НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

УДК №2123132123
Регистрационный №123123
Инв. №

УТВЕРЖДАЮ

Директор ООО «Рога и Копыта»

_____ И.И.Иванов
«__»_____ 2013 г.

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

«Оптимизация обучающих выборок с помощью генетического алгоритма»

Этап №1.1 «Обзор современного состояния торсионных наногенераторов»
(промежуточный)

Руководитель НИР,
инженер _____ Д.Е. Катаев

г. Москва, 2013

Список исполнителей

Научный руководитель,
доцент К.К.Петров _____

с.н.с, к.т.н,
Ж.Ж. Балбесов, _____

Реферат

Отчет 16 с, 2 таблица, 4 рисунок, 1 источник.

Содержание

Введение

Одним из ключевых моментов в реализации систем, использующих алгоритмы машинного обучения с учителем является подготовка обучающей выборки. В настоящее время, как правило, синтез обучающей выборки выполняется экспертом.ъ, который, однако, далеко не для всех задач в состоянии за разумное время выбрать и представить исходные данные задачи в оптимальном для обучаемой системы виде. Таким образмо имеется потребонсть в системах, которые способны автоматически формировать обучающие выборки для интеллектуальных систем из имеющихся данных таким образом, чтобы их обучение по автоматически сформированным выборкам было максимально эффективным. Естественно, подобные системы востребованы только в тез случаях, когда либо не существует, либо не сформулирован алгоритм формирования обучающей выборки. Таким образом наиболее иодходящим выглядит использование алгоритмов случайного поиска, например генетических.

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

1 Постановка задачи

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 структуры хранения данных

б) Устранение избыточности обучающих данных

в) Модификация алгоритма мутации для реализации рекурсивной мутации

г) Реализация на Python

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

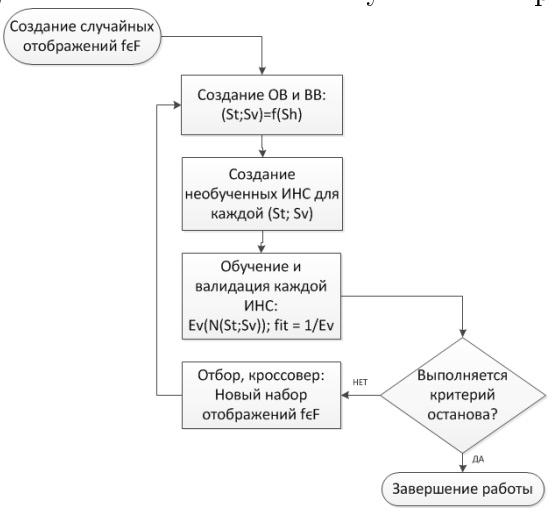
$$\exists \bar{f} : \bar{f}(S_h) = arg\{minErr_v(N(S_t; S_v)).\} \tag{1}$$

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 методов 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 описание стоящих задач

1.2.1 Модификация структуры хранения данных

В силу тяжести взаимодействия человека с исходным форматом хранения данных было решено сменить формат хранения данных при смене версии приложения. Целями смены формата было повышение читаемости человеком, легкости транспортировки и модифицирования данных эксперимента, но при этом сохранение, по возможности легкости загрузки данных в приложение, для возможности восстановить экземпляры полученные в ходе эксперимента.

Возможными структурами хранения при решении данной задачи были xml, json, исполняемый код или хранение в реляционной базе данных. К плюсам xml при решении поставленной задачи следует отнести следующее:

а) Возможность прочтения человеком структуры эксперимента в достаточно читаемом виде при наличии достаточно примитивных средств просмотра (Все остальные способы не отличаются простотой)

б) Достаточно легкая транспортабельность результатов эксперимента при необходимости. (Это преимущество работает, пожалуй, только относительно хранения в базе данных)

По результатам выбора между представленными альтернативами был выбран формат данных xml.

1.2.2 Устранение избыточности обучающих данных

В предыдущей версии возникла проблема избыточности размера экземпляров, которая при длительном ходе эксперимента могла достаточно критично повлиять на объем излишних вычислений, производимых системой. Задачей является сокращение роста избыточности экземпляров, при сохранении притом возможности такого роста.

Устранить избыточность обучающих данных можно было при помощи нескольких модификаций:

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

б) Жесткое ограничение на размер экземпляра

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

Вариант с жестким ограничением на размер экземпляра был отвергнут в связи с предугадать рельную 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.

1.2.3 Модификация алгоритма мутации для реализации рекурсивной мутации

Для осуществления возможности вносить большее разнообразие в экземпляры и большей настраивамости хода работы алгоритма требовалось введение возможности мутировать не только атомарные объекты но и блоки этих объектов.

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 полностью новый экземпляр этой сущности, в обратном же случае она вызывала мутацию всех своих сущностей-детей.

1.2.4 Реализация на Python

В ходе разработки новой версии возникла потребность реализовать систему набором средств предоставляющих инструментарий для легкой поддержки и модификации системы.

В качестве языка разработки был выбран python версии 2.7. python на данный момент является широко распространенным языком для научной разработки. Набор библиотек позволяет не заниматься реализацией большинства общепринятых алгоритмов, например алгоритм обучения с учителем. Язык обладает четким и последовательным синтаксисом, а так

же хорошей масштабируемостью, что позволяет сохранить читаемость, сопровождаемость и дополняемость кода. К тому же Python умеет работать с такими языками как Fortran, C и C++, которые уже широко используются в научных расчетах.

2 Описание результата разработки

3 Эксперименты

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.

3.1.1 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;

Реально 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).

Главной целью системы является верно спрогнозировать динамику изменения уровня воды в бассейне, то есть $\frac{dX^{(2)}}{dt}$.

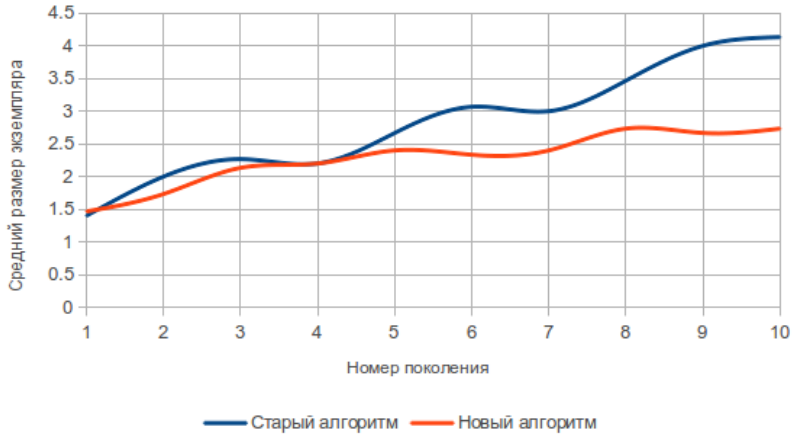
3.1.2 Нелинейная система

3.2 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 экземпля-

ра на определенной итерации, усредненный среди нескольких экспериментов, для устранения возможности влияния случайных факторов.

Рисунок 3.1 — график средних размеров 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 решении данной поставленной задачи по крайней мере в ограниченном объеме.

3.3 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 новому задавались нулевыми вероятности мутации всех элементов кроме атомарных - функций ЦОС. Для моделирования аналогичной вероятности с использованием рекурсивной мутации вероятности выставлялись одинаковыми и равными приведенной вероятности рассчитаной по формуле. (4)

$$P_n = \sqrt[3]{p-1} + 1 \tag{4}$$

Где p - вероятность для плоского случая (приведенная вероятность), а P_n - аналогичная ей вероятность для рекурсивного случая.

На основе экспериментов построены графики 3.2 и 3.3. На основе графика 3.3 можно сказать, что после реализации новой версии мутации уменьшилось время сходимости алгоритма, а так как на графике 3.2 видно, что фитнесс-функция оптимального экземпляра не только не ухудшилась но и улучшилась при таком переходе, можно сказать что мы не столкнулись с явлением преждевременной сходимости.

Рисунок 3.2 — график средней фитнесс-функции для приведенных вероятностей мутации

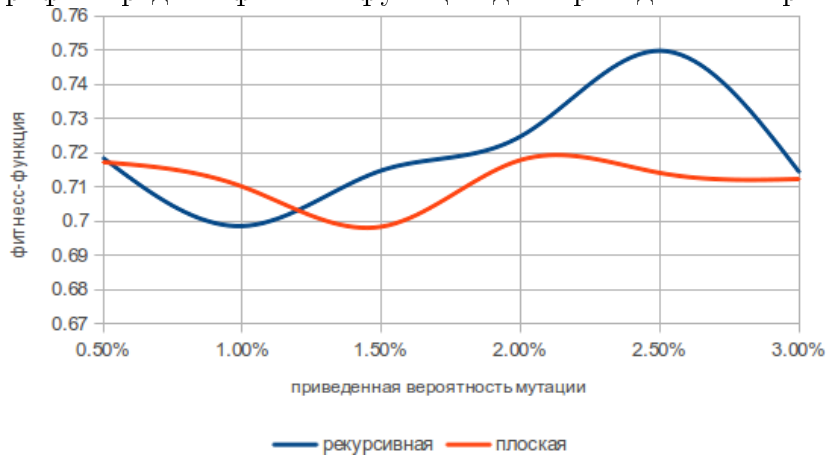
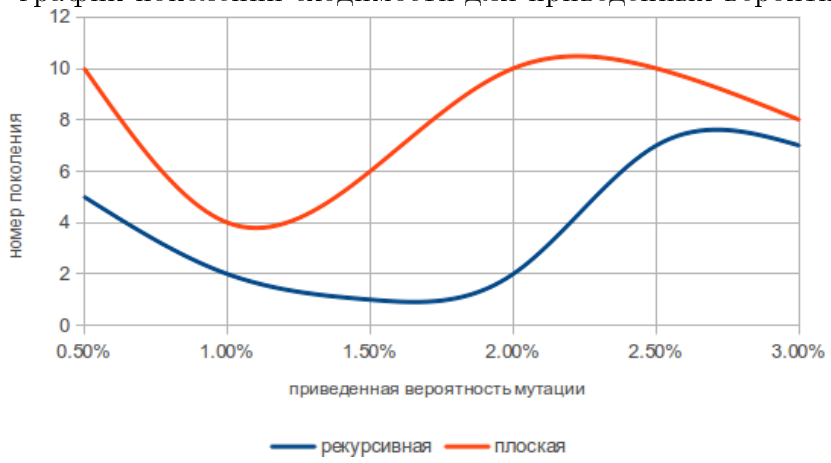


Рисунок 3.3 — график поколений сходимости для приведенных вероятностей мутации



4 Постановка задачи

В прошлой версии программного обеспечения, реализованного в дипломной работе научного руководителя присутствовала избыточность обучающих данных и слабая сопровождаемость кода.

Основной задачей была реализация новой версии ПО на python с попутным внесением технических и алгоритмических изменений Основной задачей было избавление от избыточности обучающих данных. В качестве метода решения этой задачи была выбрана модификация алгоритма кроссовера, а точнее смещение вероятности при выборе размера нового экземпляра в сторону уменьшения. В качестве критерия оценки избыточности была выбрана следующая метрика: средний размер экземпляра в популяции { на последней итерации || на момент максимума || в течение всех поколений }

В качестве оценок качества кода и сопровождаемости были выбраны следующие эксперименты, метрики и критерии:

а) Валидация PEP8 & PEP257

б) Проведение ряда экспериментов работавших в прошлой версии ПО, как средство проверки правильности практической реализации алгоритма

в) Сравнение характеристик сходимости при старом и новом алгоритме мутации

5 Решение поставленных задач

В прошлой версии программного обеспечения, реализованного в дипломной работе научного руководителя присутствовал ряд проблем:

а) Избыточность обучающих данных вызванная алгоритмом кроссовера.

б) Слабая сопровождаемость кода.

6 разнообразная хуита из шаблона.

В отчётах могут быть и таблицы - см.табл. 6.1.

Таблица 6.1 — Пример таблицы

Название 1	Название 2	Название 3
Это	пример	данных
помещённых	внутрь	таблицы

Заключение

Список использованных источников

1 Грицанов А.А. и др. Новейший философский словарь. Мн.: Книжный Дом., 2003.