве аргумента в метод move интерфейса Player. GameManagerClient предоставляет возможность получить список допустимых с точки зрения установленных на сервере правил ходов в данной позиции, а также узнать результат того или иного хода, однако лишает игрока возможности обращаться к правилам или текущему состоянию игры напрямую и изменять их.

Для демонстрации возможностей фреймворка был реализован набор компонентов для различных вариаций игры в 2048. В частности, можно сыграть в следующие игры:

* Классическое 2048;
* Игра Фибоначчи (номинал плиток – числа Фибоначчи, два соседних числа объединяются в следующее, например, тайлы с номиналами 3 и 5 объединятся в плитку с номиналом 8);
* Простая Dive-игра (две плитки объединяются только тогда, когда номинал одной делится на другую. На доску могут быть добавлены с номиналами 2, 3, 5 или 7);
* Сложная Dive-игра (аналогично простой, но на доску может быть добавлена любая плитка с номиналом, являющимся простым делителем любой плитки на доске).

Для этого класса игр реализованы дополнительные игроки, в частности следующие первые игроки:

* Игрок, придерживающийся описанной в пункте 3.1.1 примитивной стратегии выбора приоритетного хода;
* UIHumanPlayer, предоставляющий возможность сыграть живому игроку через графический интерфейс.

Кроме того, на роль второго игрока добавлен WeightedRandomPlayer, который совершает случайные ходы, однако вероятность выбора того или иного номинала тайла распределена неравномерно. Этот игрок соответствует второму игроку оригинальной игры.

Также реализован набор правил для игры в крестики-нолики.

Существенным элементом сервера является GameListenerChain, хранящая список реализаций интерфейса GameListener. Эта связка является реализацией паттерна проектирования «Слушатель» («Listener»). Каждый слушатель уведомляется о таких событиях, как начало новой партии, ход того или иного игрока, завершение партии. Соответственно, к играм можно добавлять различные слушатели, которые будут так или иначе реагировать на происходящее в игре.

В частности, были разработаны следующие слушатели: хранящий статистику по серии партий (среднее количество очков, лучший результат и тому подобное), отображающий результат каждой игры в консоль и визуализатор игры с использованием swing. В дальнейшем можно реализовать слушателя, ответственно за сохранение информации об играх в базу данных.

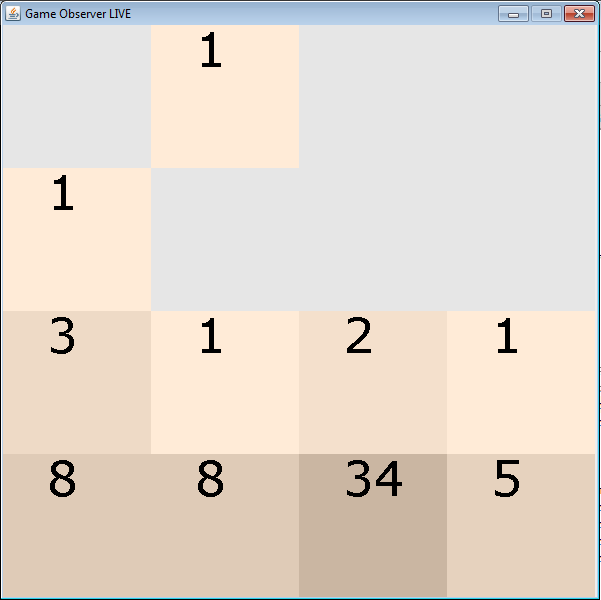


Рисунок 4. Визуализатор игры. Версия «Фибоначчи»

Также с использованием слушателя ArenaListener реализовано обучение алгоритмов.

## 4.3 Обзор обучающего компонента

Центральным компонентом обучающего модуля является объект LearningArena. Он включает в себя сервер и две обучающие сущности – Sensei.

Работу LearningArena обеспечивает ArenaListener, который неявно добавляется в цепочку слушателей на сервере.

ArenaListener хранит список объектов Sensei (двух, играющих на LearningArena, а также произвольное число обозревателей, которые могут реализовывать алгоритмы машинного обучения, не требующего участия в формировании обучающей выборке) и уведомляет каждого о событиях, происходящих на сервере. Таким образом, ArenaListener и цепочка Sensei является ещё одной реализацией паттерна «Слушатель» («Listener»).

Наконец, интерфейс Sensei и его базовая реализация AbstractSensei содержат следующие методы:

* getStudent, отвечающий за получение очередного обучающегося игрока;
* getMaster, получающий текущего «мастера» – обученного игрока;
* processEvent и более узкоспециализированные методы (processNewGameEvent, processMyMoveEvent и так далее) – методы, отрабатывающие при том или ином событии и заключающие в себе логику обучения.

Для примера были реализованы два алгоритма машинного обучения – GeneticSensei, основанный на генетическом алгоритме и подбирающий оптимальные весовые коэффициенты для ряда эвристик, оценивающих текущую позицию, а также KNNSensei, подготавливающий обучающую выборку для игрока. Соответственно, реализовано также несколько игроков, соответствующих этим алгоритмам:

* AbstractEstimatorPlayer, выбирающий ход в соответствии с построенными оценками позиций. Для оценивания используется реализация интерфейса EstimationFunction, в качестве которой можно использовать результаты работы генетического алгоритма. В качестве стратегии можно выбрать как непосредственный выбор хода, приводящего к позиции с наилучшей оценкой, так и более глубокие алгоритмы, например, минимакс;
* KNNPlayer, хранящий обучающую выборку и функцию расстояния между позициями. Стоит отметить, что эта функция может быть задана явно, но теоретически её поиск можно осуществлять с использованием генетического алгоритма.

Таким образом, для инициации обучения необходимо выполнить определённую простую последовательность действий.

1. Создать объект GameManagerServer и установить на нём правила игры;
2. Создать объект ArenaListener и установить на ней подготовленный сервер;
3. Установить длительность обучения;
4. Подготовить двух Sensei и установить их на арене;
5. Запустить обучение;
6. По окончании обучения можно получить объекты, соответствующие обученными искусственным интеллектам, использовав метод getMaster().

# ГЛАВА 5. РЕАЛИЗАЦИЯ АВТОМАТИЧЕСКИХ ИГРОКОВ

## 5.1 Алгоритм, основанный на генетическом алгоритме

### 5.1.1 Описание алгоритма

Первым из двух реализаций искусственного интеллекта стал робот, который обучается с использованием генетического алгоритма.

В общем случае подход следующий. Для заданной игры указывается набор эвристических функций оценки позиции , где – множество всех достижимых позиций в игре. При этом очевидно, что выбранные параметры могут не быть равноправными, одни из них окажутся более существенными, другие вовсе не характеризуют качество позиции. Поэтому для вычисления финальной оценки каждый высчитанный параметр умножается на определённый вес. Затем их всех возможных ходов выбирается тот, который приводит к позиции с наивысшей оценкой.

Таким образом, непосредственно в момент игры оценивание происходит по строго детерминированному алгоритму без участия машинного обучения. Подбирая произвольные значения весов, можно получать множество реализаций искусственных интеллектов, демонстрирующих различные подходы к игре.

При обучении искусственного интеллекта игре в 2048 использовались следующие факторы:

* количество пустых клеток;
* количество соседствующих тайлов равного номинала как по вертикалям и горизонталям раздельно, так и по всем направлениям в сумме;
* аналогичный параметр, но при оценке соседствующие тайлы учитываются по-разному в зависимости от номинала;
* длина длиннейшей цепи – последовательности монотонно убывающих соседствующих тайлов;
* длина длиннейшей цепи, начинающейся из максимального тайла;
* количество тайлов различных номиналов на доске;
* логический параметр, определяющий, находится ли тайл максимального номинала в углу игрового поля;
* количество не соседствующих между собой тайлов максимального номинала;
* количество строк, столбцов, в которых все тайлы расположены строго по возрастанию или строго по убыванию номиналов;
* количество пиков, то есть тайлов, которые одновременно меньше либо одновременно больше по номиналу своих соседей по одному направлению;
* номинал наибольшего тайла.

В рамках работы в соответствии с этим подходом были реализованы как первый, так и второй игроки (второй – в двух версиях, «злого», старающегося максимально ухудшить позицию, и «доброго», помогающего первому игроку). В качестве оценивающий функции было выбрано среднее количество очков, набранное за партий (большинство экспериментов проводилось при или ).

### 5.1.2 Достигнутые результаты

По результатам обучения был получен двадцатиоднокомпонентный вектор коэффициентов, который использовался для выбора стратегии искусственным интеллектом.

Для эксперимента была сыграна 1001 партия, в которой искусственный интеллект играл роль первого игрока и тайлы появлялись на поле случайным образом. Также был рассмотрен вариант бота, который применяет тот же основанный на оценке позиций подход, однако перед каждой партией генерирующий новый случайный вектор коэффициентов, то есть опускающий фазу обучения. В таблице 3 приведены результаты эксперимента:

|  |  |  |
| --- | --- | --- |
| Таблица 3 – результаты первого игрока, достигнутые с помощью различных алгоритмов | | |
| **Алгоритм** | **Genetic** | **RandomCoeffs** |
| Лучший результат | 31368 | 14104 |
| Средний результат | 7324.3076 | 5052.317 |
| Медиана | 7012 | 5388 |
| Количество побед | 1 | 0 |

Преимущество перед алгоритмом случайных коэффициентов наглядно показывает, что предварительное обучение действительно имеет эффект. Сравнение этих результатов с прочими алгоритмами будет наглядно продемонстрировано в разделе 5.3.

В победной партии было совершено 1568 ходов, робот сложил тайлы 2048 и 1024, однако в разных частях поля. Итоговая позиция продемонстрирована в таблице 4.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 4 **–** итоговая позиция в победной партии | | | |
| 2 | 1024 | 4 | 2 |
| 4 | 2 | 8 | 64 |
| 16 | 256 | 32 | 4 |
| 2 | 4 | 16 | 2048 |

Чтобы оценить алгоритм «доброго» и «злого» второго игрока, также были отыграна 1001 партия против случайного алгоритма.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 5 – результаты второго игрока, достигнутые с помощью различных алгоритмов | | | |
| **Алгоритм** | **AngryGenetic** | **KindGenetic** | **Random** |
| Лучший результат первого игрока | 1372 | 6860 | 3568 |
| Средний результат первого игрока | 420.1439 | 2858.693 | 1082.1778 |
| Медиана | 400 | 2788 | 1048 |
| Количество побед первого игрока | 0 | 0 | 0 |

На этом примере также хорошо видно, что искусственный интеллект выступает значительно лучше алгоритма случайного выбора. Так, «добрая» реализация помогла случайному алгоритму набрать 6860 очков, то есть сложить тайлы 512, 256 и 128 и вплотную приблизиться к 1024. Противостоя «злой» реализации, случайный алгоритм крайне редко справлялся плиткой в 128, а как минимум однажды не смог собрать даже 16.

Кроме того, естественно было схлестнуть двух генетических роботов между собой. Поскольку они придерживаются детерминированных алгоритмов, элемент случайности вносила исключительно стартовая позиция. Необходимости в 1001 запуске не было, эксперимент был проведён для всех возможных исходных позиций.

Против «злого» второго игрока первый игрок смог набрать максимум 3756 очков, в среднем – чуть более 2000.

Против «доброго» второго игрока первый игрок одерживает около 80% побед. В самой успешной партии было набрано 33156 очков – это собранные тайлы 2048, 1024 и 512.

Заметим, что в целом результаты могли быть гораздо выше, если бы при выборе хода проводилось более глубокое исследование, например, использовался алгоритм минимакса с просмотром на несколько ходов вперёд. Одно из преимуществ данной реализации генетического алгоритма именно в том, что она может легко интегрироваться с прочими алгоритмами.

Существенный недостаток процесса обучения заключается в том, что используемая оценивающая функция недостаточно гладкая, как того требует теория алгоритма генетического обучения. Из-за этого подбор коэффициентов может осуществляться неэффективно.

В частности, несмотря на большое количество исследованных поколений геномов, сложно выделить общую тенденцию в построении векторов коэффициентов. Единственная ярко выраженная тенденция – существенный вес имеет параметр количества пустых клеток на игровом поле.

## 5.2 Алгоритм, основанный на kNN

### 5.2.1 Описание алгоритма

Алгоритм kNN, в отличие от предыдущего, ориентирован не на анализ последующих позиций, которые могут получиться в итоге того или иного хода, а на текущую позицию. Сравнивая позицию с прочими, имеющимися в базе данных, алгоритм выбирает тот ход, который чаще всего совершался в похожих ситуациях.

Таким образом, для работы алгоритма требуется обучающая выборка. Один из подходов – обучить алгоритм на подборке партий, в которых был достигнут высокий результат. Так, в данной работе для обучения использовались ходы партий, в каждой из которых была сложена плитка 4096.

Для оценки сходства позиции важно учитывать расположение плиток на поле, поэтому необходимо избегать таких параметров, как количество пустых клеток на игровом поле.

В работе в пространстве игровых позиций введена сложная метрика. Для вычисления расстояний между элементами последовательно анализируются все комбинации из одной и двух клеток игрового поля.

### 5.2.2 Результаты

Для эксперимента была сыграна 1001 партия, в которой смоделированные искусственный интеллект играл роль первого игрока и тайлы появлялись на поле случайным образом. В среднем бот набирал 3520 очков, а в лучшей партии показал результат 12764. Медиана выборки оказалась равна 2916. Сравнение этих результатов с результатами прочих алгоритмов будет наглядно продемонстрировано в разделе 5.3.

После анализа сыгранных партий можно описать основное слабое место алгоритма. При построении обучающей выборки придерживались стратегии держать самую большую плитку в углу. Поскольку это общая тенденция по всей выборке, алгоритм её улавливает и придерживается. Тем не менее, иногда складывается ситуация, что необходимо переместить самую большую плитку в другое место. В этой ситуации требуется очень аккуратная игра, чтобы вернуть её на место, на которую не способен искусственный интеллект. В результате он совершает серию ошибочных ходов и почти сразу проигрывает.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 6 **–** итоговая позиция в лучшей партии | | | |
| 2 | 32 | 16 | 4 |
| 4 | 1024 | 512 | 2 |
| 16 | 8 | 64 | 16 |
| 2 | 4 | 8 | 4 |

## 5.3 Сравнение полученных результатов

Таким образом, в рамках проделанной работы для разных алгоритмов были сыграны 1001 партия. Для сравнительного анализа разработанных автоматических игроков сведём воедино в таблицу 7 данные все полученные результаты, выделяя лучший результат, среднее арифметическое набранных очков и медиану полученной выборки.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 7 – результаты первого игрока, достигнутые с помощью различных алгоритмов | | | | | |
| **Алгоритм** | **kNN** | **Genetic** | **RandomCoeffs** | **Random** | **Corner** |
| Лучший результат | 12764 | 31368 | 14104 | 3568 | 8968 |
| Средний результат | 3519.8833 | 7324.3076 | 5052.317 | 1082.1778 | 2678.042 |
| Медиана | 2916 | 7012 | 5388 | 1048 | 2612 |
| Количество побед | 0 | 1 | 0 | 0 | 0 |

Очевидно большое преимущество генетического алгоритма как перед алгоритмом случайного выбора, так и перед алгоритмом, придерживающимся примитивной детерминированной стратегии. Это единственный алгоритм, которому удалось одержать победу и сложить тайл 2048. Как отмечалось в разделе 5.1.2, превосходство результатов над алгоритмом RandomCoeffs показывает, что обучение имеет смысл и позволяет усовершенствовать результат.

Алгоритм kNN оказался менее эффективным, чем генетический алгоритм. Это можно объяснить тем, что данный алгоритм полностью основан на машинном обучении и только разбивает позиции на классы «похожих», в то время как реализация генетического алгоритма включает оценку позиции, благодаря чему показывает высокий результат даже без фазы обучения, что продемонстрировано на примере алгоритма случайных коэффициентов.

Тем не менее, результат алгоритма ближайших соседей превосходит результат случайного игрока и даже алгоритм, придерживающийся определённой примитивной стратегии, вследствие чего можно утверждать, что подобный подход может быть применим при реализации автоматического игрока в 2048, но нуждается в совершенствовании

# ЗАКЛЮЧЕНИЕ

В ходе проделанной работы:

1. изучены различные подходы к задачам машинного обучения, исследованы различные методы и алгоритмы;
2. более подробно исследованы алгоритм ближайших соседей и генетический алгоритм;
3. исследована игра 2048, а также некоторые её вариации;
4. сформулирована задача о применении алгоритмов машинного обучения к логической игре 2048, определены основные приёмы её решения;
5. дано обобщение этой задачи на случай любых игр для двух игроков с полной информацией;
6. реализован гибкий настраиваемый сервер для произвольных игр; полностью разработана игра 2048, заложены правила основной игры и предоставлена возможность расширения функционала;
7. реализованы и протестированы некоторые примитивные алгоритмы, представлены соответствующие результаты;
8. реализованы два автоматических игрока в 2048, в основу которых положены алгоритмы машинного обучения – генетический алгоритм и алгоритм ближайших соседей. Исследованы преимущества и недостатки этих алгоритмов в применении к данной задаче.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **gabrielecirulli / 2048.** Исходный код для игры 2048. https://github.com/gabrielecirulli/2048
2. **Викиреальность.** 2048. http://www.wikireality.ru/w/index.php?title=2048&oldid=430808
3. **http://www.machinelearning.ru.** Ресурс MachineLearning.ru
4. **К. В. Воронцов.**  Математические методы обучения по прецедентам (теория обучения машин). Лекции.
5. **С. И. Николенко.** Генетические алгоритмы. Курс лекций.
6. **Н.** **Ю. Золотых.** Машинное обучение. Курс лекций.