СОДЕРЖАНИЕ

Введение

Искусственные нейронные сети представляют собой устройства параллельных вычислений, состоящие из множества взаимодействующих простых процессоров. Такие процессоры обычно исключительно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам, и, тем не менее, будучи соединенными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

Разработка искусственных нейронных сетей началась еще на заре XX столетия, но только в 90-х годах, когда были преодолены некоторые теоретические барьеры, а вычислительные системы стали достаточно мощными, нейронные сети получили широкое признание. Слово "искусственные" в данном контексте иногда используется для того, чтобы подчеркнуть, что речь идет об искусственном устройстве, а не о реальных биологических нейронных системах типа той, которую имеет человек. Создание искусственных нейронных сетей было инспирировано попытками понять принципы работы человеческого мозга и, без сомнения, это будет влиять и на дальнейшее их развитие. Однако, в сравнении с человеческим мозгом, искусственные нейронные сети сегодня представляют собой весьма упрощенные абстракции. Когда ясно, в каком контексте обсуждаются эти сети, слово "искусственный" обычно опускают. Кроме того, когда требуется подчеркнуть вычислительные возможности, а не биологическое соответствие, искусственные нейронные сети называют коннекциями. При этом целью "коннекционистов" является наделение нейронной сети возможностью решать конкретные задачи, а не имитировать с максимальной точностью биологический процесс.

Хотя нейронные сети могут быть реализованы в виде быстрых аппаратных устройств (и такие реализации действительно существуют), большинство исследований выполняется с использованием программного моделирования на обычных компьютерах. Программное моделирование обеспечивает достаточно дешевую и гибкую среду для поиска и проверки исследовательских идей, а для многих реальных приложений такое моделирование оказывается вполне адекватным и достаточным. Например, программная реализация нейронной сети может использоваться для составления плана кредитных выплат индивидуума, обращающегося в банк за займом. Хотя решение на основе нейронной сети может выглядеть и вести себя как обычное программное обеспечение, они различны в принципе, поскольку большинство реализаций на основе нейронных сетей "обучается", а не программируется: сеть учится выполнять задачу, а не программируется непосредственно. На самом деле в большинстве случаев нейронные сети используются тогда, когда невозможно написать подходящую программу, или по причине того, что найденное нейронной сетью решение оказывается более совершенным. Например, будучи экспертом по продаже недвижимости, вы можете из своего опыта прекрасно знать, какие факторы влияют на продажную цену каждого конкретного дома, но при этом часто имеются такие особенности, которые будет весьма трудно объяснить программисту. Агентство по продаже недвижимости может пожелать иметь "предсказателя цен на основе нейронной сети", обученного на множестве примеров реальных продаж тому, какие факторы влияют на цену продаваемого объекта, и тому, какую относительную важность имеет каждый из этих факторов. Но здесь более важным оказывается то, что решение на основе нейронной сети является более гибким, поскольку соответствующая система может в дальнейшем совершенствовать точность предсказаний по мере накопления ею опыта и адаптироваться к происходящим на рынке изменениям.

Решения на основе нейронных сетей становятся все более совершенными и, несомненно, в будущем наши возможности по разработке соответствующих устройств возрастут за счет лучшего понимания их основополагающих принципов. Но уже сегодня имеется немало впечатляющих разработок. База приложений нейронных сетей просто огромна: выявление фальшивых кредитных карточек, прогнозирование изменений на фондовой бирже, составление кредитных планов, оптическое распознавание символов, профилактика и диагностика заболеваний человека, наблюдение за техническим состоянием машин и механизмов, автоматическое управление движением автомобиля, принятие решений при посадке поврежденного летательного аппарата и т.д. Дальнейшие успехи в разработке искусственных нейронных сетей будут зависеть от дальнейшего понимания принципов работы человеческого мозга, но здесь имеется и обратная связь: искусственные нейронные сети являются одним из средств, с помощью которых совершенствуется наше представление о процессах, происходящих в нервной системе человека, выступая в качестве моделей соответствующих процессов.

Будущее нейронных сетей кажется вполне ясным, и сегодня это та область знаний, о которой должны иметь определенное представление все научные специалисты, работающие в области компьютерных технологий, равно как и многие инженеры и научные работники смежных специальностей.

1. ПОСТАНОВКА ЗАДАЧИ

Для реализации данного проекта необходимо найти наикротчайший путь движения с препятствиями, собаки к сосиске, по заданной карте.
Нам дан размер сетки - M. N. Агент начинает в крайней левой ячейке в основании. Цель – самая крайняя правая ячейка в основании, то есть, $(1, N)$. В некоторых других ячейках может располагаться речка. Если агент попадет в нее, то он умирает. Так, цель агента состоит в том, чтобы достигнуть конечного пункта. Первая часть разрабатывается для Stupid dog при помощи randome. Во второй части мы используем генетический или другие виды алгоритмов обучения.

2. АРХИТЕКТУРА НЕЙРОННОЙ СЕТИ И АЛГОРИТМ ОБУЧЕНИЯ

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

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи

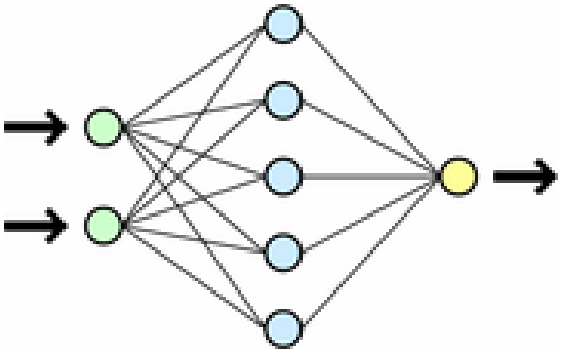

Схема простой нейросети. Зелёным обозначены входные элементы, жёлтым — выходной элемент

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть — способ решения проблемы эффективного параллелизма. А с точки зрения искусственного интеллекта, ИНС является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Нейронные сети не программируются в привычном смысле этого слова, они **обучаются**. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что, в случае успешного обучения, сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

Известные применения

Распознавание образов и классификация

В качестве образов могут выступать различные по своей природе объекты: символы текста, изображения, образцы звуков и т. д. При обучении сети предлагаются различные образцы образов с указанием того, к какому классу они относятся. Образец, как правило, представляется как вектор значений

признаков. При этом совокупность всех признаков должна однозначно определять класс, к которому относится образец. В случае, если признаков недостаточно, сеть может соотнести один и тот же образец с несколькими классами, что неверно. По окончании обучения сети ей можно предъявлять неизвестные ранее образы и получать ответ о принадлежности к определённому классу.

Топология такой сети характеризуется тем, что количество нейронов в выходном слое, как правило, равно количеству определяемых классов. При этом устанавливается соответствие между выходом нейронной сети и классом, который он представляет. Когда сети предъявляется некий образ, на одном из её выходов должен появиться признак того, что образ принадлежит этому классу. В то же время на других выходах должен быть признак того, что образ данному классу не принадлежит. Если на двух или более выходах есть признак принадлежности к классу, считается что сеть «не уверена» в своём ответе.

Принятие решений и управление

Эта задача близка к задаче классификации. Классификации подлежат ситуации, характеристики которых поступают на вход нейронной сети. На выходе сети при этом должен появиться признак решения, которое она приняла. При этом в качестве входных сигналов используются различные критерии описания состояния управляемой системы.

Кластеризация

Под кластеризацией понимается разбиение множества входных сигналов на классы, при том, что ни количество, ни признаки классов заранее не известны. После обучения такая сеть способна определять, к какому классу относится входной сигнал. Сеть также может сигнализировать о том, что входной сигнал не относится ни к одному из выделенных классов — это является признаком новых, отсутствующих в обучающей выборке, данных.

Таким образом, подобная сеть может выявлять новые, неизвестные ранее классы сигналов. Соответствие между классами, выделенными сетью, и классами, существующими в предметной области, устанавливается человеком. Кластеризацию осуществляют, например, нейронные сети Кохонена.

Прогнозирование и аппроксимация

Способности нейронной сети к прогнозированию напрямую следуют из ее способности к обобщению и выделению скрытых зависимостей между входными и выходными данными. После обучения сеть способна предсказать будущее значение некой последовательности на основе нескольких предыдущих значений и/или каких-то существующих в настоящий момент факторов. Следует отметить, что прогнозирование возможно только тогда, когда предыдущие изменения действительно в какой-то степени предопределяют будущие. Например, прогнозирование котировок акций на основе котировок за прошлую неделю может оказаться успешным (а может и не оказаться), тогда как прогнозирование результатов завтрашней лотереи на основе данных за последние 50 лет почти наверняка не даст никаких результатов.

Сжатие данных и Ассоциативная память

Способность нейросетей к выявлению взаимосвязей между различными параметрами дает возможность выразить данные большой размерности более компактно, если данные тесно взаимосвязаны друг с другом. Обратный процесс — восстановление исходного набора данных из части информации — называется (авто)ассоциативной памятью. Ассоциативная память позволяет также восстанавливать исходный сигнал/образ из зашумленных/поврежденных входных данных. Решение задачи гетероассоциативной памяти позволяет реализовать память, адресуемую по содержимому.

Этапы решения задач

- Сбор данных для обучения;
- Подготовка и нормализация данных;
- Выбор топологии сети;
- Экспериментальный подбор характеристик сети;
- Экспериментальный подбор параметров обучения;
- Собственно обучение;
- Проверка адекватности обучения;
- Корректировка параметров, окончательное обучение;
- Вербализация сети с целью дальнейшего использования.

Следует рассмотреть подробнее некоторые из этих этапов.

Сбор данных для обучения

Выбор данных для обучения сети и их обработка является самым сложным этапом решения задачи. Набор данных для обучения должен удовлетворять нескольким критериям:

- Репрезентативность — данные должны иллюстрировать истинное положение вещей в предметной области;
- Непротиворечивость — противоречивые данные в обучающей выборке приведут к плохому качеству обучения сети.

Исходные данные преобразуются к виду, в котором их можно подать на входы сети. Каждая запись в файле данных называется обучающей парой или обучающим вектором. Обучающий вектор содержит по одному значению на каждый вход сети и, в зависимости от типа обучения (с учителем или без), по одному значению для каждого выхода сети. Обучение сети на «сыром» наборе, как правило, не даёт качественных результатов. Существует ряд способов улучшить «восприятие» сети.

- Нормировка выполняется, когда на различные входы подаются данные разной размерности. Например, на первый вход сети подается величины со значениями от нуля до единицы, а на второй — от ста до тысячи. При отсутствии нормировки значения на втором входе будут всегда оказывать существенно большее влияние на выход сети, чем значения на первом входе. При нормировке размерности всех входных и выходных данных сводятся воедино;
- Квантование выполняется над непрерывными величинами, для которых выделяется конечный набор дискретных значений. Например, квантование используют для задания частот звуковых сигналов при распознавании речи;
- Фильтрация выполняется для «зашумленных» данных.

Кроме того, большую роль играет само представление как входных, так и выходных данных. Предположим, сеть обучается распознаванию букв на изображениях и имеет один числовой выход — номер буквы в алфавите. В этом случае сеть получит ложное представление о том, что буквы с номерами 1 и 2 более похожи, чем буквы с номерами 1 и 3, что, в общем, неверно. Для того, чтобы избежать такой ситуации, используют топологию сети с большим числом выходов, когда каждый выход имеет свой смысл. Чем больше выходов в сети, тем большее расстояние между классами и тем сложнее их спутать.

Выбор топологии сети

Выбирать тип сети следует исходя из постановки задачи и имеющихся данных для обучения. Для обучения с учителем требуется наличие для каждого элемента выборки «экспертной» оценки. Иногда получение такой оценки для большого массива данных просто невозможно. В этих случаях естественным выбором является сеть, обучающаяся без учителя, например, самоорганизующаяся карта Кохонена или нейронная сеть Хопфилда. При решении других задач, таких как прогнозирование временных рядов,

экспертная оценка уже содержится в исходных данных и может быть выделена при их обработке. В этом случае можно использовать многослойный перцептрон или сеть Ворда.

Экспериментальный подбор характеристик сети

После выбора общей структуры нужно экспериментально подобрать параметры сети. Для сетей, подобных перцептрону, это будет число слоев, число блоков в скрытых слоях (для сетей Ворда), наличие или отсутствие обходных соединений, передаточные функции нейронов. При выборе количества слоев и нейронов в них следует исходить из того, что способности сети к обобщению тем выше, чем больше суммарное число связей между нейронами. С другой стороны, число связей ограничено сверху количеством записей в обучающих данных.

Экспериментальный подбор параметров обучения

После выбора конкретной топологии, необходимо выбрать параметры обучения нейронной сети. Этот этап особенно важен для сетей, обучающихся с учителем. От правильного выбора параметров зависит не только то, насколько быстро ответы сети будут сходиться к правильным ответам. Например, выбор низкой скорости обучения увеличит время схождения, однако иногда позволяет избежать паралича сети. Увеличение момента обучения может привести как к увеличению, так и к уменьшению времени сходимости, в зависимости от формы поверхности ошибки. Исходя из такого противоречивого влияния параметров, можно сделать вывод, что их значения нужно выбирать экспериментально, руководствуясь при этом критерием завершения обучения (например, минимизация ошибки или ограничение по времени обучения).

Собственно обучение сети

В процессе обучения сеть в определенном порядке просматривает обучающую выборку. Порядок просмотра может быть последовательным, случайным и т. д. Некоторые сети, обучающиеся без учителя, например, сети Хопфилда просматривают выборку только один раз. Другие, например, сети Кохонена, а также сети, обучающиеся с учителем, просматривают выборку множество раз, при этом один полный проход по выборке называется эпохой обучения. При обучении с учителем набор исходных данных делят на две части — собственно обучающую выборку и тестовые данные; принцип разделения может быть произвольным. Обучающие данные подаются сети для обучения, а проверочные используются для расчета ошибки сети (проверочные данные никогда для обучения сети не применяются). Таким образом, если на проверочных данных ошибка уменьшается, то сеть действительно выполняет обобщение. Если ошибка на обучающих данных продолжает уменьшаться, а ошибка на тестовых данных увеличивается, значит, сеть перестала выполнять обобщение и просто «запоминает» обучающие данные. Это явление называется переобучением сети или оверфиттингом. В таких случаях обучение обычно прекращают. В процессе обучения могут проявиться другие проблемы, такие как паралич или попадание сети в локальный минимум поверхности ошибок. Невозможно заранее предсказать проявление той или иной проблемы, равно как и дать однозначные рекомендации к их разрешению

Проверка адекватности обучения

Даже в случае успешного, на первый взгляд, обучения сеть не всегда обучается именно тому, чего от неё хотел создатель. Известен случай, когда сеть обучалась распознаванию изображений танков по фотографиям, однако позднее выяснилось, что все танки были сфотографированы на одном и том же фоне. В результате сеть «научилась» распознавать этот тип ландшафта,

вместо того, чтобы «научиться» распознавать танки. Таким образом, сеть «понимает» не то, что от неё требовалось, а то, что проще всего обобщить.

Классификация по типу входной информации

- Аналоговые нейронные сети (используют информацию в форме действительных чисел);
- Двоичные нейронные сети (оперируют с информацией, представленной в двоичном виде).

Классификация по характеру обучения

- Обучение с учителем — выходное пространство решений нейронной сети известно;
- Обучение без учителя — нейронная сеть формирует выходное пространство решений только на основе входных воздействий. Такие сети называют самоорганизующимися;

Обучение с подкреплением — система назначения штрафов и поощрений от среды

Известные типы сетей

- Персептрон Розенблатта;
- Многослойный перцептрон;
- Сеть Джордана;
- Сеть Элмана;
- Сеть Хэмминга;
- Сеть Ворда;
- Сеть Хопфилда;
- Сеть Кохонена;
- Нейронный газ
- Когнитрон;
- Неокогнитрон;

- Хаотическая нейронная сеть;
- Осцилляторная нейронная сеть;
- Сеть встречного распространения;
- Сеть радиальных базисных функций (RBF-сеть);
- Сеть обобщенной регрессии;
- Вероятностная сеть;
- Сиамская нейронная сеть;
- Сети адаптивного резонанса.

Отличия от машин с архитектурой фон Неймана

Вычислительные системы, основанные на искусственных нейронных сетях, обладают рядом качеств, которые отсутствуют в машинах с архитектурой фон Неймана (но присущи мозгу человека):

- Массовый параллелизм;
- Распределённое представление информации и вычисления;
- Способность к обучению и обобщению;
- Адаптивность;
- Свойство контекстуальной обработки информации;
- Толерантность к ошибкам;
- Низкое энергопотребление.

Нейронные сети — универсальные аппроксиматоры

Нейронные сети — универсальные аппроксимирующие устройства и могут с любой точностью имитировать любой непрерывный автомат. Доказана обобщённая аппроксимационная теорема: с помощью линейных операций и каскадного соединения можно из произвольного нелинейного элемента получить устройство, вычисляющее любую непрерывную функцию с любой наперёд заданной точностью. Это означает, что нелинейная характеристика нейрона может быть произвольной: от сигмоидальной до произвольного волнового пакета или вейвлета, синуса или многочлена. От выбора

нелинейной функции может зависеть сложность конкретной сети, но с любой нелинейностью сеть остаётся универсальным аппроксиматором и при правильном выборе структуры может сколь угодно точно аппроксимировать функционирование любого непрерывного автомата.

3.2. Алгоритм обучения

В теории нейронных сетей существуют две актуальных проблемы, одной из которых является выбор оптимальной структуры нейронной сети, а другой - построение эффективного алгоритма обучения нейронной сети.

Оптимизация нейронной сети направлена на уменьшение объёма вычислений при условии сохранения точности решения задачи на требуемом уровне. Параметрами оптимизации в нейронной сети могут быть:

- размерность и структура входного сигнала нейросети;
- синапсы нейронов сети. Они упрощаются с помощью удаления из сети или заданием "нужной" или "оптимальной" величины веса синапса;
- количество нейронов каждого слоя сети: нейрон целиком удаляется из сети, с автоматическим удалением тех синапсов нейронов следующего слоя, по которым проходил его выходной сигнал;
- количество слоёв сети.

Вторая проблема заключается в разработке качественных алгоритмов обучения нейросети, позволяющих за минимальное время настроить нейросеть на распознавание заданного набора входных образов.

Процесс обучения нейронной сети заключается в необходимости настройки сети таким образом, чтобы для некоторого множества входов давать желаемое (или, по крайней мере, близкое, сообразное с ним) множество выходов.

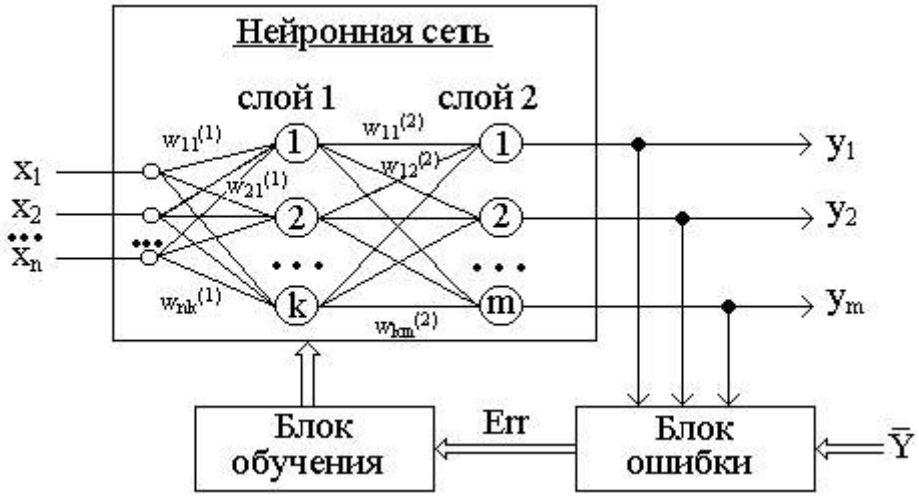

Рисунок 1 - Обучение "с учителем" многослойной нейронной сети

Стратегия "обучение с учителем" (рис. 1) предполагает, что есть обучающее множество {X,Y}.

Обучение осуществляется путем последовательного предъявления векторов обучающего множества, с одновременной подстройкой весов в соответствии с определенной процедурой, пока ошибка настройки по всему множеству не достигнет приемлемого низкого уровня.

$$Err = \sum_{i=1}^{N} \left| \overline{Y}_i - Y_i \right| \rightarrow \min$$

Зависимость реального выходного сигнала Y от входного сигнала X можно записать в виде: Y = F(W,X)+Err, где:

F(W,X) - некоторая функция, вид которой задается алгоритмом обучения нейронной сети;

W - множество параметров, позволяющих настроить функцию на решение определенной задачи распознавания образов (количество слоев сети, количество нейронов в каждом слое сети, матрица синаптических весов сети);

Err - некоторая ошибка, возникающая из-за неполного соответствия реального значения выходного сигнала нейронной сети требуемому значению, а также погрешности в вычислениях.

В последнее время для обучения многослойного персептрона широко используется процедура, получившая название алгоритма с обратным распространением ошибки (error backpropagation).

Однако этому алгоритму свойственны недостатки:

1. возможность преждевременной остановки из-за попадания в область локального минимума;
2. необходимость многократного предъявления всего обучающего множества для получения заданного качества распознавания.
3. отсутствие сколько-нибудь приемлемых оценок времени обучения.

Проблемы, связанные с алгоритмом обратного распространения привели к разработке альтернативных методов расчета весовых коэффициентов нейронных сетей. Впервые в 1989 году Дэвид Монтана и Лоуренс Дэвис использовали генетические алгоритмы в качестве средства подстройки весов скрытых и выходных слоев для фиксированного набора связей.

Генетический алгоритм является самым известным на данный момент представителем эволюционных алгоритмов и по своей сути является алгоритмом для нахождения глобального экстремума многоэкстремальной функции. Он заключается в параллельной обработке множества альтернативных решений. При этом поиск концентрируется на наиболее перспективных из них. Это говорит о возможности использования генетических алгоритмов при решении любых задач искусственного интеллекта, оптимизации, принятия решений.

Генетические алгоритмы для подстройки весов скрытых и выходных слоев используются следующим образом. Каждая хромосома (решение, последовательность, индивидуальность, "родитель", "потомок", "ребенок") представляет собой вектор из весовых коэффициентов (веса считываются с нейронной сети в установленном порядке - слева направо и сверху вниз).

Хромосома $a=(a_1, a_2, a_3, \dots, a_n)$ состоит из генов ai, которые могут иметь числовые значения, называемые "аллели".

Популяцией называют набор хромосом (решений).

Эволюция популяций - это чередование поколений, в которых хромосомы изменяют свои признаки, чтобы каждая новая популяция наилучшим способом приспосабливалась к внешней среде.

Начальная популяция выбирается случайно. Для обучения сети к начальной популяции применяются простые операции: селекция, скрещивание, мутация, - в результате чего генерируются новые популяции.

У генетического алгоритма есть такое свойство как вероятность. Т.е. описываемые операторы не обязательно применяются ко всем хромосомам, что вносит дополнительный элемент неопределенности в процесс поиска решения. В данном случае, неопределенность не подразумевает негативный фактор, а является своеобразной "степенью свободы" работы генетического алгоритма.

Оператор селекции (reproduction, selection) осуществляет отбор хромосом в соответствии со значениями их функции приспособленности. Здесь применим метод рулетки (roulette-wheel selection), с помощью которого осуществляется отбор. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i-го сектора пропорционален соответствующей величине $P_{sel}(i)$ вычисляемой по формуле:

$f_i(x)$ - значение целевой функции i-й хромосомы в популяции;

$$P_i = \frac{f_i(x)}{sumf(x)}$$

sumf(x) - суммарное значение целевой функции всех хромосом в популяции;

Для вычисления минимума функции f(x) требуется найти нормализованную величину k_i: $k_i 2p^{cp}-p_i$

p^{cp} - среднее значение нормализованной величины p.

Ожидаемое число копий i-й хромосомы после оператора селекция определяются так: $N_i=k_i n$

где n - число анализируемых хромосом.

При таком отборе члены популяции с более высокой приспособленностью будут чаще выбираться, чем особи с низкой приспособленностью.

1	0.6	6%
2	0.2	31%
3	0.1	19%
4	0.9	44%

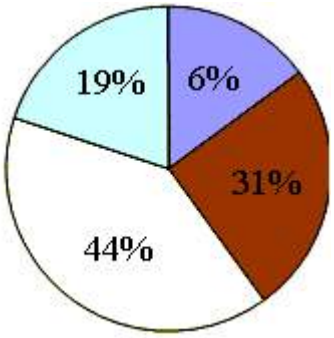

Рисунок 2 - Оператор селекции типа колеса рулетки с пропорциональными функции приспособленности секторами

Оператор кроссовера (crossover operator), иногда называемый кроссинговером, является основным генетическим оператором (рис. 3), за счет которого производится обмен частями хромосом между двумя (может быть и больше) хромосомами в популяции. Может быть одноточечным или многоточечным. Одноточечным называется кроссовер, если при нем родительские хромосомы разрезаются только в одной случайной точке. Для реализации N-точечного кроссовера м.б. использовано два подхода :

Точек разрыва меньше, чем генов в хромосоме, либо

Если длина хромосомы L битов, то число точек разрыва равно (L-1), при этом потомки наследуют биты следующим образом: первому потомку достаются нечетные биты первого родителя и четные биты второго; у второго же потомка все наоборот. Т.е. получается как бы "расческа".

Кроме описанных типов кроссовера есть ещё однородный кроссовер. Его особенность заключается в том, что значение каждого бита в хромосоме потомка определяется случайным образом из соответствующих битов родителей. Для этого вводится некоторая величина $0<P_0<1$, и если случайное число больше p_0, то на n-ю позицию первого потомка попадает n-й бит первого родителя, а на n-ю позицию второго - n-й бит второго родителя. В противном случае к первому потомку попадает бит второго родителя, а ко второму - первого. Такая операция проводится для всех битов хромосомы.

Вероятность кроссовера самая высокая среди генетических операторов и равна обычно 60% и больше.

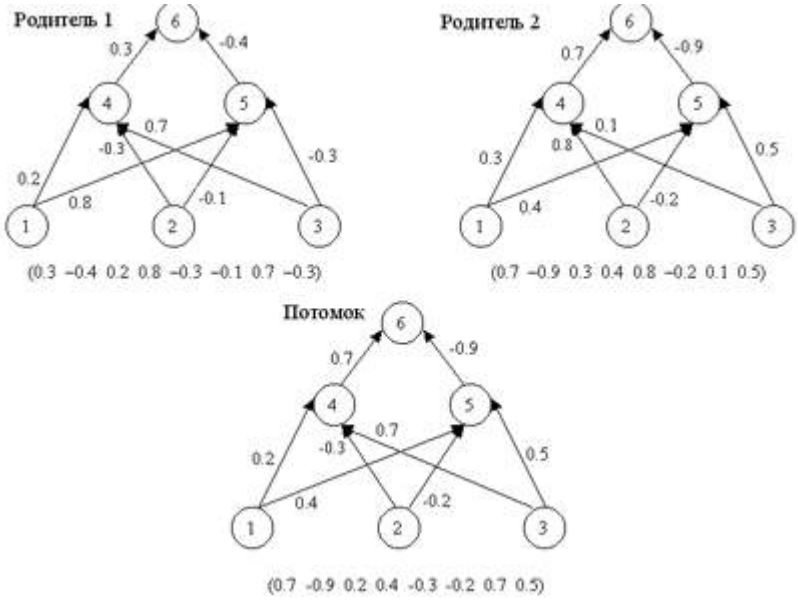

Рисунок 3 - Оператор скрещивания

Оператор мутации (mutation operator) - стохастическое изменение части хромосом. Он необходим для "выбивания" популяции из локального экстремума и способствует защите от преждевременной сходимости. Достигаются эти чудеса за счет того, что каждый ген строки, которая подвергается мутации, с малой вероятностью Pmut меняется на другой ген (добавляется случайная величина между -1.0 и 1.0 к весу). Это показано на рисунке 4.

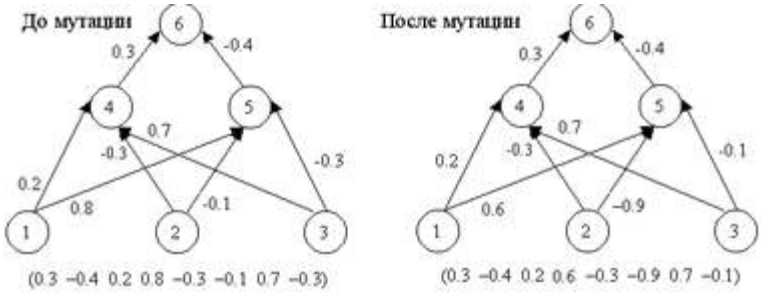

Рисунок 4 - Оператор мутации

Так же как и кроссовер, мутация проводится не только по одной случайной точке. Можно выбирать некоторое количество точек в хромосоме для изменения, причем их число также может быть случайным. Также можно изменять сразу некоторую группу подряд идущих точек. Вероятность мутации значительно меньше вероятности кроссовера и редко превышает 1%.

Обычно при реализации ГА к скрещиваемым особям сначала применяют оператор кроссовера, а потом оператор мутации, хотя возможны варианты.

Например, оператор мутации можно применять только если к данной паре родительских особей не был применен оператор кроссовера (свойство вероятности). В принципе, никто не мешает вообще не использовать вероятность проведения данного генетического оператора и применять кроссовер ко всем отобранным особям, а мутацию к каждому 100 потомку. В общем здесь - полная свобода действий.

Следует заметить, что генетические алгоритмы будут наиболее полезны для подстройки весов в задачах, где обратное распространение и его аналоги не могут использоваться, например, в неконтролируемых задачах, когда ошибка выхода не может быть вычислена. Это часто имеет место для задач нейроуправления, в которых нейронные сети используются, чтобы управлять сложными системами (например, проведение роботов в незнакомых окружающих средах).

3. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ СИСТЕМЫ

Для реализации проекта была выбрана среда программирования Borland Delphi 6.0. Выбор обусловлен тем, что Borland Delphi 6.0 базируется на технологии объектно-ориентированного программирования и позволяет создавать приложения под операционные системы Windows 9x, XP с удобным для пользователя интерфейсом.
Целью курсового проекта является исследование нейронных сетей для прогнозирования и аппроксимации функции. В соответствии с этим было разработано программное обеспечение, позволяющее проводить обучение нейронной сети для выполнения данных задач.

Ниже представлены модули разработанной системы:

M0 – главный модуль;
M1 – Модуль работы с картой;
M2 – Модуль работы с “Random Walker”;
M3 – Модуль работы с “Genetic algorithm”;
M4 – построение графика работы модуля M3;

При запуске модуля M0 мы видим главную форму на которой изображено поле где мы будем устанавливать объекты. Также внизу расположены клавиши для работы с программой.
На модуле M1 мы производим расстановку объектов. Для нашего задания это – собака, сосиска и река.
Модуль M2 работает с картой с уже установленными объектами. Рэндомно находит путь, либо же его не находит. И на карте показывает пройденный путь для достижения цели.
Модуль M3 работает также с картой с установленными объектами, но для нахождения оптимального пути используется генетический алгоритм.

Модуль М4 используется для построения графика результатов полученных из модуля М3. На оси x указывается номер популяции, а на оси y среднее значение количества шагов для популяции.

4. ТЕСТИРОВАНИЕ СИСТЕМЫ

На рисунке 5 представлена главная форма с которой мы производим основные операции.

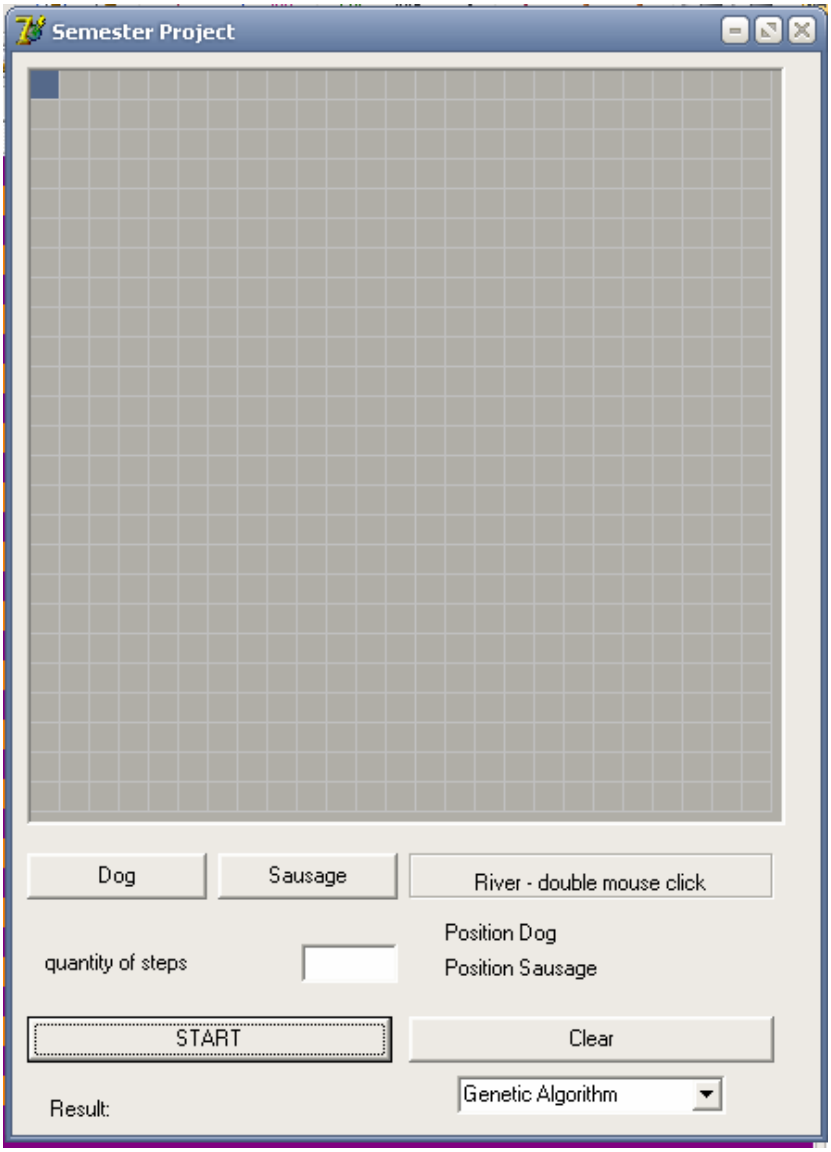

Рисунок 5. Главная форма

На ней мы расставляем позиции объектов. Задаем количество шагов в поле Quantity of steps. Кнопкой Dog мы устанавливаем положение собаки, кнопкой Sausage устанавливаем положение сосиски, двойным щелчком мыши устанавливаем реку. В списке выбираем Random Walker. (Рисунок 6)

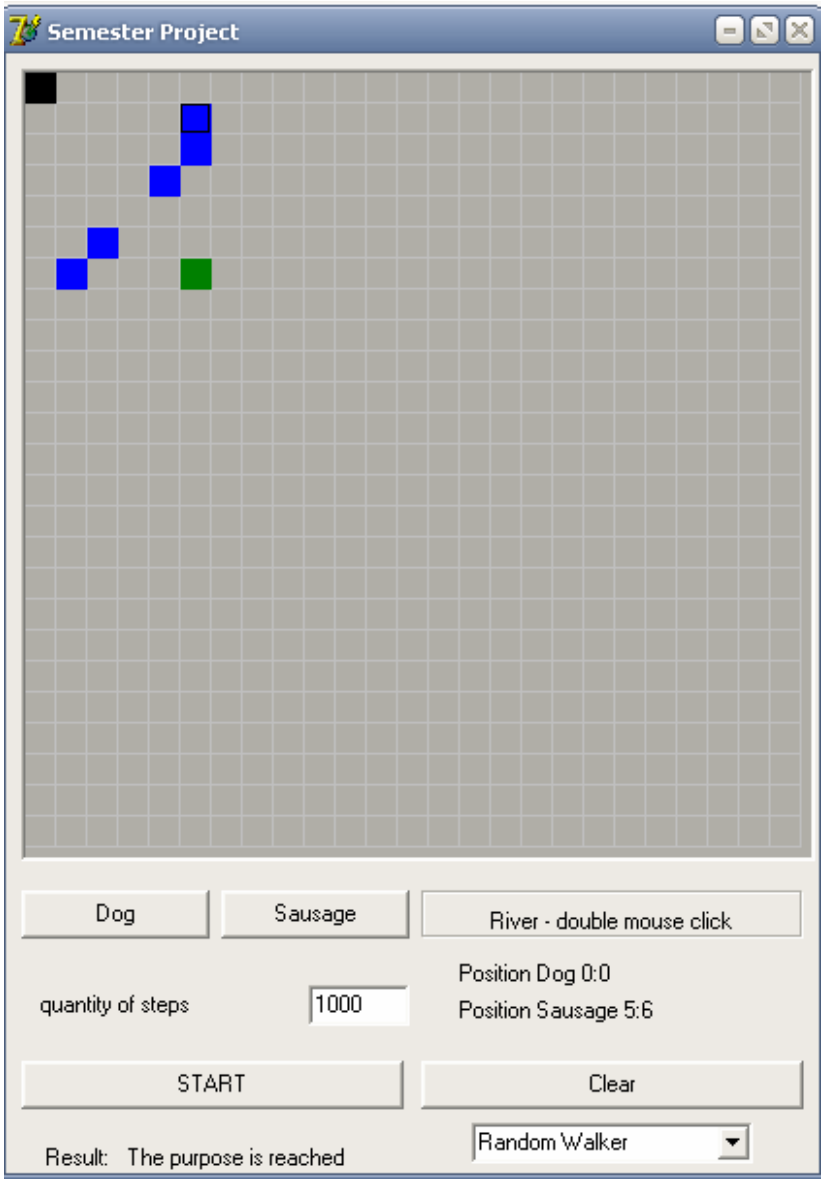

Рисунок 6. Заполненная форма для Random Walker

При нажатии на кнопку START мы получаем результат достигнута цель либо нет, изображение пройденного пути в поле Quantity of steps указывается количество шагов, за которые была достигнута цель. (Рисунок 7)

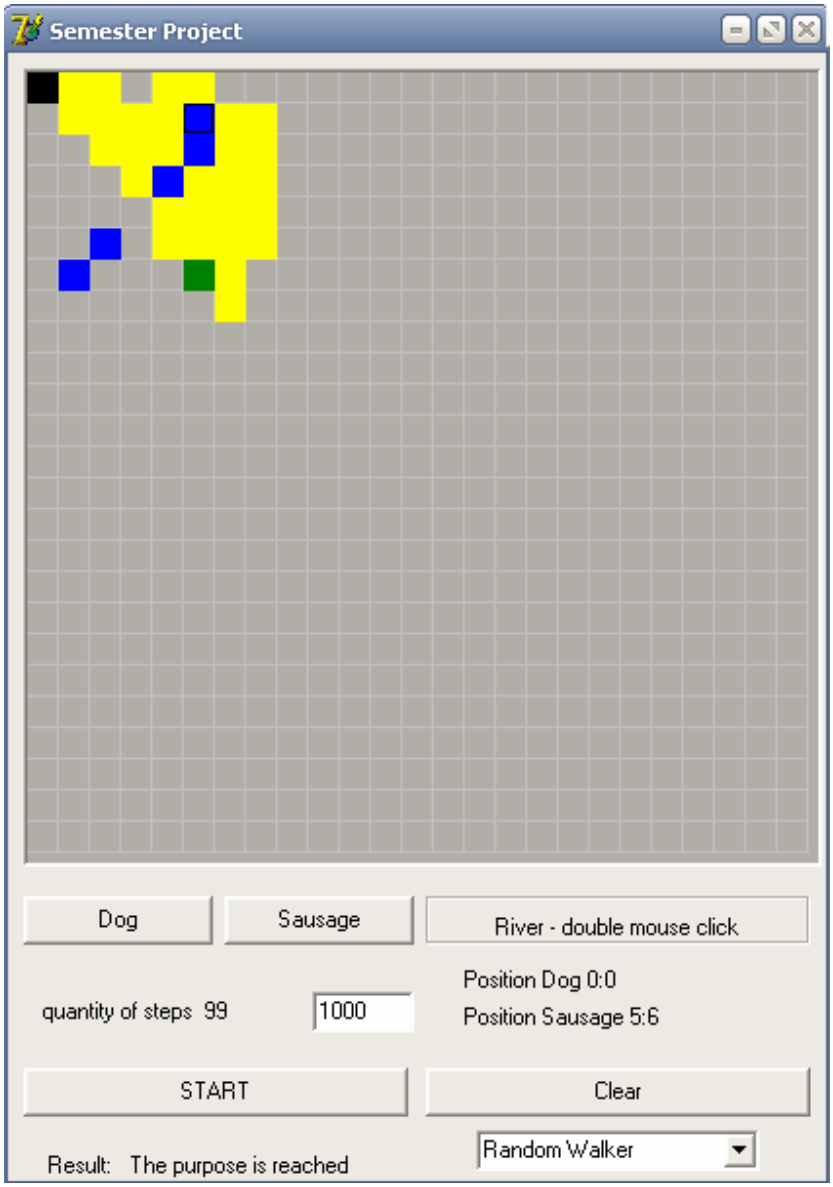

Рисунок 7. Результат метода Random Walker

В результате получаем графики успешных агентов (Рисунок 8, 9) для разных заданных значений.

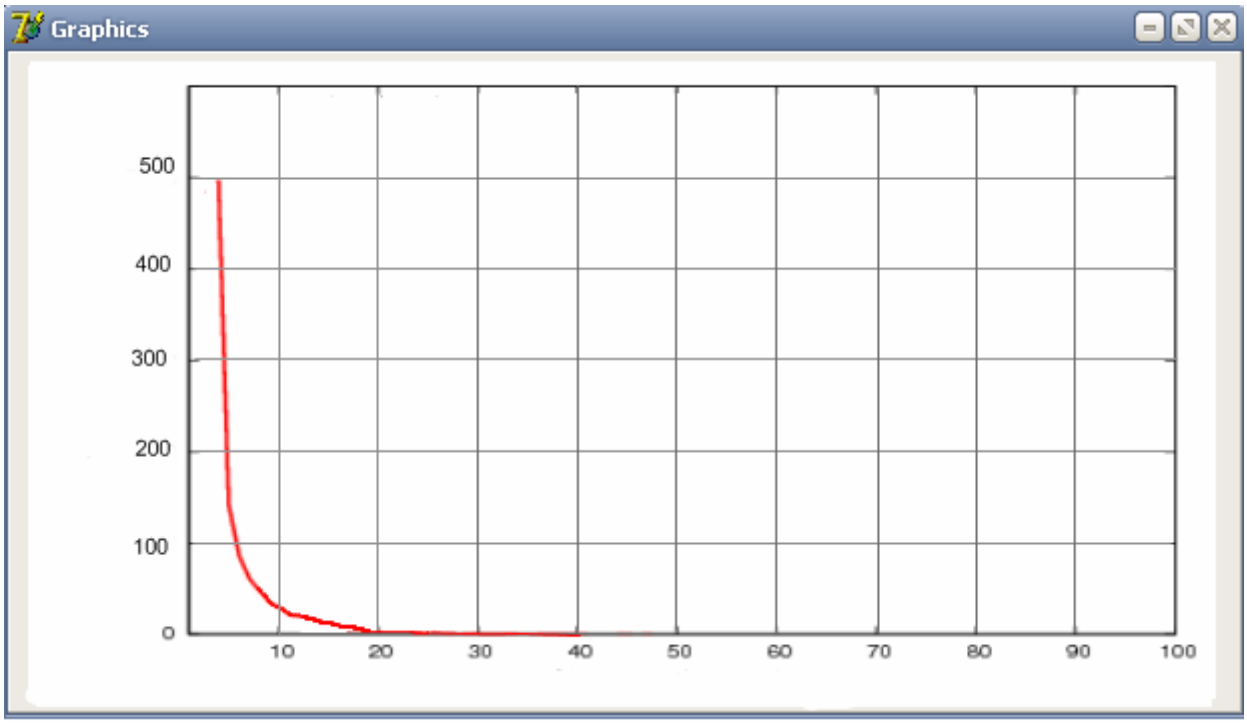

Рисунок 8. Количество успешных агентов (5000 агентов и 200 жизней)

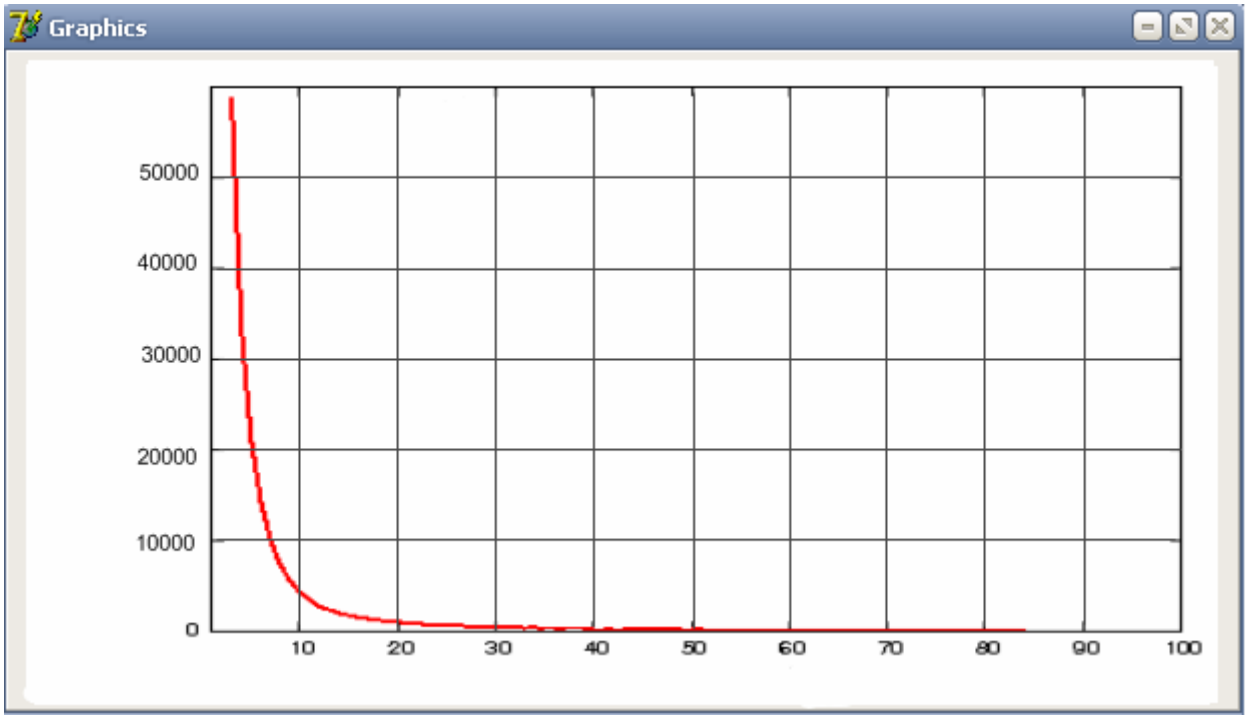

Рисунок 9. Количество успешных агентов (100000 агентов и 300 жизней)

Для получения результатов при помощи генетического алгоритма мы также наставляем объекты. В списке выбираем Genetic algorithm (Рисунок 10).

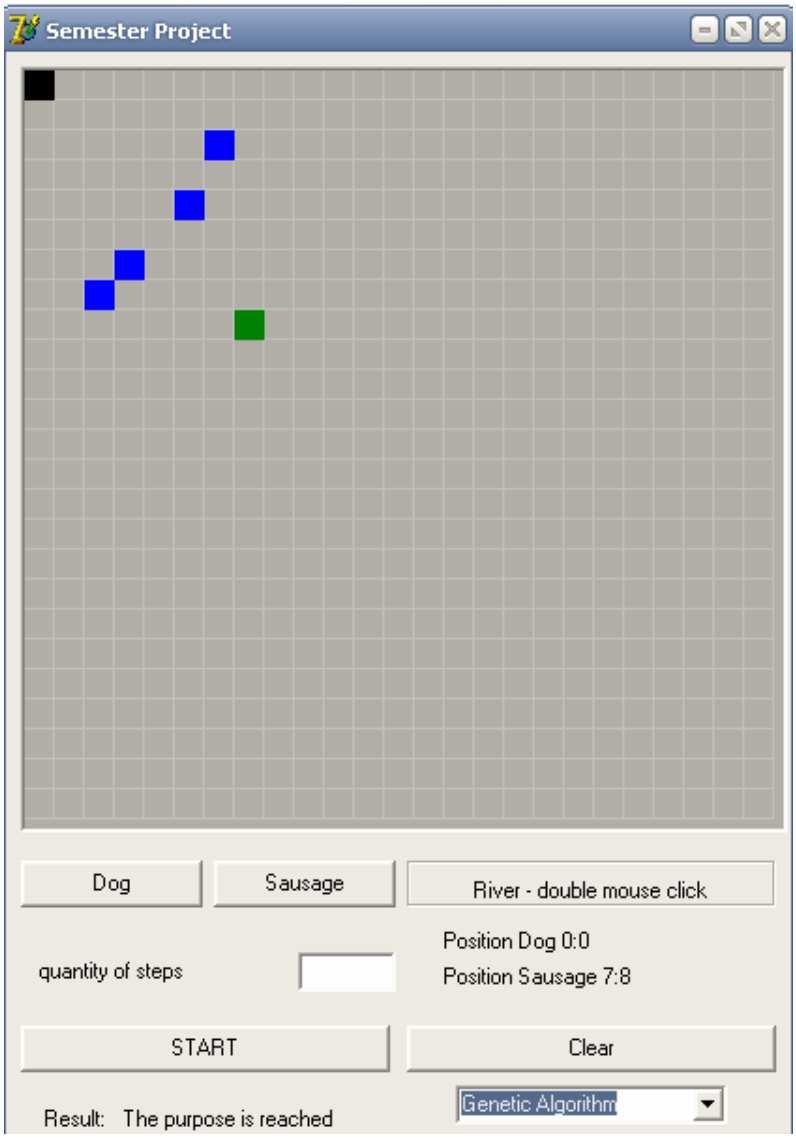

Рисунок 10. Заполненная форма для Genetic Algotrithm

После нажатия на START вводим количество популяций (Рисунок 11)

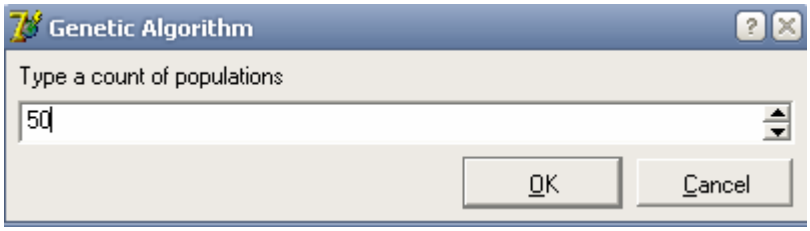

Рисунок 11. Ввод количества популяций

И количество шагов (Рисунок 12)

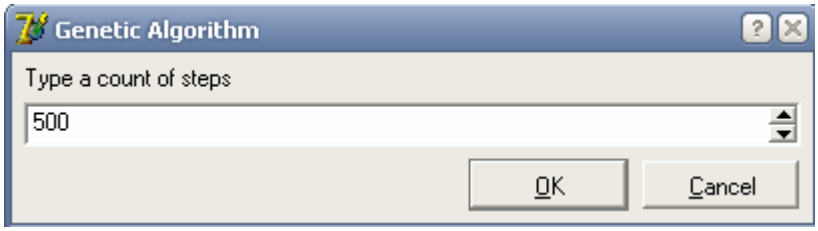

Рисунок 12. Ввод количества шагов

В результате получаем график для количества поколений (Рисунок 13), для обученного агента с длинной вектора 100 и 50 (Рисунок 14, 15).

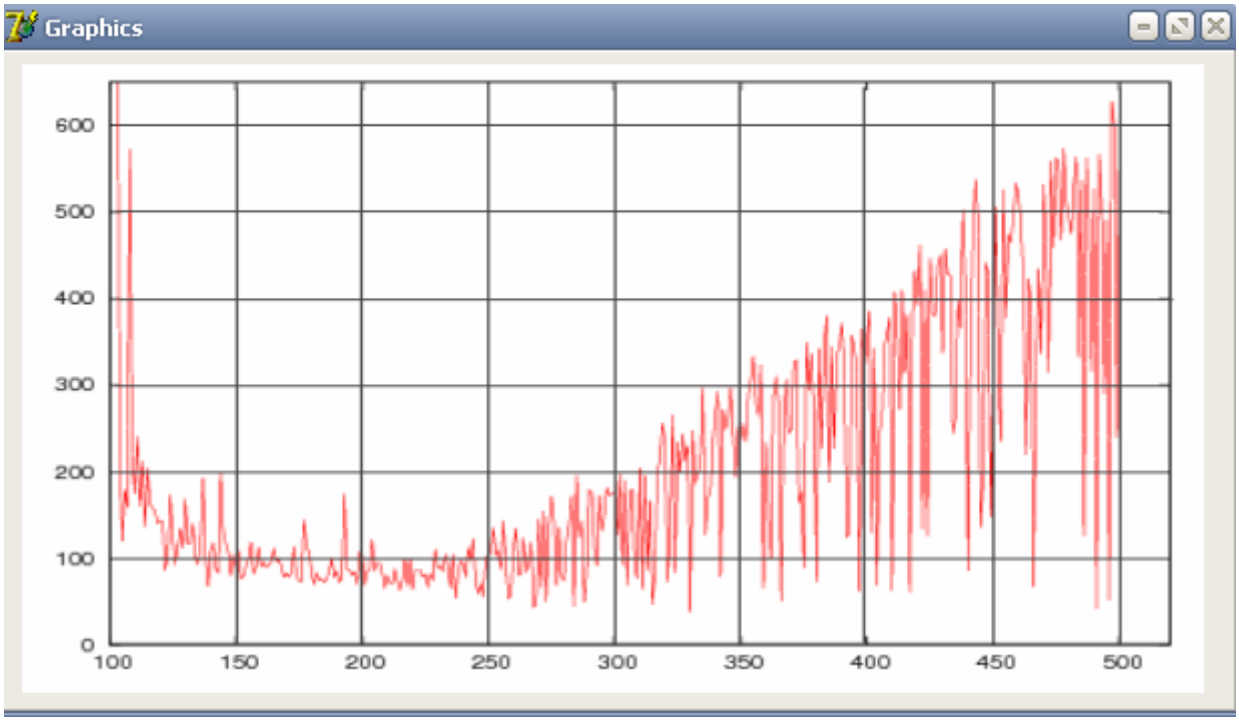

Рисунок 13. Количество поколений

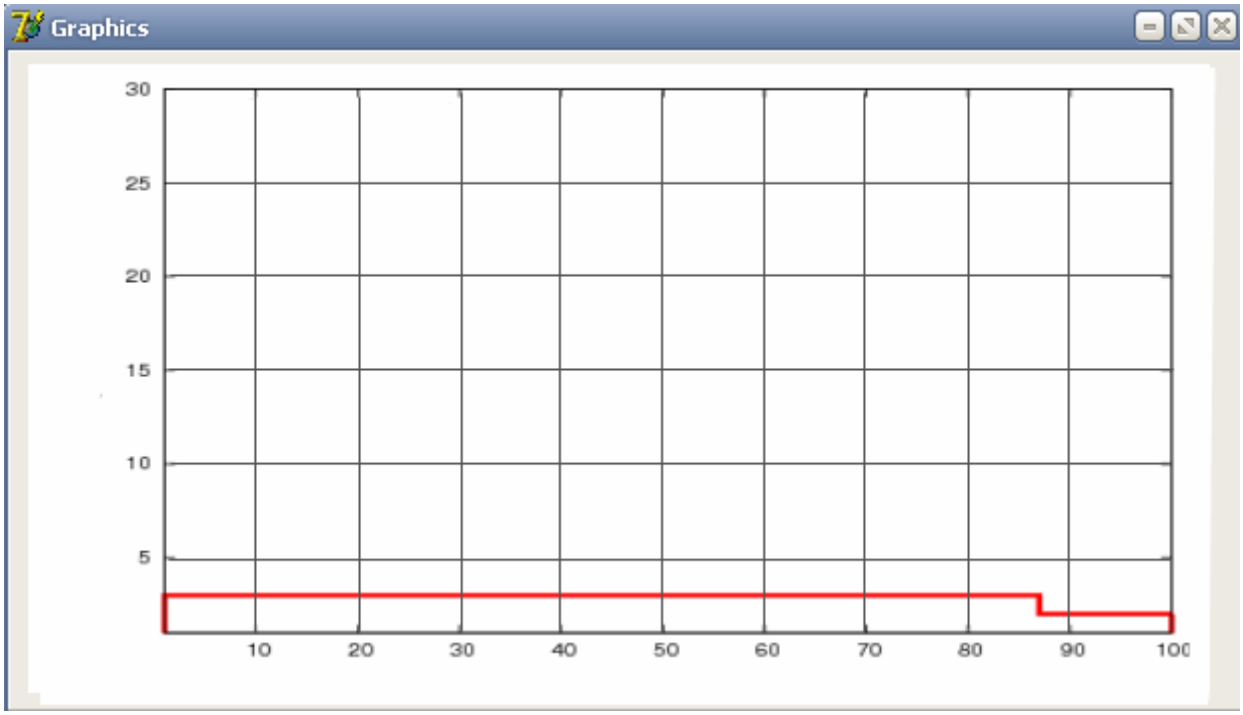

Рисунок 14.Обучение агента с длинной вектора 100

Рисунок 15. Обучение агента с длинной вектора 50.

ЗАКЛЮЧЕНИЕ

В ходе проведенной работы была написана программы для поиска наикратчайшего пути от одного объекта до другого. Работа разделялась на две части. В первой поиск наикратчайшего пути осуществлялся рэндомно. Во второй же части мы использовали генетический алгоритм.

ЛИТЕРАТУРА

1)Конспект лекций по дисциплине «Методы и системы принятия решений».

2) Лекции по теории и приложениям искусственных нейронных сетей. http://alife.narod.ru/lectures/neural/,2002

3)ГОСТ 19.106-78. ЕСПД: Требования к программным документам, выполненным печатным способом.

4) Методические указания.