Эволюционные вычисления: нейронные сети и генетические алгоритмы

Т. Н. Кондратьев
Финансовый университет при Правительстве Российской Федерации (Финуниверситет), Financial University
bestkondor@bk.ru

***Аннотация*. Общий пул технологий, объединенных под общим термином «системы искусственного интеллекта» не так уж и велик. К нему с разной степенью достоверности можно отнести: нейронные сети, генетические алгоритмы и так далее. Они все имею общие черты: предполагают обучение системы, основа — это база фактов или обучающая выборка, как совокупность образцов в рамках классифицирующих признаков, результатом вычисления являются, как правило, какие-либо прецеденты из заранее установленного списка. Таким образом, целью данного исследования является изучение нейрокомпьютерных сетей и генетических алгоритмов и областей их применения.**

Ключевые слова: нейронные сети; генетические алгоритмы; эксперементальные исследования; эволюционные вычисления

I. Введение

Нейронная сеть — это последовательность нейронов, соединенных между собой синапсами. Структура нейронной сети пришла в мир разных наук прямиком из биологии. Благодаря такой структуре, например, машина обретает способность анализировать и даже запоминать различную информацию. Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти.

Основная цель создания нейронных сетей и вытекающая из этого задача обучения – это избавление от необходимости промежуточных вычислений-выводов при анализе профиля-матрицы входящих сигналов. Данная цель достигается за счет создания базы эталонных профилей, каждому из которых на выходе должен соответствовать единственный нейрон – ячейка результирующей матрицы. Каждому такому нейрону приписывается определенная трактовка-результат.

Нейронная сеть состоит из простейших вычислительных элементов – искусственных нейронов, связанных между собой. Каждый нейрон имеет несколько входных и одну выходную связь. В процессе работы нейронной сети значения входных переменных xixi передаются по межнейронным связям и умножаются на весовые коэффициенты wiwi, полученные значения взвешенно суммируются в нейроне. Основным достоинством нейронных сетей является возможность эффективно строить нелинейные зависимости, более точно описывающие наборы данных по сравнению с линейными методами статистики.

Данный обработчик позволяет задать структуру нейронной сети, определить ее параметры и обучить с помощью одного из доступных в системе алгоритмов. В результате будет получен эмулятор нейронной сети, который может быть использован для решения задач прогнозирования, классификации, поиска скрытых закономерностей, сжатия данных и многих других приложений.

В нейронных сетях нейроны объединяются в слои, при этом выходы нейронов предыдущего слоя являются входами нейронов следующего слоя. В каждом слое нейроны выполняют параллельную обработку данных.

Основная проблема, как с точки зрения решаемой задачи, так и собственно обучения – это зашумленность поступающих на анализ профилей-матриц возбужденных нейронов входящего слоя. Поэтому одним из основных условий является наличие качественной обучающей выборки. Если обучающая выборка низкого качества, то сильная зашумленность приведет к большому количеству ошибок. Впрочем, и большой размер обучающей выборки может привести к тому же результату.

В какой-то степени, работу нейронной сети можно сопоставить с работой безусловных рефлексов живых существ со всеми вытекающими недостатками.

Нейросети способны решить следующие виды задач:

- определить класс объекта и найти его фотографию;
- разделить полученные данные на группы со схожими признаками;
- выявить зависимость и обобщить данные;
- составить календарь предстоящих событий и т.д.

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является:

Классификация – распределение данных по параметрам. Например, на вход дается набор людей и нужно решить, кому из них давать кредит, а кому нет. Эту работу может сделать нейронная сеть, анализируя такую информацию как: возраст, платежеспособность, кредитная история и т.д.

Предсказание – возможность предсказывать следующий шаг. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

Распознавание – в настоящее время, самое широкое применение нейронных сетей. Используется в Google, когда вы ищете фото или в камерах телефонов, когда оно определяет положение вашего лица и выделяет его и многое другое.

II. Исследование генетических алгоритмов и нейросетей

Сегодняшние нейронные сети, являющиеся основой искусственного интеллекта для различных, в том числе и бытовых приборов, уже отчасти освоили подобные функции. Они не только способны следовать простым и заранее известным правилам (например, если хозяин приблизился к двери дома, надо ее открыть и подать подогретые тапочки), но и вырабатывать новые сложные знания, делать собственные открытия. Они могут быть отдельными устройствами, но чаще существуют в форме самообучающихся программ в памяти компьютеров.

Важно, что это не аналог мозга, а техническая идея, благодаря которой система может работать похожим образом. Пока ученые слишком мало знают о функционировании человеческого мозга, но уже понятно, что его особые качества определяются связями между нейронами. В структурах нейронных сетей ученые пытаются воспроизвести систему этих связей.

Первая схема базового принципа работы нейронных сетей основана на суммировании полученного опыта, напрямую не относящегося к решению задачи. Допустим, нейронной сети предъявили фотографии лгущих и говорящих правду людей, боящихся и смелых, злящихся или испытывающих другие эмоции. После некоторого обучения нейросеть может определить ложь, страх, злобу и другие эмоции по выражению лица, которое до этого не видела. Также нейросети могут производить знания для использования человеком и переносить их в другие программные продукты. Примерами таких задач являются медицинская диагностика и поиск наилучших вариантов лечения с учетом персональных особенностей пациента.

Искусственные нейронные сети прочно вошли в нашу жизнь и в настоящее время широко используются при решении самых разных задач и активно применяются там, где обычные алгоритмические решения оказываются неэффективными или вовсе невозможными. В числе задач, решение которых доверяют искусственным нейронным сетям, можно назвать следующие: распознавание текстов, игра на бирже, контекстная реклама в Интернете, фильтрация спама, проверка проведения подозрительных операций по банковским картам, системы безопасности и видеонаблюдения – и это далеко не все.

Генетический алгоритм – это компьютерная программа, которая якобы имитирует биологическую эволюцию.

Очевидно, что существует множество переменных, которые могут влиять на генетический алгоритм: это размер популяции, число поколений (итерации в алгоритме), методы слияния, функция приспособленности, то, как влияет приспособленность на шансы для производства потомства и сколько произойдет мутаций.

Также существуют и некоторые недостатки алгоритма. Он лучше работает, если функция приспособленности применяется к ДНК как набор битов. Иными словами, лучше, если ДНК представляет собой набор бинарных опций: да или нет. Голубые глаза? Карие глаза? Рыжие волосы? Черные волосы? Слияние родительских ДНК и последующие мутации не должны разрешать некоторые комбинации битов, так как получающаяся в результате ДНК не будет допустимым решением исходной задачи. Вспомним, что "ДНК" является ничем иным, как решением для математической формулировки приспособленности. Некоторые величины, используемые в этой формулировке, могут быть несправедливыми – например, деление на ноль.

Кроме того, генетический алгоритм не связан со временем. Вы отбираете количество поколений. Вы можете определить некоторую цель, например, "найти индивидуума с приспособленностью 0.99999" и остановиться, но тогда алгоритм никогда не закончит работу, так как он не сможет найти такого индивидуума. Если вы ставите нереальные цели или задаете слишком мало поколений, то могут возникнуть проблемы. Метод проб и ошибок и хорошее чутье являются лучшими помощниками в этой области.

Формула приспособленности должна выдавать число с плавающей точкой от 0 до 1. Могут быть использованы и другие диапазоны, но, по моему опыту, числа с плавающей точкой работают лучше. Вы можете захотеть интервал от 0 до 32767, если вы хотите для оптимизации использовать 7-битовое число для приспособленности.

Существуют три "хороших" способа выхода из генетического алгоритма. Во-первых, вы можете решить выйти из алгоритма, если больше нет разнообразия в фонде ДНК. Это, в действительности, хитрый тест, и модуль CPAN для нахождения протяженности строк может быть полезным в случае, если вы можете представлять ДНК как строку. Во-вторых, вы можете выйти, если вы достигли заданной приспособленности. Пока вы хорошо не поймете формулу приспособленности (для этого вам вообще может быть не нужен генетический алгоритм) постановка задач по приспособленности будет приводить к одному из двух результатов: или бесконечные циклы, или индивидуум, который является всего лишь "довольно хорошим". В-третьих, вы можете выйти после заданного числа итераций или "поколений".

На практике все эти три способа (или, по крайней мере, второй и третий) используются для контроля генетического алгоритма. Небольшое число прогонов, возможно 10 или 20, даст вам правильное ощущение того, как быстро сходится алгоритм и какую степень приспособленности можно ожидать. Они нашли применение в создании новых технических решений, например, электронных схем, которые отфильтровывают определённые частоты. Генетические алгоритмы

используют математические конструкции, которые сравнивают с мутациями (случайными изменениями в переменных или коэффициентах), естественным отбором (устранение изменений в схеме, которые, например, не приближают нас к цели – отклику на определённую частоту), и даже некоторого рода «рекомбинациями» (которые происходят при половом размножении). Поэтому некоторые апологеты эволюции утверждают, что генетические алгоритмы якобы показывают, что биологическая эволюция может создавать информацию, необходимую для продвижения от менее сложных к более сложным организмам (т.е. с увеличением количества генетической информации).

Генетические алгоритмы могут отбирать только по очень ограниченному числу признаков. Даже в случае с простейшими бактериями, которые не так уж и просты для того, чтобы бактерия была жизнеспособной (выжила) необходимы сотни признаков; отбор должен работать со *всеми* признаками, которые влияют на выживание.

Что-то *всегда* выживает, чтобы процесс продолжался. У эволюции нет правила, которое говорит, что некоторые организмы в эволюционирующей популяции остаются жизнеспособными, какие бы мутации не происходили. На самом деле, генетические алгоритмы, которые я рассматривал, искусственно сохраняют лучшее из предыдущего поколения и защищают его от мутаций или рекомбинации в случае, если ничего лучшего не производилось в следующей итерации. То есть происходит эффект храповика, что гарантирует, что генетический алгоритм произведёт желаемый результат – любой шаг в правильном направлении защищён.

Часто применяется идеальный отбор (коэффициент отбора, $s = 1{,}0$) то есть в каждом поколении только лучшее выживает и «размножается», производя следующее поколение. В настоящем мире реалистичными считаются коэффициенты отбора менее 0,01, и в этом случае потребуется много поколений, чтобы мутации, увеличившие количество информации, распространились по всей популяции. Другими словами, игнорируется *стоимость замещения*.

Обратной стороной этого является то, что «используются» высокие темпы воспроизводства. Бактерии могут только удвоить своё число за поколение. Многие «высшие» организмы могут размножаться немного быстрее, но генетические алгоритмы обычно производят от 100 до 1000 «потомков» за поколение. Например, если из популяции в 1000 бактерий выживет только одна (999 погибнет), то потребуется 10 поколений, чтобы вернуться к прежнему размеру популяции (1000 бактерий).

Время поколения игнорируется. Поколение может пройти в компьютере за микросекунды, в то время как деление даже самой «быстрой» бактерии займёт около 20 минут. Многоклеточные организмы имеют гораздо более долгий период поколения. Частота мутаций искусственно завышена (на много порядков). Это допустимо, поскольку «геномы» малы (см. следующий пункт), а также задействованы искусственные правила, например, чтобы защитить лучшие «организмы» от мутаций. Такие темпы мутаций у реальных организмов приведут к тому, что всё потомство станет нежизнеспособным (катастрофа ошибок). Вот почему живые существа обладают изысканными корректирующими механизмами для минимизации ошибок копирования (приблизительно до 1-ой ошибки на миллиард на 1 деление клетки).

«Геном» является неестественно маленьким и выполняет только одну задачу. Наименьший реальный геном состоит из более 0,5 млн пар оснований (и это – облигатный паразит, который зависит от своего хозяина, обеспечивающего его многими необходимыми субстратами), которые кодируют несколько сотен белков. Это эквивалентно более чем миллиону бит информации. Даже *если* генетический алгоритм может сгенерировать 1800 бит реальной информации, как в одном из наиболее часто рекламируемых примеров, это соответствует, возможно, одному небольшому ферменту – и это было достигнуто при совершенно нереалистичной частоте мутаций, временем жизни поколения, коэффициентами отбора, и т.д., и т.п. На самом деле, эти искусственно созданные условия больше похожи на то, как иммунная система организма развивает специфические антитела, в особенных условиях, *совершенно иных*, чем для любого целого организм.

В реальных организмах мутации происходят по всему геному, а не только в гене или участке, который определяет данный признак. Это означает, что все вредные изменения других признаков должны быть устранены одновременно с отбором редких желательных изменений в признаке, к которым мы стремимся. Это игнорируется в генетических алгоритмах. В генетических алгоритмах сама программа защищена от мутаций; только последовательности-мишени мутируют. Фактически, если бы она не была изолирована от мутаций, программа бы очень быстро рухнула. Однако воспроизводящие механизмы организма не защищены от мутаций.

Генетический алгоритм требует для своей работы следующих вещей:

1. **Функция оценки качества решения**. В терминологии эволюции – *выживаемость*. Использование данной функции позволит отличать более годные решения от менее годных.
2. **Начальный набор решений**. Это может быть просто случайный набор. В эволюции – изначальный набор видов (точнее, их цепочки ДНК).
3. **Генетические операторы**. Они обеспечивают «мутацию» отдельных решений и «скрещивание» решений друг между другом. Мутация – произвольное изменение решения. Скрещивание может представлять собой частичную комбинацию (кусок из одной «цепочки», кусок – из другой), применение какой-либо функции (сложение, умножение) и т.д.

В этом случае получаем примерный алгоритм:

Сформировать начальный набор решений (текущая популяция размером N).

1. Определить самое лучшее решение. Если оно нас устроит, то выход. Результат работы алгоритма – найденное решение.
2. Применить к текущей популяции генетические операторы (провести мутацию и скрещивание) и поместить результаты в новую популяцию.
3. Также добавить решения из текущей популяции в новую.
4. Сгенерировать несколько случайных решений и также добавить их в новую популяцию.
5. Отобрать N лучших решений из новой популяции и сделать их текущей популяцией, остальное отбросить.
6. Перейти к шагу 2.

В практическом применении данной методологии возможны следующие проблемы:

1. Недостаточный размер популяции. В этом случае решения, находящиеся в процессе «становления», будут отброшены слишком рано.
2. Чрезмерный размер популяции. Алгоритм будет работать слишком долго.
3. Плохой стартовый набор. Все *N* начальных решений должны быть максимально разнообразны.
4. Плохие генетические операторы. Если операторов мало или они не покрывают все разнообразные возможности мутации и комбинирования решений, то процесс поиска решений будет «застревать». Впрочем, отчасти это компенсируется шагом 5.

Генетические алгоритмы применяются в задачах построения расписаний, поиска глобальных оптимумов функций и др. Одним из наиболее важных преимуществ генетических алгоритмов является отсутствие необходимости информации о поведении функции и незначительное влияние возможных разрывов на процессы оптимизации. Также, как и в случае нейронных сетей, происходит уход от необходимости анализа причинно-следственных связей, путем построения «итогового» образа — целевой функции. В этом смысле, с точки зрения решения анализа текста, поиска генетические задачи решают такие же задачи или очень похожие, что и методы латентно-семантического анализа. При этом надо отдать должное, в вопросах семантического поиска и индексации текстов генетические алгоритмы имеют гораздо большие перспективы, по сравнению методами латентно-семантического анализа. С точки зрения распознавания образов, с очень сильной натяжкой целевую функцию можно сравнить со слоем входных нейронов и с ожиданием максимума как аналога максимизации сигнала нейрона выходного слоя. Хотя правильнее было бы говорить, что генетические алгоритмы используются для повышения эффективности обучения нейронных сетей, но все же не могут рассматриваться как конкуренция нейронными сетям. Задачи разные. Общий же недостаток – отсутствие индукционных алгоритмов – присутствует в полной мере.

III. Заключение

Эволюционные вычисления в широком смысле можно определить как область информатики, в которой используются вычислительные модели, аналогичные идеям Дарвиновского эволюционного процесса. Существует огромное множество методов оптимизации на основе эволюционных методов, у нейронных сетей много важных свойств, но ключевое из них – это способность к обучению. Обучение нейронной сети в первую очередь заключается в изменении «силы» синаптических связей между нейронами. Следующий пример наглядно это демонстрирует. В классическом опыте Павлова каждый раз непосредственно перед кормлением собаки звонил колокольчик. Собака достаточно быстро научилась ассоциировать звонок колокольчика с приемом пищи. Это явилось следствием того, что синаптические связи между участками головного мозга, ответственными за слух и слюнные железы, усилились. И в последующем возбуждение нейронной сети звуком колокольчика, стало приводить к более сильному слюноотделению у собаки. На сегодняшний день нейронные сети являются одним из приоритетных направлений исследований в области искусственного интеллекта. Генетические алгоритмы активно применяются в робототехнике, компьютерных играх, обучении нейронных сетей, создании моделей искусственной жизни, составлении расписаний, оптимизации запросов к базам данных, поиске оптимальных маршрутов и т.д. Такие алгоритмы могут стать хорошим помощником в бизнесе, сократить убытки и увеличить прибыль за счёт выбора оптимальных стратегий.

Список литературы

[1] Завгородний С.Д. Генетические алгоритмы и их применение [Текст] / С.Д. Завгородний, В.В. Швейкин, И.В. Танаев, Е.А. Дмитриев // Студенческая наука XXI века: материалы XII Междунар. студенч. науч.-практ. конф. (Чебоксары, 25 янв. 2017 г.)

[2] Звягин Л.С. Процесс обработки информации при реализации концепции "мягких" измерений // Международная конференция по мягким вычислениям и измерениям. 2017. Т. 1. С. 104-109.

[3] Звягин Л.С. Применение экспертного прогнозирования в системно-аналитических задачах // Экономика и управление: проблемы, решения. 2017. Т. 4. № 1. С. 86-93.

[4] Круг П.Г. Нейронные сети и нейрокомпьютеры: Учеб. пособие по курсу«Микропроцессоры». М.: Издательство МЭИ, 2002. 176 с.

[5] Панченко Т.В. Генетические алгоритмы [Текст]: учебно-методическое пособие / под ред. Ю.Ю. Тарасевича. Астрахань: Издательский дом «Астраханский университет», 2007. 87 [3] с.

[6] Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы / Пер. с польск. И.Д. Рудинского. М.: Горячая линия -Телеком, 2006. 452 с.

[7] Системный анализ в экономике и управлении предприятием: Учеб. пособие / Н.Г. Остроухова. Саратов: Издательство «КУБиК», 2014. 90 с.

[8] Zvyagin L.S. Process of information processing when realizing the concept of 'soft' measurements // Proceedings of 2017 XX IEEE International Conference on Soft Computing And Measurements (SCM-2017) 2017. С. 70-73.