ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки бакалавров по направлению   
38.03.05 Бизнес-информатика**

*Тарасов Александр Вячеславович*

**КУРСОВАЯ РАБОТА**

Предсказание результатов баскетбольных матчей на основе нейронных сетей

|  |  |
| --- | --- |
|  | Научный руководитель  Кандидат технических наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  О. Р. Набиуллин |

Нижний Новгород, 2017

Содержание

Введение………………………………………………………..….....................4

Глава 1.Нейронные сети и их использование для проблемы предсказания баскетбольных матчей………………………………………..…....……..........6

1.1Понятие нейронных сетей………………………………….……...……..6

1.2 Пример использования нейронных сетей для предсказания результатов баскетбольных матчей.……… …………………………………………........11

Глава 2. Описание алгоритма нейроэволюции нарастающих топологий……………………………………………………………………12

Глава 3. Создание собственных нейронных сетей. Их тестирование и сравнение……………………………………………………………………..21

3.1 Реализация нейросети с использованием алгоритма нейроэволюции нарастающих топологий.…………………………………………………..21

3.2. Нейронная сеть с результатами последних пяти матчей команд в качестве входных параметров……………………………………………..23

3.3. Нейронная сеть с числом побед команд в сезоне в качестве входных параметров…………………………………………………………………..26

3.4. Нейронная сеть с результатами последних пяти матчей команд и числом побед команд в сезоне в качестве входных параметров………..30

3.5. Нейронная сеть с числом дней отдыха и количеством побед команд в сезоне в качестве входных параметров…………………………………..33

3.6. Сравнение полученных нейронных сетей. Пути улучшения получаемых прогнозов.…………………………………………………..36

Заключение ………………………………….……….…….……………38

Список литературы…………….………………………….......................39

Приложения……………..…………………………………......................40

**Введение**

На сегодняшний день применение нейронных сетей во всем мире очень разнообразно. Оно далеко не ограничивается комьютерным зрением, где эта математическая модель получила, наверное, самое широкое распространение, благодаря широкому спектру возможных дивидендов.

Нейронные сети применяются для разпознавания самых разных объектов при разработке электрокаров, постановки медицинского диагноза, фильтрации спама в почтовых клиентах.

Одним из направлений использования нейронных сетей является и прогнозирование. Например, курсов валют, цен на сырье. Свое развитие начали и разработки нейронных сетей с целью предсказания спортивных событий.

В своей работе я собираюсь применить модель нейронных сетей для предсказания победителей матчей Национальной Баскетбольной Ассоциации (НБА).

Этот спорт и эта лига выбраны не случайно.

Баскетбол является одним из самых «оцифрованных» на данный момент видов спорта. Количество всевозможной статистики, собираемой во время баскетбольных матчей, огромно. Лига НБА же является самой популярной баскетбольной лигой в мире, поэтому статистические показатели прошлого, которые так важны для обучения нейронных сетей, найти для нее намного проще. Более того, команды проводят значительно больше игр, чем в других видах спорта, что увеличивает выборку, упрощая прогноз.

**Цель работы** – исследовать понятие нейронных сетей, получить информацию о результатах баскетбольных матчей, прошедших в выбранной лиге, применить полученные знания для построения собственной нейронной сети для прогнозирования результатов предстоящих матчей. В этой работе для построения нейронной сети я буду использовать алгоритм нейроэволюции нарастающих топологий. Применение данного алгоритма для построения нейронных сетей с целью предсказания результатов баскетбольных матчей ранее не рассматривалось в научных статьях, что говорит о новизне темы. Этот алгоритм позволит не только определить подходящие веса нейронов в сети, но и определить наиболее эффективную архитектуру нейронной сети.

**Задачи работы:**

1. исследовать понятие нейронных сетей;
2. рассмотреть результаты исследований по теме предсказания результатов баскетбольных матчей, полученные ранее в научных работах;
3. реализовать собственную нейронную сеть на основе алгоритма нейроэволюции нарастающих топологий для прогнозирования результатов предстоящих матчей;
4. оценить качество предоставленных прогнозов;
5. сделать выводы, определить пути улучшения модели для повышения качества прогнозов;

**Объектом исследования** является нейронная сеть, прогнозирующая результаты баскетбольных матчей.

**Предмет исследования** – оценка качества прогнозов, факторов влияющих на него.

**Глава 1.Нейронные сети и их использование для проблемы предсказания баскетбольных матчей**

**1.1Понятие нейронных сетей.**

Иску́сственные нейро́нные се́ти (ИНС) — математические модели или программные их имплементации, которые построены в соответствии с методами работы биологических нейронных сетей —связанных между собой нервных клеток живых организмов. Определение впервые появилось при описании процессов, протекающих в мозге человека, и попытках воспроизвести эти процессы в моделях. В дальнейшем, когда были разработаны обучающие алгоритмы, симулирующие работу мозга, такие модели начали использоваться в практических задачах: при управлении организациями, различных прогнозах, для распознавания образов и других.

Искуственная нейронная сеть представляет из себя набор простых процессоров (искусственных нейронов), которые соединены и взаимодействуют между собой. Эти процессоры, как правило, достаточно просты, если сравнивать их с процессорами, которые используются в персональных компьютерах. Любой процессор подобной искуственной сети взаимодействует только посредством сигналов, которые он получает от других, окружающих его процессоров, и сигналами, которые он периодически отправляет другим процессорам. Однако, будучи соединёнными в обширную сеть с настраиваемым взаимодействием, эти отдельно простые нейроны вместе имеют способность выполнять совсем нетривиальные задачи.

Нейронные сети построены не на принципах простых программируемых алгоритмов, они самообучаются. Такая способность обучаться и есть то главное отличие и преимущество ИНС в сравнении с обычными повсеместно применяемыми алгоритмами. Весь процесс обучения технически завязан на изменении весов связей между различными нейронами, которые объединены в сеть. Обучаясь, сеть может выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Именно поэтому, когда обучение проходит успешно, выходные нейроны способны вывести правильный результат, основываясь на новых данных, не предоставленных ей на этапе обучения, а также неполных и/или «зашумленных», частично искаженных данных.

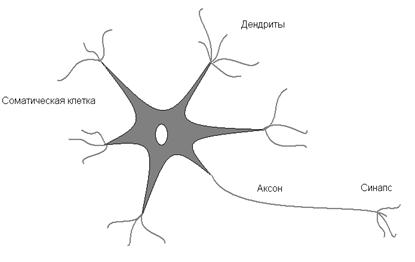


Рис.1 Схематическое изображение биологического нейрона (прообраза нейрона ИНС)

Основу каждой искусственной нейронной сети составляют относительно простые, в большинстве случаев - однотипные, элементы (ячейки), имитирующие работу нейронов мозга (далее под нейроном мы будем подразумевать искусственный нейрон, ячейку искусственной нейронной сети).

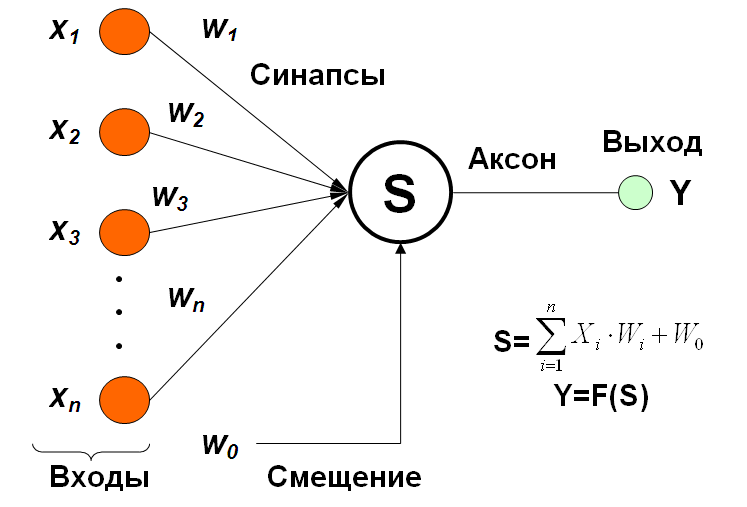


Рис. 2 - Искусственный нейрон

· Нейрон обладает группой синапсов - однонаправленных входных связей, соединенных с выходами других нейронов. Каждый синапс характеризуется величиной синоптической связи или ее весом wi.

· Каждый нейрон имеет текущее состояние, которое обычно определяется, как взвешенная сумма его входов:

· Нейрон имеет аксон - выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на синапсы следующих нейронов. Выход нейрона есть функция его состояния:

y = f(s)

Функция f называется функцией активации.

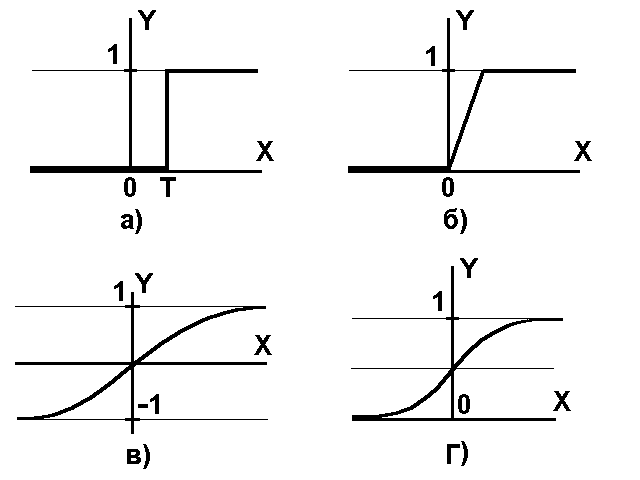


Рис. 3 - Функция активации

Функция активации может иметь разный вид :

· пороговый ( рис. 3.a),

· кусочно-линейный ( рис. 3.б),

· сигмоид( рис. 3.в, 3.г ).

Множество всех нейронов искусственной нейронной сети можно разделить на подмножества - т.н. слои. Взаимодействие нейронов происходит послойно.

Слой искусственной нейронной сети - это множество нейронов на которые в каждый такт времени параллельно поступают сигналы от других нейронов данной сети

Выбор архитектуры искусственной нейронной сети определяется задачей. Для некоторых классов задач уже существуют оптимальные конфигурации. Если же задача не может быть сведена ни к одному из известных классов, разработчику приходится решать задачу синтеза новой конфигурации. Проблема синтеза искусственной нейронной сети сильно зависит от задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант искусственной нейронной сети получается опытным путем.

Искусственные нейронные сети могут быть программного и аппаратного исполнения. Реализация аппаратная обычно представляет собой параллельный вычислитель, состоящий из множества простых процессоров.

**1.2 Пример использования нейронных сетей для предсказания результатов баскетбольных матчей**

## В научной статье Predicting NBA Games Using Neural Networks (Bernard Loeffelholz, Earl Bednar, and Kenneth W. Bauer, 2009) журнала Journal of Quantitative Analysis in Sports проверялось использование нейронных сетей как инструмента для предсказания успеха баскетбольных команд в Национальной Баскетбольной Организации (НБА). Статистика 620 матчей была собрана и использована для тренировки нескольких типов нейронных сетей, таких как нейронные сети прямого распространения (feed-forward), сеть радиально-базисных функций (radial basis), вероятностная неронная сеть (probabilistic), обобщённо-регрессионные нейронные сети (generalized regression neural networks).

Кроме того было исследовано, какой именно набор входных параметров для нейронных сетей обладает наиболее существенными признаками для прогнозирования. Для этого использовалось, как мнение экспертов – прогнозистов, так и степень объяснения признаками итогового результата. Результаты, полученные от нейронных сетей сравнили с предсказаниями нескольких баскетбольных аналитиков.

Лучшие нейронные сети были способны предсказать 74.33 процента результатов верно (в среднем), в то время как эксперты были правы в 68.67 процентах всех случаев. Более того, каждая нейронная сеть из четырех названных выше показала более высокое качество прогнозов, чем баскетбольные эксперты.

**Глава 2. Описание алгоритма нейроэволюции нарастающих топологий**

Описание данного алгоритма впервые было приведено в статье **Efficient Evolution of Neural Network Topologies (**Kenneth O. Stanley and Risto Miikkulainen).

Нейроэволюция – это искуственная эволюция нейронных сетей с использованием генетических алгоритмов.

Алгоритм нейроэволюции нарастающих топологий был разработан, чтобы соответствовать трем главным требованиям.

(1) Создать такое генетическое представление, которое позволит разрозненным топологиям скрещиваться таким образом, чтобы это имело наибольший смысл. Для этого в алгоритме применяется историческая маркировка для выявления генов с одинаковым происхождением.

(2) Найти способ защиты топологических инноваций, которые нуждаются в нескольких поколениях для оптимизации, чтобы не исчезнуть из популяции преждевременно. Для этого в алгоритме каждая инновация выделяется в отдельный вид, чтобы ей не приходилось конкурировать с более оптимизированными топологиями.

(3) Минимизировать топологии во время эволюции без использования специальных функций, измеряющих сложность. Представленное решение состоит в том, что изначально любая структура минимальна и растет только в случае необходимости.

Каждый геном (рис.4) включает в себя список *соединительных генов*, которые ссылаются на два *узловых гена*, которые они соединяют. Каждый соединительный ген включает в себя входной узел, выходной узел, вес соединения, активировано ли данное соединение (бит активации) и номер инновации, который позволяет найти соответствующие гены во время скрещивания (объясняется в дальнейшем).

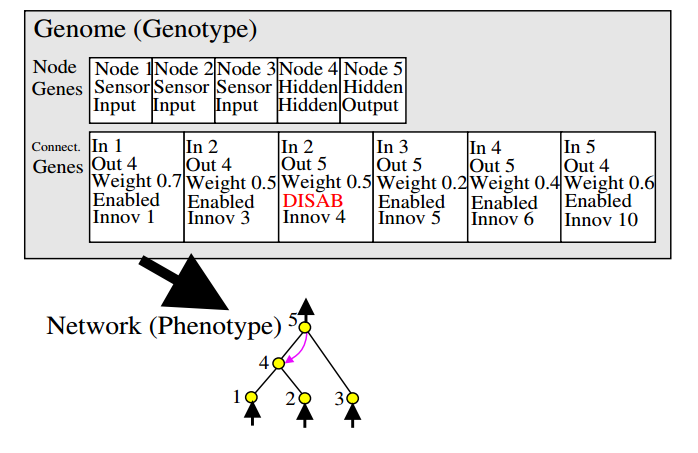


Рис.4 Пример преобразования генотипа (способ записи сети) в фенотип (визуальное представление сети). Третий ген исключен, поэтому соответствующая связь (между узлами 2 и 5) в фенотипе отсутствует.

Мутации согласно алгоритму могут изменять как веса соединений, так и структуру сети. Веса соединений мутируют случайным образом путем генерации действительных чисел. Структурные мутации, расширяющие геном бывают двух видов (рис.5). При мутации добавления связи один новый ген связи добавляется в геном, соединяя два, ранее не соединенных, узла. При мутации добавления узла существующее соединения разбивается, и новый узел помещается вместо старого соединения. Ген старого соединения исключается, два новых соединительных гена добавляются в геном. Такой метод добавления узлов был выбран, чтобы немедленно интегрировать узлы в сеть, но при этом сохранить возможность проследить происхождение узлов.

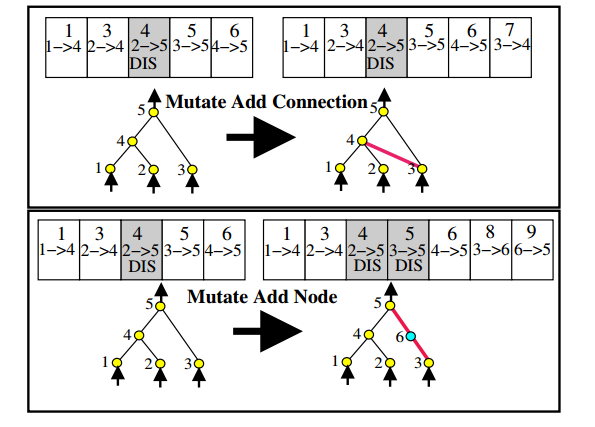


Рис.5. Два вида структурных мутаций в алгоритме нейроэволюции нарастающих топологий. Оба типа, добавления соединения и добавления узла, проиллюстрированы генотипом и фенотипом. Верхнее число каждого гена – это его инновационное число. Эти цифры идентифицируют первоначального исторического предка каждого гена, что позволяет находить совпадающие гены во время скрещивания. Новые гены обозначаются новыми, все более высокими числами.

Посредством мутаций создаются геномы разных размеров, иногда с абсолютно разными соединениями в одинаковых позициях.

Чтобы производить скрещивания, система должна быть способна определить совпадающие гены в популяции генотипов. Ключевое наблюдение состоит в том, что два гена, которые имеют одинаковое историческое происхождение, представляют одну и ту же структуру (возможно, с разными весами), поскольку оба они были получены из одного и того же наследственного гена однажды в прошлом. Таким образом, все, что системе нужно делать, чтобы знать, какие гены соединяются с какими, это проследить историческое происхождение каждого гена в системе.

Отслеживание исторического происхождения требует совсем немного вычислений. Каждый раз, когда появляется новый ген (путем структурных мутаций), глобальный номер инновации инкрементируется и присваивается этому гену. Номер инновации, таким образом, представляет хронологию каждого гена системы. Например, предположим, что две мутации на рис.5 появились одна за другой в системе. Первому соединительному гену, произошедшему во время первой мутации добавления связи, назначается номер инновации 7, и двум новым соединительным генам, добавленным во время следующей мутации добавления узла, назначаются номера 8 и 9В будущем, как бы эти геномы не скрещивались, потомки будут наследовать те же номера инноваций на каждом гене; номера инноваций никогда не меняются. Таким образом, историческое происхождение любого гена системы известно на протяжении всей эволюции.

Историческая маркировка дает алгоритму нейроэволюции нарастающих топологий новую мощную возможность. Система теперь точно знает, какие гены с какими совпадают (рис.6). Гены, которые не совпадают либо дизъюнктивные, либо избыточные, в зависимости от того, появляются они в пределах или вне диапазона других родительских номеров инноваций. Во время скрещивания, гены, которые присутствуют в обоих родителях с одинаковыми номерами инновации, наследуются. Гены, которые не совпадают, наследуются от более подходящего родителя, или, если они одинаково подходят, то от любого родителя случайным образом. Так историческая маркировка дает алгоритму нейроэволюции нарастающих топологий возможность производить скрещивание без сложного топологического анализа.

Представленный метод скрещивания отличается своей простотой. Любые две структуры могут быть объединены вместе без необходимости в каком-либо топологическом анализе.

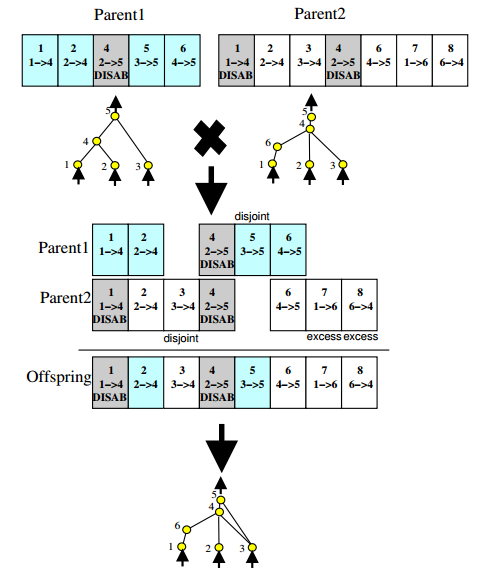


Рис.6. Объединение геномов разных сетевых топологий с использованием номеров инновации. Хоть оба родителя выглядят непохожими, их номера инноваций (обозначенные вверху каждого гена) указывают, какие гены следует объединить.

Добавление в нейронную сеть новых структур, как правило, изначально снижает приспособленность (fitness) сети. Однако алгоритм нейроэволюции нарастающих топологий определяет популяции таким образом, что отдельные генотипы конкурируют в пределах собственных ниш, а не всей популяции в целом. Таким образом, инновации в топологии защищены и имеют время на оптимизацию, прежде чем им придется конкурировать с остальными нишами из популяции.

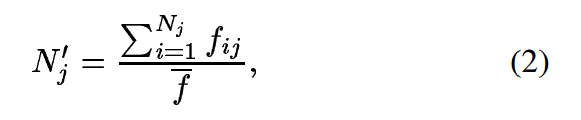
Исторические маркировки позволяют системе разделить популяцию на виды, базируясь на топологической схожести. Количество избыточных и дизъюнктивных генов между парой геномов это естественный показатель уровня их совместимости. Чем более дизъюнктивны два генома, тем менее схожи их эволюционные истории, тем, соответственно, они менее совместимы. Поэтому, мы можем выразить отличие разных структур в алгоритме нейроэволюции нарастающих топологий как простую линейную комбинацию количества их избыточных (E) и дизъюнктивных (D) генов и средней разницы в весах совпадающих генов (Wa):

a  *(1)*

Коэффициенты с1, с2, с3 регулируют важность трех факторов, а N обозначает количество генов в более крупном геноме и нормализует размер генома.

Мера отличия нам разделять геномы по видам, используя пороговое значение t . Геномы тестируются по одному за раз. Если мера отличия генома со случайно выбранным представителем вида меньше t, то геном помещается в этот же вид. Каждый геном помещается в первый вид, где условие попадания выполняется, поэтому каждый геном не может принадлежать более, чем одному виду.

В качестве механизма репродукции в алгоритме используется *явное разделение приспособленности*, то есть организмы одного вида должны быть схожи по уровню приспособленности в своей нише, что затруднит захват всей популяции одним отдельным видом. Поэтому исходные уровни приспособленности корректируются делением на число организмов в виде. Виды со временем расширяются или сокращаются в зависимости от того, выше их скорректированная приспособленность или ниже, чем средняя приспособленность всей популяции:



, где Nj и N’j это старое и новое количество организмов в виде j, fij это скорректированная приспособленность генома i из вида j, а это средняя скорректированная приспособленность всей популяции. Заранее выбранный процент представителей каждого вида скрещиваются случайным образом, чтобы сгенерировать N’j число потомков, которые заменяют собой весь вид.

В редких случаях, когда приспособленность всей популяции не растет на протяжении более чем 20 поколений, только два лучших вида производят репродукцию, тем самым поиск оптимальной структуры фокусируется в самых многообещающих местах.

Алгоритм начинает с однородной совокупности сетей без скрытых слоев. Благодаря разделению на виды и, как следствие, защите инноваций, первоначальные сети могут быть минимально возможными, а обрастать новыми структурами только по необходимости. Усложнение структуры происходит постепенно в результате мутаций, а выживают эти новые структуры только в том случае, если показывают свою полезность, повышая приспособленность сети. Таким образом, алгоритм нейроэволюции нарастающих топологий производит поиск наиболее подходящей сети среди минимально возможных структур, что очень значительно снижает число поколений необходимых для нахождения работоспособного решения.

**Глава 3. Создание собственных нейронных сетей. Их тестирование и сравнение.**

**3.1 Реализация нейросети с использованием алгоритма нейроэволюции нарастающих топологий.**

Реализован алгоритм был на языке Python с использованием библиотеки NEAT-Python.

Библиотека реализовывает симуляцию такой системы, что каждый геном состоит из двух видов генов, который описывают систему построения нейронной сети:

Узловые гены, каждый из которых обозначает одиночный нейрон.

Соединительные гены, каждый из которых обозначает связь между двумя нейронами.

Для того, чтобы начать эволюцию нейросети, которая призвана решать определенную задачу, нужно было предоставить функцию приспособленности (fitness function), с помощью которой можно вычислить одно действительное число, измеряющее качество генома: лучшая способность решить рпоблему должна вести к более высокому показателю функции приспособленности. Алгоритм выполняется предустановленное количество раз (поколений), генерируя новые поколения путем мутаций и скрещивания наиболее подходящих геномов предыдущего поколения. Алгоритм также может остановиться по нахождению генома с нужным пороговым значением функции приспособленности.

В качестве выходного (output) параметра от нейронной сети ожидается победитель баскетбольного матча. В случае победы домашней команды – 1. В случае победы гостевой – 0. Иного варианта в НБА произойти не может, так как победитель каждого матча в случае равенства набранных очков в основное время в конце концов выявляется серией овертаймов.

В качестве входных (input) параметров нейронная сеть получает результаты (разницу в счете) пяти последних игр команд, исход матча между которыми требуется предсказать. Вторым вариантом была передача в качестве входных параметров текущего количества побед команд. Таким образом, первый вариант нейросети пытается определить текущую силу команд, а второй сможет делать вывод, принимая в расчет среднесезонные результаты команды. Очевидно, что команды, находящиеся в хорошей или плохой форме, могут демонстрировать результаты отличные от среднесезонных.

Функция приспособленности была определена следующим образом: из числа матчей в выборке вычитается количество матчей, предсказанных нейронной сетью неправильно, полученное число делится на число матчей в выборке. Таким образом, функция приспособленности реально показывает среднюю правильность (accuracy) прогноза победителя матча.

Результаты прошедших баскетбольных матчей за сезоны c 2010-2011 по 2014-2015 года были взяты с сайта http://www.basketball-reference.com/ . Матчи сезона 2010-2011 были использованы для обучения нейронной сети.

**3.2. Нейронная сеть с результатами последних пяти матчей команд в качестве входных параметров**

На обучающей выборке первым вариантом нейронной сети (входные параметры: результаты пяти последних матчей) была достигнута правильность (accuracy) прогноза победителя равная 66%.

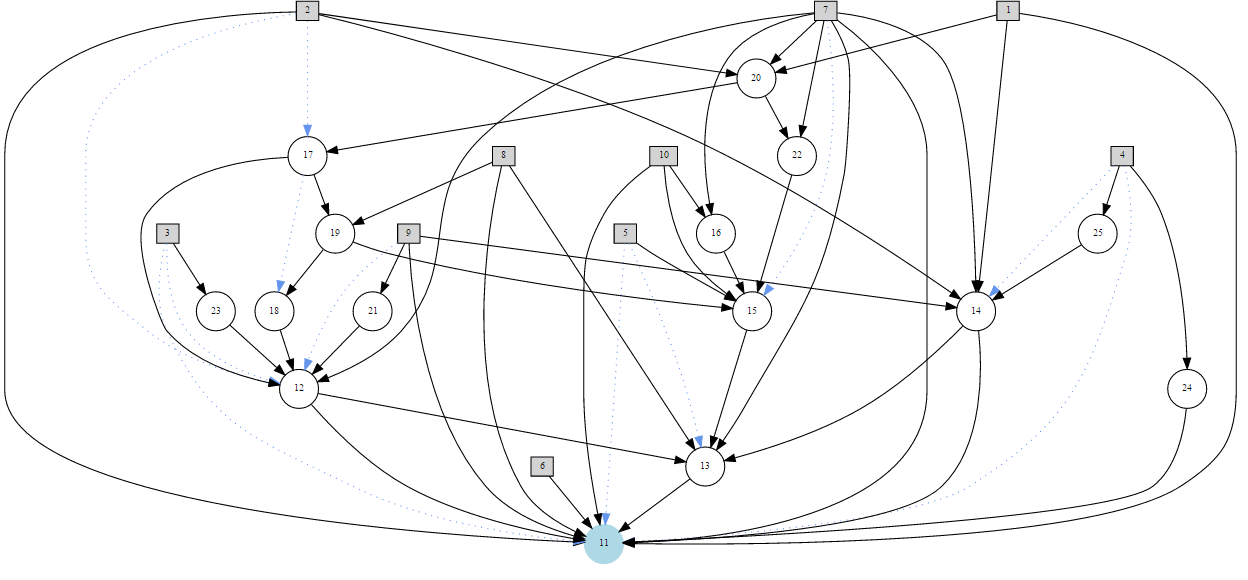


Рисунок 6. Результат нейроэволюции нейронной сети с результатами последних пяти матчей в качестве входных параметров.

На рис. 6 изображена структура нейронной сети с высшей приспособленностью для первого варианта входных параметров.

В качестве контрольных выборок использовались матчи сезонов с 2011-2012 по 2014-2015 года.

Результаты предсказаний по сезонам первого варианта нейросети:

2011-2012

Правильность: 60%

Точность (precision): 65%

Полнота (recall): 67.6%

Процент предсказаний в пользу домашней команды: 58%

Процент побед домашней команды в действительности: 60%

2012-2013

Правильность: 61%

Точность (precision): 69,1%

Полнота (recall): 66%

Процент предсказаний в пользу домашней команды: 59%

Процент побед домашней команды в действительности: 61%

2013-2014

Правильность: 62%

Точность (precision): 66%

Полнота (recall): 67%

Процент предсказаний в пользу домашней команды: 58%

Процент побед домашней команды в действительности: 57%

2014-2015

Правильность: 61%

Точность (precision): 65%

Полнота (recall): 67%

Процент предсказаний в пользу домашней команды: 58%

Процент побед домашней команды в действительности: 57%

Средняя правильность прогнозов составила 61%. Важно отметить, что правильность в различных сезонах отличалась не более чем на 1.5%, что означает, что нейронная сеть, обученная на результатах одного сезона, сохраняет качество прогнозов на несколько будущих сезонов, не требуя скорого переобучения.

Нейронная сеть в 58% случаев предсказывает победу домашней команды. В действительности команда, играя дома, добивается победы в 57 - 58% случаев. Видно, что сеть правильно оценивает важность фактора домашней площадки. Об этом также говорят и очень близкие значения характеристик точности (precision) и полноты (recall) (65% и 66% в среднем соответственно).

**3.3. Нейронная сеть с числом побед команд в сезоне в качестве входных параметров**

На обучающей выборке вторым вариантом нейронной сети (входные параметры: количество побед команды) была достигнута правильность (accuracy) прогноза победителя равная 70%.

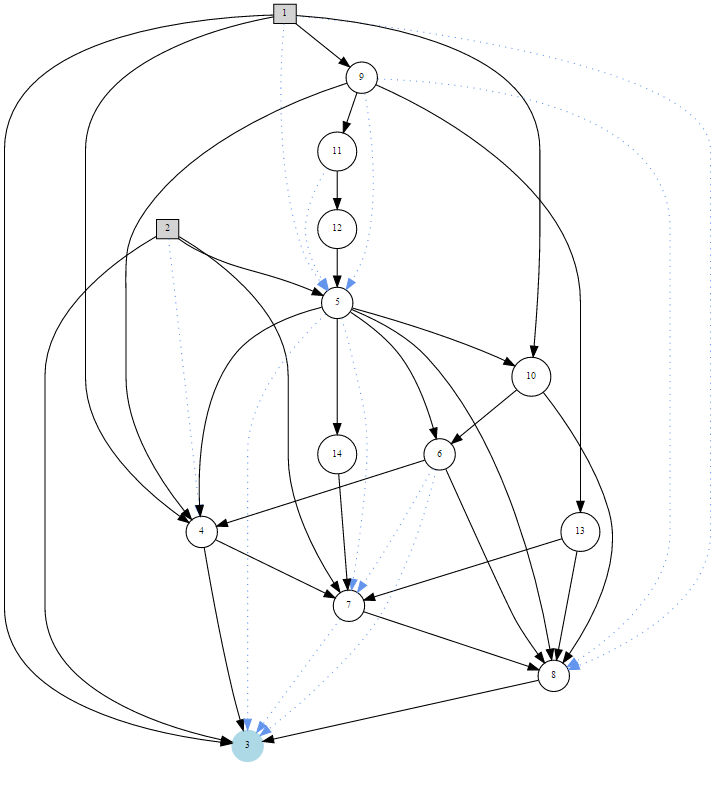


Рисунок 7. Результат нейроэволюции нейронной сети с количеством побед команд в сезоне в качестве входных параметров.

На рис. 7 изображена структура нейронной сети с высшей приспособленностью для второго варианта входных параметров.

В качестве контрольных выборок также использовались матчи сезонов с 2011-2012 по 2014-2015 года.

Результаты предсказаний по сезонам второго варианта нейросети:

2011-2012

Правильность: 66%

Точность (precision): 66%

Полнота (recall): 83%

Процент предсказаний в пользу домашней команды: 73%

Процент побед домашней команды в действительности: 58%

2012-2013

Правильность: 68%

Точность (precision): 70%

Полнота (recall): 82%

Процент предсказаний в пользу домашней команды: 72%

Процент побед домашней команды в действительности: 61%

2013-2014

Правильность: 65%

Точность (precision): 65%

Полнота (recall): 82%

Процент предсказаний в пользу домашней команды: 72%

Процент побед домашней команды в действительности: 57%

2014-2015

Правильность: 68%

Точность (precision): 67.5%

Полнота (recall): 84%

Процент предсказаний в пользу домашней команды: 71%

Процент побед домашней команды в действительности: 57%

Средняя правильность прогнозов составила 67%. Второй вариант смог достичь более высокого показателя правильности прогнозов. Однако, он сильно переоценил фактор домашней площадки. Для того, чтобы скорректировать значительность фактора домашней площадки было решено в качестве третьего варианта входных параметров объединить входные параметры предыдущих двух вариантов.

**3.4. Нейронная сеть с результатами последних пяти матчей команд и числом побед команд в сезоне в качестве входных параметров**

Таким образом, третий вариант входных параметров – это результаты последних пяти матчей и количество побед команды в сезоне.

На обучающей выборке третьим вариантом нейронной сети (входные параметры: количество побед команд и результаты последних пяти матчей команд) была достигнута правильность (accuracy) прогноза победителя равная 70%.

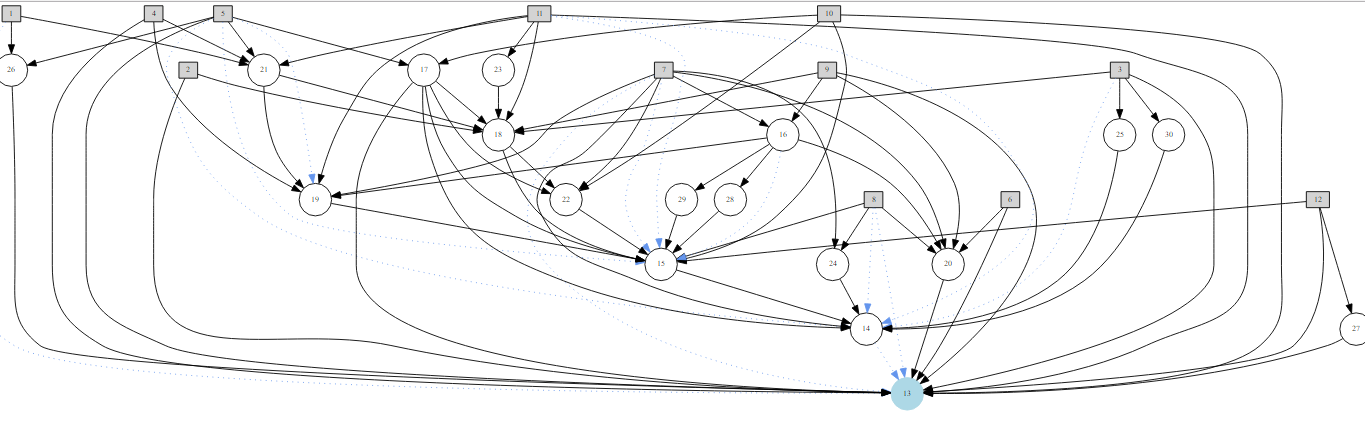


Рисунок 8. Результат нейроэволюции нейронной сети с количеством побед команд в сезоне и результатами последних 5 матчей команд в качестве входных параметров.

На рис. 8 изображена структура нейронной сети с высшей приспособленностью для третьего варианта входных параметров.

В качестве контрольных выборок также использовались матчи сезонов с 2011-2012 по 2014-2015 года.

Результаты предсказаний по сезонам третьего варианта нейросети:

2011-2012

Правильность: 61%

Точность (precision): 63%

Полнота (recall): 74%

Процент предсказаний в пользу домашней команды: 67%

Процент побед домашней команды в действительности: 58%

2012-2013

Правильность: 65%

Точность (precision): 69%

Полнота (recall): 77%

Процент предсказаний в пользу домашней команды: 68%

Процент побед домашней команды в действительности: 61%

2013-2014

Правильность: 64%

Точность (precision): 65%

Полнота (recall): 78%

Процент предсказаний в пользу домашней команды: 68%

Процент побед домашней команды в действительности: 57%

2014-2015

Правильность: 67%

Точность (precision): 68%

Полнота (recall): 80%

Процент предсказаний в пользу домашней команды: 67%

Процент побед домашней команды в действительности: 57%

Средняя правильность прогнозов составила 64%. Третий вариант показал более низкий показатель правильности предсказаний. Более того, он лишь отчасти справился с задачей корректировки значимости фактора домашней площадки.

В дальнейшем было решено к входным параметрам, которые показали лучшую правильность прогнозов, добавить число дней отдыха команд перед матчем для того, чтобы учесть уровень усталости команды в случаях, когда команде приходится играть третью игру за 2 дня, что очень важно при плотном календаре матчей НБА.

**3.5. Нейронная сеть с числом дней отдыха и количеством побед команд в сезоне в качестве входных параметров**

На обучающей выборке четвертым вариантом нейронной сети (входные параметры: количество побед команд и число дней отдыха) была достигнута правильность (accuracy) прогноза победителя равная 70%.

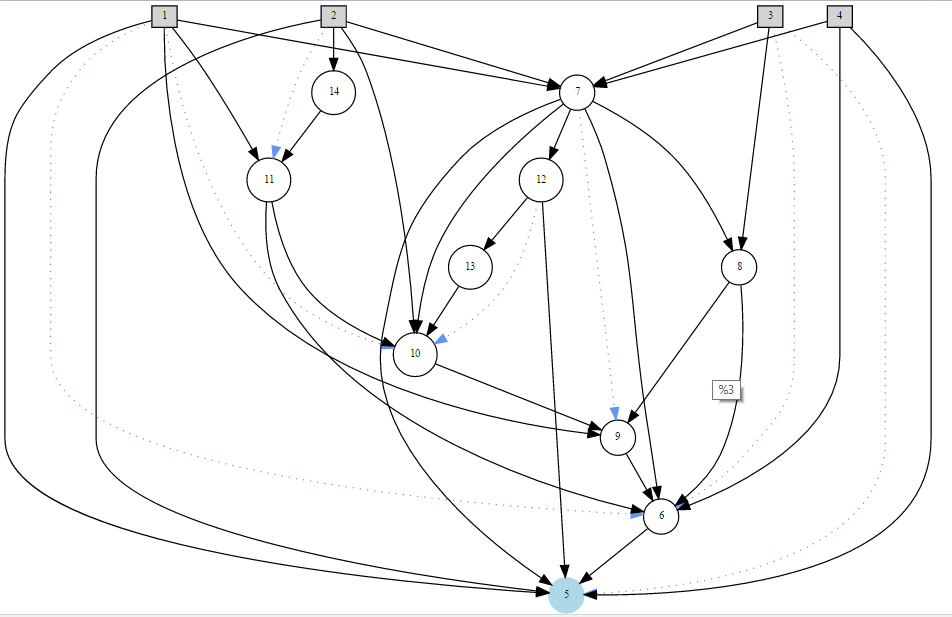


Рисунок 9. Результат нейроэволюции нейронной сети с количеством побед команд в сезоне и количеством дней отдыха перед матчем команд в качестве входных параметров.

На рис. 9 изображена структура нейронной сети с высшей приспособленностью для четвертого варианта входных параметров.

В качестве контрольных выборок также использовались матчи сезонов с 2011-2012 по 2014-2015 года.

Результаты предсказаний по сезонам третьего варианта нейросети:

2011-2012

Правильность: 66%

Точность (precision): 66%

Полнота (recall): 83%

Процент предсказаний в пользу домашней команды: 73%

Процент побед домашней команды в действительности: 58%

2012-2013

Правильность: 65%

Точность (precision): 69%

Полнота (recall): 78%

Процент предсказаний в пользу домашней команды: 69%

Процент побед домашней команды в действительности: 61%

2013-2014

Правильность: 65%

Точность (precision): 66%

Полнота (recall): 80%

Процент предсказаний в пользу домашней команды: 70%

Процент побед домашней команды в действительности: 57%

2014-2015

Правильность: 66%

Точность (precision): 68%

Полнота (recall): 80%

Процент предсказаний в пользу домашней команды: 67%

Процент побед домашней команды в действительности: 57%

Видно, что добавление количества дней отдыха команд перед матчем ко входным параметрам привело нейронную сеть к корректировке значимости фактора домашней площадки, что впрочем не изменило правильность прогнозов.

**3.6. Сравнение полученных нейронных сетей. Пути улучшения получаемых прогнозов.**

Параметр «Ошибка в оценке фактора домашней площадки» означает разницу между процентом предсказаний победы домашней команды и процентом побед, которые реально одержали домашние команды

Таблица 1. Сравнение полученных нейронных сетей

|  |  |  |
| --- | --- | --- |
| Входные параметры | Правильность прогнозов | Ошибка в оценке фактора домашней площадки |
| Результаты последних пяти матчей команд | 61% | +0,5% |
| Количество побед команд в сезоне | 67% | +14% |
| Результаты последних пяти матчей команд и количество побед команд в сезоне | 64% | +10% |
| Количество побед команд в сезоне и число дней отдыха команд перед матчем | 67% | +11% |

В целом, наиболее предпочтительным вариантом можно считать последний вариант нейронной сети с выходными параметрами: количество побед команд в сезоне и число дней отдыха команд перед матчем. Он обладает наивысшим показателем правильности прогнозов наравне с вариантом входных параметров с количеством побед команды в сезоне, однако лучше второго оценивает значимость фактора домашней площадки.

Правильность полученных в результате прогнозов лучшими вариантами нейронных сетей составила в среднем 67%, что даже выше средней правильности прогнозов экспертов. Например, команда прогнозистов сайта espn.com верно предсказывает победителей матчей НБА в 65% случаев. Очевидно, что предоставленных данных нейронной сети недостаточно для достижения более высокой правильности прогнозов. Выбранные схемы входных значений неплохо показывают себя, но обладают и рядом недостатков. Схема дает неправильное представление о силе команды в случаях травм важных игроков, переходов ключевых игроков в другие команды в результате обменов. Не учитывается возможное изменение уровня подготовки команды после длительного перерыва в середине сезона.

Схема входных параметров в будущем требует изменения с целью учесть вышеперечисленные факторы. Но было замечено, что при увеличении входных параметров выбранный алгоритм построения нейронной сети склонен к ускорению разрастания структуры нейронной сети. Как итог, разросшейся структуре требуется дополнительное время на оптимизацию, что занимает очень длительное время.

Поэтому в дальнейшем планируется выбрать другой более подходящий к данной тематике алгоритм для построения и оптимизации нейронной сети.

**4.Заключение**

В ходе работы исследовано понятие нейронных сетей,

рассмотрены результаты исследований по теме предсказания результатов баскетбольных матчей, полученные ранее в научных работах;

реализованы собственные нейронные сети на основе алгоритма нейроэволюции нарастающих топологий для прогнозирования результатов предстоящих матчей,

оценено качество предоставленных прогнозов,

сделаны выводы, определить пути улучшения модели для повышения качества прогнозов.

В ходе работы написано более 1100 строк кода. Среднее время получения оптимальных структур нейронных сетей в среднем составляло 8-10 минут на компьютере с двухъядерным процессором Intel Core i5-5200U и 4 Гб оперативной памяти.

Задачи работы считаю выполненными, а цели достигнутыми.

Важно отметить, что результаты полученные в рамках использования этого алгоритма имеют большое прикладное значение, ведь правильность прогнозов сети превысила правильность прогнозов экспертов, поэтому полученные нейронные сети могут стать очень полезны в ставках на матчи НБА. Оценка прибыльности ставок по прогнозам полученных нейронных сетей в рамках данной работы не проводилась.

*Список литературы*

[1]. Kenneth O. Stanley and Risto Miikkulainen, (2002). Efficient Evolution of Neural Network Topologies

[2]. Bernard Loeffelholz, Earl Bednar, and Kenneth W. Bauer, (2009). Predicting NBA Games Using Neural Networks

[3].NEAT-Python’s documentation. [Электронный ресурс] - режим доступа: http://neat-python.readthedocs.io/en/latest/

[4]. Курс «Машинное обучение» от Стэнфордского университета. [Электронный ресурс] - режим доступа: https://www.coursera.org/learn/machine-learning/home/welcome

[5].Что такое искусственные нейронные сети? [Электронный ресурс].- режим доступа: <https://habrahabr.ru/post/134998/>

**Приложение 1. Классы Game и Team.**

**class** Game:  
 **def** \_\_init\_\_(self, team1, team2, score1, score2, when):  
 self.home\_name = team1  
 self.away\_name = team2  
 self.diff = score1 - score2  
 self.home\_sc=score1  
 self.away\_sc=score2  
 self.date = when  
  
 **def** g\_date(self):  
 **return** self.date  
  
 **def** home(self):  
 **return** self.home\_name  
  
 **def** away(self):  
 **return** self.away\_name  
   
 **def** result(self):  
 **return** self.diff  
  
 **def** home\_score(self):  
 **return** self.home\_sc  
  
 **def** away\_score(self):  
 **return** self.away\_sc  
  
  
  
**class** Team:  
 **def** \_\_init\_\_(self, s):  
 self.stats = s  
  
 **def** team\_stats(self):  
 **return** self.stats

**Приложение 2. Класс Scraper (Импорт в программу данных по командам и матчам)**

**import** stats  
**import** csv  
**from** stats **import** Game, Team  
**import** datetime  
  
**class** Scraper:  
 **def** get\_teams\_dict(self):  
 **with** open(**'teams\_2015.csv'**) **as** csvfile:  
 reader = csv.DictReader(csvfile)  
 team\_names=[]  
 **for** row **in** reader:  
 **try**:  
 team\_names.append(row[**"Team"**].replace(**"\*"**,**""**))  
 **except** ValueError:  
 **break** team\_names.append(**'Charlotte Bobcats'**)  
 team\_names.append(**'New Jersey Nets'**)  
 team\_names.append(**'New Orleans Hornets'**)  
 teams\_dictionary = dict.fromkeys(team\_names)  
 **for** team\_name **in** teams\_dictionary.keys():  
 teams\_dictionary.\_\_setitem\_\_(team\_name, [0,0])  
 **return** teams\_dictionary  
  
 **def** get\_games(self):  
 format = **"%a %b %d %Y"  
 with** open(**'games13.csv'**) **as** csvfile:  
 reader = csv.DictReader(csvfile)  
 my\_games = []  
 **for** row **in** reader:  
 **try**:  
 my\_games.append(  
 Game(row[**'Home/Neutral'**], row[**'Visitor/Neutral'**], int(row[**'PTSH'**]), int(row[**'PTSA'**]), datetime.datetime.strptime(row[**'Date'**], format)))  
 **except** ValueError:  
 **break  
 return** my\_games  
  
 **def** get\_teams\_rest(self, num\_of\_game):  
 games = self.get\_games()  
 result = []  
 **for** i **in** range(len(games)):  
 *# print 'i=', i* **if** i == num\_of\_game:  
 home\_team = games[i].home()  
 away\_team = games[i].away()  
 c=0  
 *# print away\_team, ' ', home\_team* **for** j **in** range(i - 1, 0, -1):  
 *# print 'j1=', j* cur\_game = games[j]  
 **if** cur\_game.home() == home\_team:  
 c = c + 1  
 result.append((games[i].g\_date()-cur\_game.g\_date()).days)  
 **if** cur\_game.away() == home\_team:  
 c = c + 1  
 result.append((games[i].g\_date() - cur\_game.g\_date()).days)  
 **if** c == 1:  
 *# print 'gonna break j1'* **break  
 else**:  
 **continue  
 for** k **in** range(i - 1, 0, -1):  
 *# print 'j2=', k* cur\_game = games[k]  
 **if** cur\_game.home() == away\_team:  
 c = c + 1  
 result.append((games[i].g\_date() - cur\_game.g\_date()).days)  
 **if** cur\_game.away() == away\_team:  
 c = c + 1  
 result.append((games[i].g\_date() - cur\_game.g\_date()).days)  
 **if** c == 2:  
 *# print 'gonna break j2'* **break  
 else**:  
 **continue  
 break** *# print 'rest done with ', num\_of\_game* **return** result  
  
 **def** get\_teams\_shape(self, num\_of\_game):  
 games = self.get\_games()  
 result = []  
 **for** i **in** range(len(games)):  
 *#print 'i=', i* **if** i == num\_of\_game:  
 home\_team = games[i].home()  
 away\_team = games[i].away()  
 *#print away\_team, ' ', home\_team* c=0  
 **for** j **in** range(i - 1, 0, -1):  
 *#print 'j1=', j* cur\_game=games[j]  
 **if** cur\_game.home() == home\_team:  
 c = c + 1  
 result.append(cur\_game.home\_score()-cur\_game.away\_score())  
 **if** cur\_game.away() == home\_team:  
 c = c + 1  
 result.append(cur\_game.away\_score() - cur\_game.home\_score())  
 **if** c == 5:  
 *#print 'gonna break j1'* **break  
 else**:  
 **continue  
 for** k **in** range(i - 1, 0, -1):  
 *#print 'j2=', k* cur\_game=games[k]  
 **if** cur\_game.home() == away\_team:  
 c = c + 1  
 result.append(cur\_game.home\_score() - cur\_game.away\_score())  
 **if** cur\_game.away() == away\_team:  
 c = c + 1  
 result.append(cur\_game.away\_score()-cur\_game.home\_score())  
 **if** c == 10:  
 *#print 'gonna break j2'* **break  
 else**:  
 **continue  
 break  
 print 'done with '**,num\_of\_game  
 **return** result

**Приложение 3. Класс Shapes (калькуляция необходимой статистики на основе информации по матчам)**

**import** csv  
**import** scraper  
**import** cPickle **as** pickle  
**from** scraper **import** Scraper  
**import** copy  
**from** copy **import** deepcopy  
  
stats\_scraper = Scraper()  
games = stats\_scraper.get\_games()  
teams\_dictionary = stats\_scraper.get\_teams\_dict()  
i = 0  
shapes = []  
tables = []  
rests = []  
*# print teams\_dictionary***for** game **in** games:  
 teams\_dictionary.\_\_setitem\_\_(game.home(), [teams\_dictionary.\_\_getitem\_\_(game.home())[0] + (game.result() > 0),  
 teams\_dictionary.\_\_getitem\_\_(game.home())[1] + (game.result() < 0)])  
 teams\_dictionary.\_\_setitem\_\_(game.away(), [teams\_dictionary.\_\_getitem\_\_(game.away())[0] + (game.result() < 0),  
 teams\_dictionary.\_\_getitem\_\_(game.away())[1] + (game.result() > 0)])  
 tables.append(deepcopy(teams\_dictionary))  
 **if** i >= 100:  
 shape = stats\_scraper.get\_teams\_shape(i)  
 shapes.append(shape)  
 rest = stats\_scraper.get\_teams\_rest(i)  
 rests.append(rest)  
 **else**:  
 rests.append([0,0])  
 shapes.append([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])  
 i = i + 1  
file = open(**'shapes'**, **'w'**)  
pickle.dump(shapes, file)  
file.close()  
file = open(**'tables'**, **'w'**)  
pickle.dump(tables, file)  
file.close()  
file = open(**'rests'**, **'w'**)  
pickle.dump(rests, file)  
file.close()  
**print "saved"***#print tables  
#print rests*

**Приложение 4. Обучение нейронной сети.**

**import** math  
**import** random  
**import** cPickle **as** pickle  
  
**import** time  
**import** platform  
**import** traceback  
**import** sys  
**import** copy  
  
**from** scraper **import** Scraper  
  
**from** neat **import** config, population, chromosome, genome  
**from** neat.nn **import** nn\_pure **as** nn  
  
config.load(**'nba\_predict\_config\_all\_rest'**)  
  
*# set node gene type*chromosome.node\_gene\_type = genome.NodeGene  
  
*# scrape web for training set statistics*stats\_scraper = Scraper()  
games = stats\_scraper.get\_games()  
  
file = open(**"tables"**)  
tables = pickle.load(file)  
file.close()  
file = open(**"shapes"**)  
shapes = pickle.load(file)  
file.close()  
  
file = open(**"rests"**)  
rests = pickle.load(file)  
file.close()  
  
MAX\_FITNESS = len(games)-100  
  
**def** eval\_fitness(population):  
 **for** chromo **in** population:  
 net = nn.create\_ffphenotype(chromo)  
  
 fitness = MAX\_FITNESS  
 possible\_fitness = 0  
 num\_correct = 0  
 **for** i **in** range(100,len(games)):  
 *# print i* inputs = []  
 game = games[i]  
 *# # print shapes[i]* **for** shape **in** shapes[i]:  
 inputs.append(shape)  
 inputs.append(tables[i-1].get(game.home())[0])  
 inputs.append(tables[i-1].get(game.away())[0])  
 **for** rest **in** rests[i]:  
 inputs.append(rest)  
 *# print inputs* outputs = net.sactivate(inputs)  
  
 home\_win\_prob = outputs[0]  
 margin = outputs[0]  
  
 **if**((game.result()>0)!=(margin>0)):  
 fitness -= 1  
  
 chromo.fitness = fitness / float(MAX\_FITNESS)  
 *#print "A fitness was " + str(chromo.fitness)*population.Population.evaluate = eval\_fitness  
  
pop = population.Population()  
pop.epoch(eval\_fitness, 100, report=**True**, save\_best=**False**)  
  
winner = pop.population[0]  
**print 'Number of evaluations: %d'** %winner.id  
  
*# Visualize the winner network (requires PyDot)  
#visualize.draw\_net(winner) # best chromosome  
  
# Plots the evolution of the best/average fitness (requires Biggles)  
#visualize.plot\_stats(pop.stats)  
# Visualizes speciation  
#visualize.plot\_species(pop.species\_log)***print '\nBest network output:'**brain = nn.create\_ffphenotype(winner)  
**print** brain.neurons  
**print** brain.synapses  
  
*# saves the winner*file = open(**'winner\_chromosome\_all\_rest'**, **'w'**)  
pickle.dump(winner, file)  
file.close()

**Приложение 5. Тестирование нейронной сети**

**import** random  
**import** cPickle **as** pickle  
**import** math  
  
**from** scraper **import** Scraper  
**from** neat **import** visualize  
**from** neat.nn **import** nn\_pure **as** nn  
  
stats\_scraper = Scraper()  
games = stats\_scraper.get\_games()  
  
file = open(**"winner\_chromosome\_all\_rest"**)  
chromo = pickle.load(file)  
best\_net = nn.create\_ffphenotype(chromo)  
file.close()  
  
visualize.draw\_net(chromo)  
  
file = open(**"rests"**)  
rests = pickle.load(file)  
file.close()  
  
file = open(**"shapes"**)  
shapes = pickle.load(file)  
file.close()  
  
file = open(**"tables"**)  
tables = pickle.load(file)  
file.close()  
  
j = 0  
s = float(0)  
tp = float(0)  
fp = float(0)  
fn = float(0)  
hw = float(0)  
rhw = float(0)  
pred\_start = 110  
**for** i **in** range(pred\_start, len(games)):  
 j = j + 1  
 **print** j  
 game = games[i]  
 home\_team = game.home()  
 **print** home\_team  
 away\_team = game.away()  
 **print** away\_team  
 inputs = []  
 **for** shape **in** shapes[i]:  
 inputs.append(shape)  
 inputs.append(tables[i - 1].get(game.home())[0])  
 inputs.append(tables[i - 1].get(game.away())[0])  
 **for** rest **in** rests[i]:  
 inputs.append(rest)  
 outcome\_prediction = best\_net.sactivate(inputs)[0]  
 mov = float(best\_net.sactivate(inputs)[0])  
 total\_err = 0  
 **print "Probability home team wins: "** + str(outcome\_prediction)  
 **print "Margin of victory: "**, mov  
 **if** (mov > float(0)):  
 **if** (game.result() > 0):  
 **print "SUCCESS"** s = s + 1  
 tp = tp + 1  
 rhw=rhw+1  
 **if** (game.result() < 0):  
 fp = fp + 1  
 **print "FAIL"** hw=hw+1  
 **else**:  
 **if** (game.result() < 0):  
 **print "SUCCESS"** s = s + 1  
 **if** (game.result() > 0):  
 fn = fn + 1  
 rhw=rhw+1  
 **print "FAIL"**spred = float(s/(len(games) - pred\_start))  
**print "successful predicted:"**, spred  
**print** tp,fp,fn  
P=float(tp/(tp+fp))  
R=float(tp/(tp+fn))  
F=2\*P\*R/(P+R)  
**print 'precision = '**,P  
**print 'recall = '**,R  
**print 'F-measure = '**, F  
**print 'Home win predicted '**,float(hw/(len(games) - pred\_start))  
**print 'Real home wins '**, float(rhw/(len(games) - pred\_start))