МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

ФАКУЛЬТЕТ КИБЕРНЕТИКИ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ
Кафедра «Информатика и процессы управления» (№ 17)

ОТЧЕТ
ОБ УЧЕБНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ
ПО ТЕМЕ:

«Исследование оптимизации обучающих выборок с помощью генетического алгоритма»

Студент группы	*К7-171*		*Жаров Я.М.*
	(номер группы)	(подпись, дата)	(ФИО студента)
Руководитель			*Катаев Д.Е.*
		(подпись, дата)	(уч. степень, уч. звание, ФИО руководителя)

Москва
2014

Содержание

Введение

Одним из ключевых моментов в реализации систем, использующих алгоритмы машинного обучения с учителем, является подготовка обучающей выборки. Этот процесс включает в себя отбор наиболее релевантных признаков из доступных (feature selection) и предобработку данных с целью устранения шумов, выбросов, либо применения специфичных для задачи вспомогательных преобразований. Часто для этого используется человеческий труд, однако в настоящее время существует множество работ, посвященных автоматическому решению данных задач. Существующие алгоритмы и системы можно условно разделить на системы обучающихся классификаторов и решения задач оптимизации параметров систем машинного обучения с помощью разнообразных эвристик. Их общей чертой является использование алгоритмов случайного поиска, как правило, генетических. В рамках данной работы исследуется генетический алгоритм (с нейронной сетью как фитнесс-функцией), используемый для отбора признаков и предобработки данных. [1, 2, 6, 7, 4]

Цель данной работы разработка новой версии ПО для генетической оптимизации обучающих выборок систем машинного обучения с учителем. Модификация алгоритма для устранения избыточности обучающих данных. А так же оценка класса решаемых алгоритмом задач.

1 Система оптимизации обучающих выборок

1.1 Описание исходной системы

Пусть $f(S_h) = (S_t; S_v))$ — отображение f исторических данных S_h в совокупность обучающей S_t и валидационной S_v выборок. Пусть $N(S_t; S_v)$ — искусственная нейронная сеть прямого распространения, обученная по S_t и проверяемая по S_v, а $Err_v(N(S_t; S_v))$ — ее ошибка валидации, определяемая как среднеквадратичное отклонение всех выходов сети от соответствующих эталонных значений при подаче на вход элементов валидационной выборки. Тогда

$$\exists \bar{f} : \bar{f}(S_h) = arg\{minErr_v(N(S_t; S_v))\} \tag{1}$$

Требуется найти $\bar{f}$. Для этого реализован следующий алгоритм, представленный на рис. 1.1

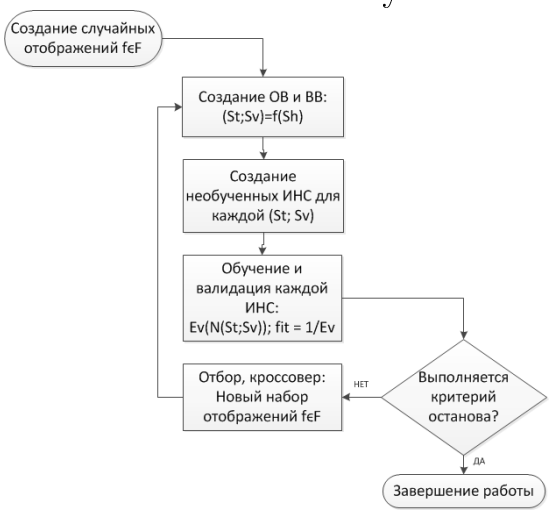
Критерием останова в используемой реализации является прохождение заданного количества итераций. В текущей реализации используются ИНС прямого распространения в одним скрытым слоем, обучаемые методом эластичного распространения (RPROP).

Отображения f представляют собой конструкции следующего вида:

$$f(S_h) = \begin{vmatrix} (\lambda_{i_1} \circ \lambda_{i_2} \circ ... \circ \lambda_{i_m})(S_{h_{j_1}}) \\ ... \\ (\lambda_{i_{n-l}} \circ ... \circ \lambda_{i_n})(S_{h_{j_k}}) \end{vmatrix}, \lambda \in \Lambda, i,n,m,j \in N, \tag{2}$$

где Λ — множество доступных системе элементарных преобразований сигналов (дифференцирование, фильтры, добавление лагов и т.п.), S_{h_j} — произвольный сигнал из исторических данных . Атомарной единицей при кроссовере является $f(S_h)_i$, однако мутации возможны и внутри нее.

Рисунок 1.1 — Схема используемого алгоритма



При хранении экземпляр кодировался линейной последовательностью исполняемых команд и списком команд, результаты выполенения которых попадают в обучающую выборку. Такая структура данных не способствовала тонкой настройке алгоритмов кроссовера и мутации.

1.2 Недостатки системы

При работе со старой системой был выявлен быстрый рост размера экземпляров в ходе работы генетического алгоритма, в результате чего росло количество лишних данных и вычислений. Также в ней использовался крайне примитивный и плохонастраиваемый алгоритм мутации, предположительно не являющийся наилучшим. В прошлой версии системы использовалось распределеннные вычисления, что, хотя и позволяло использовать большие вычислительные мощности, неоправданно усложняло процесс эксплуации.

1.3 Постановка задачи учебно-исследовательской работы

Основной задачей являлась разработка новой версии программного обеспечения для оптимизации обучающих выборок при помощи генетического алгоритма на основе старой с реализацией ряда модификаций:

а) Модификация формата вывода данных для облегчения взаимодействия с пользователем;

б) Устранение избыточности обучающих данных для снижения вычислительной мощности;

в) Модификация алгоритма мутации для реализации рекурсивной мутации.

2 Разработка и реализация модернизированной системы оптимизации обучающих выборок

2.1 Разработка системы оптимизации обучающих выборок

2.1.1 Модификация структуры вывода данных

В силу тяжести взаимодействия пользователя с исходным форматом вывода данных было решено сменить этот формат. Целями смены формата были: повышение человекочитаемости, повышение легкости транспортировки и модифицирования данных эксперимента, по возможности сохранение легкости загрузки данных в приложение. Возможными структурами хранения при решении данной задачи были XML, JSON, исполняемый код или база данных.

Таблица 2.1 Сравнение возможных структур хранения данных

Средство	Читаемость	Передача	Скорость
XML	Читаемый в сыром виде формат. Для прочтения в удобном виде достаточно элементарного браузера.	Для передачи достаточно просто переслать интересующее поддерево или файл целиком.	Парсинг требует затрат времени. Затраты, однако, можно сократить оптимизацией, основанной на жесткости структуры.
JSON	Читаемость формата в сыром виде несколько затруднена. Требуется специальный просмотровщик.	Для передачи достаточно переслать интересующий объект или файл целиком.	Парсинг требует минимального времени — фактически это преобразование из строки в код.

Продолжение таблицы 2.1

Средство	Читаемость	Передача	Скорость
Исполняемый код	Читаемость формата в сыром виде требует специальных навыков. Желательна подсветка синтаксиса и знание структуры программы.	Для передачи данных достаточно передать цельный кусок кода, что тоже требует специальных навыков.	Парсинг требует минимального времени — фактически это преобразование из строки в исполняемый код.
База данных	Читаемость в сыром почти невозможна. Требуются специальные просмотровщики. Цельность данных с точки зрения просмотра теряется.	Передача данных как таковых требует либо объединенной БД для нескольких экземпляров системы, между которыми идет передача, либо выгрузки из базы данных.	Скорость работы зависит от СУБД и скорости соединения, если база данных удалена, однако скорость работы любой базы данных на локальной машине — выше, чем скорость преобразования текстовых файлов.

По результатам анализа разницы между представленными альтернативами был выбран формат данных XML. [A]

2.1.2 Устранение избыточности обучающих данных

В предыдущей версии возникла проблема избыточности размера экземпляров, которая достаточно критично влияла на объем вычислений, производимых системой. Задачей является сокращение роста экземпляров при сохранении возможности такого роста.

Устранить избыточность обучающих данных можно было при помощи нескольких модификаций:

а) Отбрасывание на каждой мутационной фазе алгоритма одного элемента из экземпляра и проверка направления изменения фитнесс-функции;

б) Жесткое ограничение на размер экземпляра;

в) Изменение распределения вероятностей в пользу уменьшения, а не увеличения размера экземпляра.

Вариант с жестким ограничением на размер экземпляра был отвергнут в связи с невозможностью в общем случае предугадать реальную правильную длину. Вариант с изменением алгоритма мутации был исключен в силу увеличения времени работы каждого шага почти в два раза. Реализуемым вариантом стал вариант с изменением распределения вероятностей. Конкретным изменением послужило увеличение возможного интервала размеров на следующем шаге в сторону уменьшения.

2.1.3 Модификация алгоритма мутации для реализации рекурсивной мутации

Для осуществления возможности вносить большее разнообразие в экземпляры и большей настраиваемости хода работы алгоритма требовалось введение возможности мутировать не только атомарные объекты, но и объединяющие их структуры.

Для реализации рекурсивной мутации в каждой сущности (вплоть до атомарного элемента — функции ЦОС) была модифицирована процедура мутации. На фазе мутации каждая сущность верхнего уровня с заданной вероятностью мутировала полностью — генерировался новый экземпляр этой сущности, в обратном же случае она вызывала мутацию всех своих сущностей-детей.

2.2 Реализация системы оптимизации обучающих выборок

Текущая версия ПО представляет из себя пакет скриптов на языке python версии 2.7, содержащий инструменты для проведения вычислений и анализа результатов.

Алгоритм по существу не менялся по сравнению с прошлой версией. Однако изменению подверглись его подалгоритмы и методы реализации.

Модуль вычислений описан в диаграмме классов 2.1

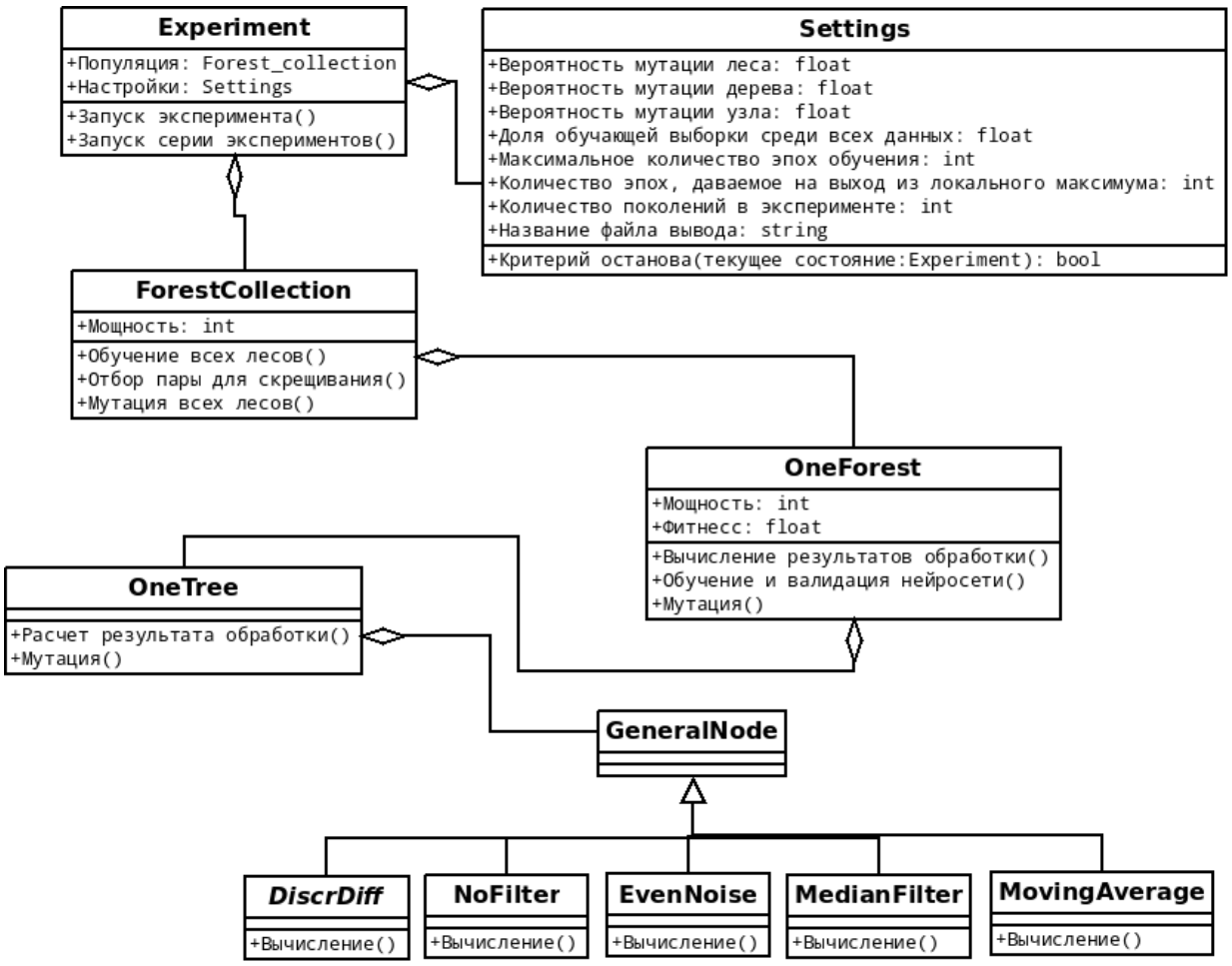


Рисунок 2.1 — диаграмма классов модуля проведения вычислений

Для инициализации вычислений требуется представить входные данные для обработки и входные данные для вычислительного модуля. Стандартным методом получения данных для обработки является набор csv-файлов со значениями. Входные данные для вычислительного модуля фактически представляют собой экземпляр класса настроек. Выходные данные вычислительного модуля представлены в XML-формате и выгружаются в файл с настраиваемым названием.

Модуль обработки результатов вычислений представляет собой набор скриптов, позволяющих собирать информацию о ходе и результатах генетического алгоритма как с одного, так и с набора XML-файлов. По результатам работы модуль строит csv-файл, содержащий в себе выбранные и обработанные значения.

Используемые библиотеки:

PyBrain — для создания и обучения искусственных нейросетей;

SciPy и NumPy — для ускорения вычислений. [8]

3 Тестирование разработанной системы

3.1 Используемые тестовые сценарии

Для тестирования использовалось два типа систем: линейная и нелинейная. Под сценарием эксперимента подразумевается набор из входных и целевых данных для нейросети. [3]

3.1.1 Линейная система

Пусть существует бассейн, который является объектом управления. К нему подведены две трубы: через одну в бассейн может поступать вода с произвольной скоростью (в данном тестовом сценарии не будут учитываться физические ограничения), через вторую трубу вода может уходить из бассейна с произвольной скоростью. Через эти две трубы осуществляется управление бассейном. Ограничения на то, что объем воды в бассейне должен быть неотрицательным нет, но, стоит отметить, что в данном сценарии не возникает соответствующей ситуации. Пусть существуют следующие экспериментальные данные:
$X^{(0)}$ – объем поступившей за Δt воды;
$X^{(1)}$ – объем откачанной за Δt воды;
$X^{(2)}$ – объем воды, находящейся в бассейне в данный момент времени;
$X^{(3)}$ – посторонний сигнал, отличающийся от остальных;
Бассейн реагирует на управляющие воздействия следующим образом:

$$X_i^{(2)} = X_{i-1}^{(2)} + X_{i-1}^{(0)} - X_{i-1}^{(1)}; X_0^{(2)} = 10 \tag{3}$$

Пусть испытуемая система не имеет априорной информации об объекте управления и имеет доступ к следующим данным в качестве исходных:
$X^{(0)\Delta t}$ – объем поступившей за Δt воды, на этот сигнал наложен импульсный шум (inflow_noizd);
$X^{(1)\Delta t}$ – объем откачанной за Δt воды (outflow);
$X^{(3)}$ – посторонний сигнал (trash).

3.1.2 Нелинейная система

Пусть $X^{(0)}, .., X^{(5)}$ — действительные временные ряды из которых $X^{(3)}, .., X^{(5)}$ не оказывают на систему никакого воздействия.
$X^{(6)}$ — выход системы.

Система реагирует на управляющие воздействия следующим образом:

$$X_i^{(6)} = |X^{(0)} - X^{(1)}| + X^{(2)} \tag{4}$$

Перед подачей в вычислительный модуль на $X^{(0)}$ был наложен постоянный шум с амплитудой 0.5, на $X^{(1)}$ наложен шум, прибавляющий или вычитающий 1 с вероятностями по 2.5% (условно такой шум можно считать импульсным). Остальные сигналы зашумлению не подвергались.

3.2 Экспериментальная проверка работоспособности системы

В прошлой версии для анализа хода работы системы использовался метод, предполагающий постановку одного эксперимента с большим размером популяции. Для аналогичных целей в данной работе был использован метод, предполагающий постановку серии экспериментов с малым размером популяции и усреднение данных среди нескольких экспериментов с одинаковыми условиями. Предполагалось, что данный метод позволит усматривать более тонкие зависимости за счет увеличения времени сходимости, а усреднение данных среди нескольких экспериментов позволит избежать влияния случайных совпадений среди сгенерированной стартовой популяции. [5]

3.2.1 Работоспособность модуля для линейной системы

Для проверки работоспособности модуля при прогнозировании работы линейной системы из серии экспериментов с использованием линейного тестового сценария был выбран лучший экземпляр и построен прогноз, данный нейронной сетью, обученной лучшим экземпляром, в сравнении с истинными значениями. Результаты построения приведены на рисунке 3.1. Прогноз строился для контрольной части данных.

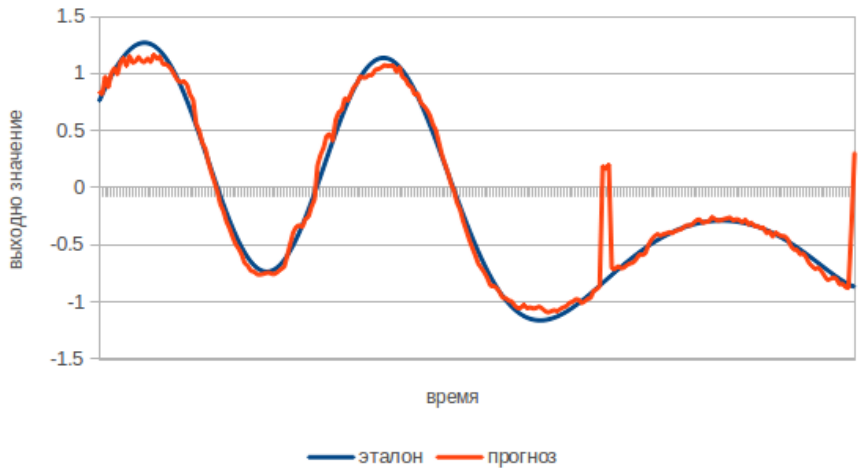


Рисунок 3.1 — Результаты прогноза и эталонная функция выхода

Доверительный интервал с уровнем доверия 98% оказался равен ±0.3

3.2.2 Работоспособность модуля для нелинейной системы

Для проверки работоспособности модуля при прогнозировании работы нелинейной системы из серии экспериментов с использованием нелинейного тестового сценария был выбран лучший экземпляр и построен прогноз, данный нейросетью, обученной лучшим экземпляром, в сравнении с истинными значениями. Результаты построения приведены на рисунке 3.2. Прогноз строился для контрольной части данных.

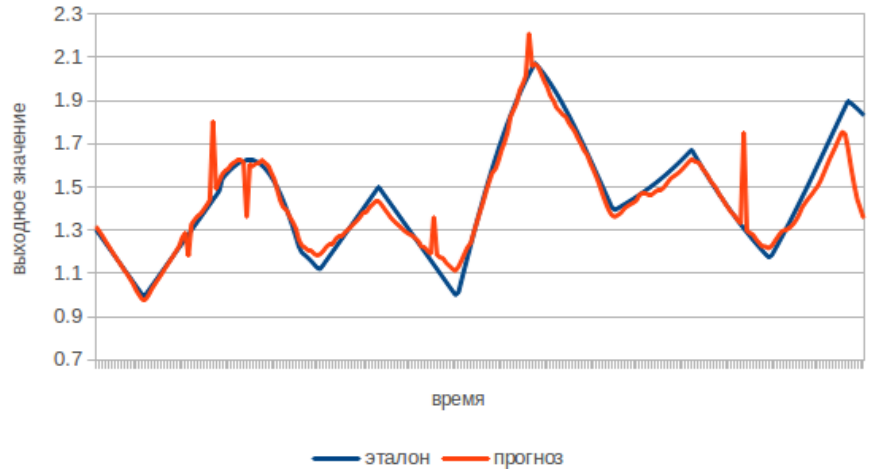


Рисунок 3.2 — Результаты прогноза и эталонная функция выхода

Доверительный интервал с уровнем доверия 97% оказался равен ±0.3

3.2.3 Устранение избыточности

В качестве эксперимента для проверки работоспособности решения по устранению избыточности было решено провести пару тестов на одинаковых входных данных для старого и нового алгоритмов кроссовера. В качестве метрики для сравнения был выбран размер экземпляра на определенной итерации, усредненный среди нескольких экспериментов, для устранения возможности влияния случайных факторов.

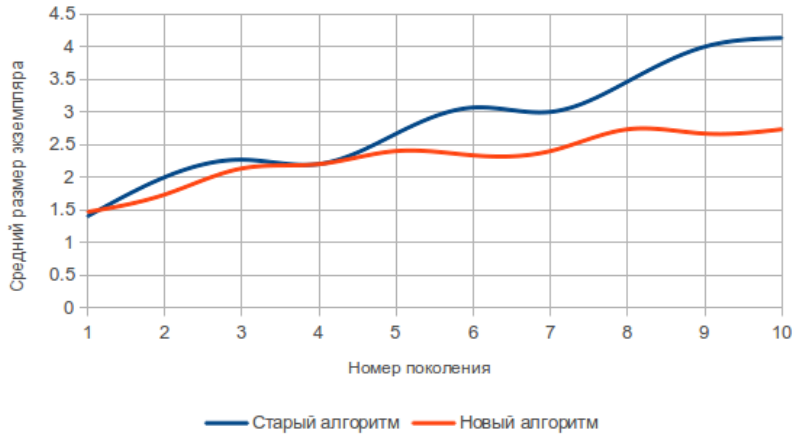


Рисунок 3.3 — Средние размеры экземпляров по поколениям для старого и нового алгоритма

Средняя производная по таким графикам различается в два раза в пользу новой версии алгоритма кроссовера, что говорит о существенном снижении объемов избыточных данных в обучающей выборке. Важно также отметить, что истинным размером является два, что близко к полученному в этом эксперименте для нового алгоритма.

3.2.4 Модернизация алгоритма мутации

Для оценки качества модификации алгоритма мутации было проведено равное количество экспериментов при одинаковых начальных условиях для старого и нового алгоритма мутации для разных вероятностей мутации. Для моделирования старого алгоритма (плоский случай) новому задавались нулевыми вероятности мутации всех элементов кроме атомарных — функций ЦОС. Для моделирования алгоритма с использованием рекурсивной мутации вероятности

выставлялись одинаковыми и равными приведенной вероятности рассчитаной по формуле 5

$$P_n = \sqrt[3]{p-1} + 1 \tag{5}$$

, где p — вероятность для плоского случая (приведенная вероятность), а P_n — аналогичная ей вероятность для рекурсивного случая.

На основе экспериментов, проведенных с линейной системой, построены графики, приведенные на рисунках 3.4 и 3.5. На основе графика 3.5 можно сказать, что после реализации новой версии мутации уменьшилось время сходимости алгоритма, а также на графике 3.4 видно, что фитнесс-функция оптимального экземпляра не только не ухудшилась, но и улучшилась при таком переходе.

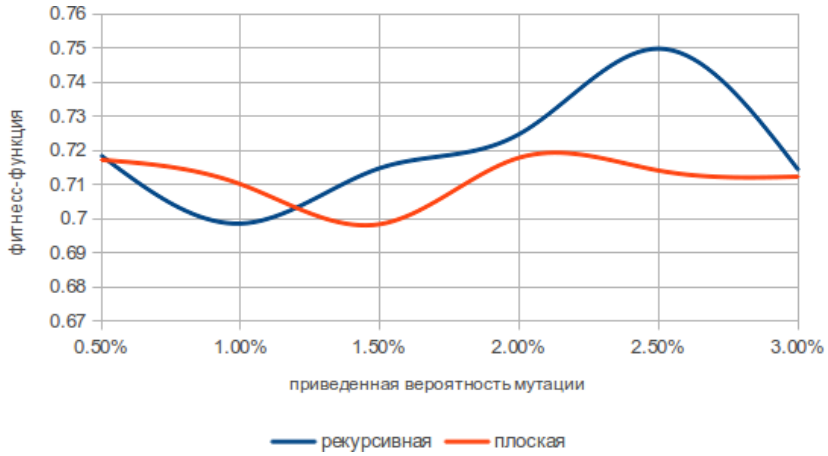


Рисунок 3.4 — средняя фитнесс-функция для приведенных вероятностей мутации

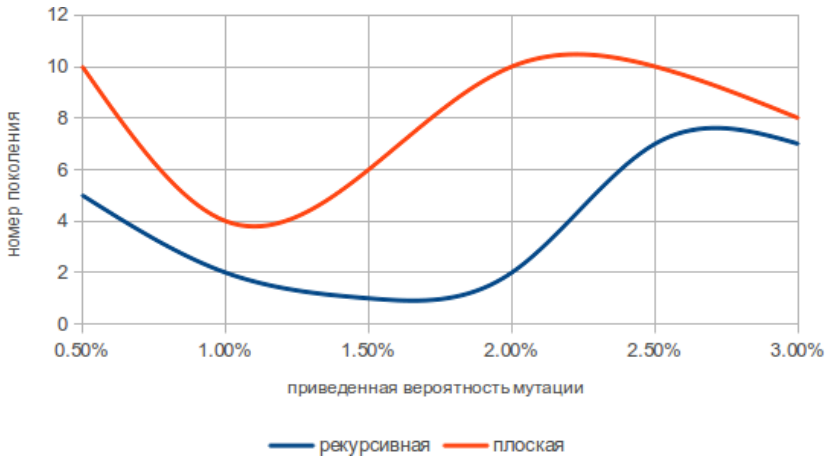


Рисунок 3.5 — время сходимости для приведенных вероятностей мутации

На основе экспериментов, проведенных с нелинейной системой, построены графики, приведенные на рисунках 3.6 и 3.7. Средние и максимальные значения

на графике 3.6 для рекурсивной и плоской мутации оказались примерно равными. То есть в пределах поставленных экспериментов нельзя сказать, что какая-либо из представленных функций сходится к более высокому значению. Как видно на графике 3.7, рекурсивный алгоритм почти всегда сходится быстрее, а также сходится быстрее в среднем.

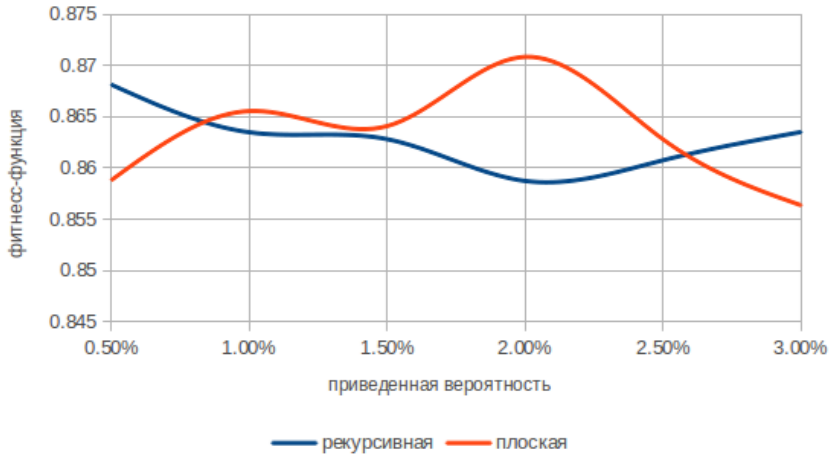


Рисунок 3.6 — средняя фитнесс-функция для приведенных вероятностей мутации

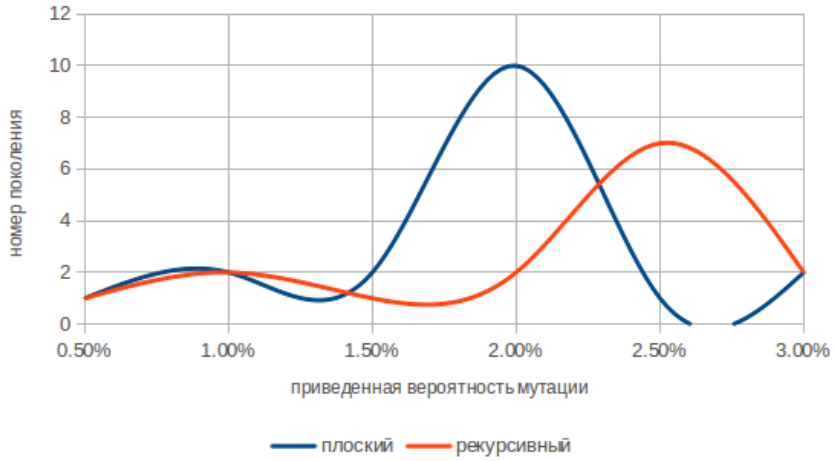


Рисунок 3.7 — время сходимости для приведенных вероятностей мутации

3.2.5 Модификация структуры вывода данных

В силу вычислительной сложности задачи эксперименты зачастую ставились на сторонней машине, после чего результаты вычислений пересылались для анализа на основную машину. Путь передачи с копированием XML-файлов показал себя как надежный и простой. За время передачи данных не произошло

ни одной потери и синхронная включенность обеих машин не требовалась для передачи данных.

3.3 Результаты анализа экспериментов

В ходе тестирования разработанной системы были получены следующие результаты:

а) Разработанная система способна найти субоптимальное решение как для линейной, так и для нелинейной системы. Обученная нейросеть в таком случае дает прогноз для доверительного интервала ± 0.3 в 98% и 97% соответственно;

б) Модификация алгоритма для устранения избыточности позволяет снизить прирост размера экземпляров в два раза;

в) Модернизация алгоритма мутации способна ускорить сходимость и, по крайней мере в некоторых случаях, улучшить результаты поиска;

г) Был использован новый метод исследования работы алгоритма, который, однако, при увеличении сложности анализа и вычислительной сложности не дал положительного прироста информативности.

Заключение

В результате данной работы была разработана и реализована новая версия программного обеспечения для оптимизации обучающих выборок с помощью генетического алгоритма. В ходе разработки были проведены модификации алгоритмов некоторых модулей системы, позволившие достигнуть улучшения работы системы по сравнению с предыдущей версией. В процессе реализации был использован набор сторонних библиотек, что позволило упростить дальнейшее модифицирование системы и уменьшить объем реализуемого исходного кода (PyBrain для работы с нейронными сетями и SciPy + NumPy для вычислительных задач). Был проведен переход системы на другой язык программирования (Python 2.7) и использование системы контроля версий (git) для упрощения дальнейшей модификации и поддержания системы.

В результате тестирования разработанной системы были получены следующие новые результаты:

а) Система показала работоспособность в случае нелинейных систем. То есть способность найти субоптимальное решение такое, что прогноз укладывается в доверительный интервал ± 0.3 с вероятностью 97% для исследованной системы;

б) Модификация алгоритма для устранения избыточности позволяет снизить прирост размера экземпляров в два раза;

в) Модернизация алгоритма мутации способна ускорить сходимость и, по крайней мере в некоторых случаях, улучшить результаты поиска.

Список использованных источников

1 Oreski S., Oreski D., Oreski G. Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. // Expert Systems with Applications. 2012 № 39 С. 12605–12617

2 Brownlee J. Clever Algorithms: Nature-Inspired Programming Recipes. ISBN: 978-1-4467-8506-5, 2012. 438 с.

3 Комашинский В.И., Смирнов Д.А. К63 Нейронные сети и их применение в системах управления и связи. М.: Горячая линия-Телеком, 2003. 94 с. ISBN 5-93517-094-9

4 Жданов А.А. Автономный искусственный интеллект 2-е изд. М.: БИНОМ. Лаборатория знаний, 2009. 359 с. ISBN 978-5-94774-995-3

5 Николенко С.И., Тулупьев А.Л. Н63 Самообучающиеся системы. М.: МЦНМО, 2009. 288 с. ISBN 978-5-940570-506-1

6 Han J., Kamber M., Pei J. Data mining: concepts and techniques 3-rd ed. ISBN 978-0-12-381479-1

7 Gabrys B., Ruta D. Genetic algorithms in classifier fusion // Applied Soft Computing. 2006 № 6 С. 337-347

8 Oliphant T.E. Python for Scientific Computing // Computing in Science & Engineering. 2007 № 9 С. 10-20

A Пример экземпляра популяции генетического алгоритма записанного в XML

```
<forest power="4"  fitness="0.839943138225">
<tree input="xor2" power="5">
<node function="MedianFilter" enter_params="{'frame': 3}"/>
<node function="DiscrDiff" enter_params="{}"/>
<node function="MedianFilter" enter_params="{'frame': 5}"/>
<node function="EvenNoise" enter_params="{'frame': 0}"/>
<node function="MedianFilter" enter_params="{'frame': 7}"/>
</tree>
<tree input="trash3" power="2">
<node function="DiscrDiff" enter_params="{}"/>
<node function="NoFilter" enter_params="{}"/>
</tree>
<tree input="trash1" power="3">
<node function="MedianFilter" enter_params="{'frame': 7}"/>
<node function="NoFilter" enter_params="{}"/>
<node function="DiscrDiff" enter_params="{}"/>
</tree>
<tree input="trash3" power="5">
<node function="DiscrDiff" enter_params="{}"/>
<node function="EvenNoise" enter_params="{'frame': 0}"/>
<node function="MedianFilter" enter_params="{'frame': 7}"/>
<node function="EvenNoise" enter_params="{'frame': 0}"/>
<node function="MovingAverage" enter_params="{'window': 7}"/>
</tree>
</forest>
```