УДК МИРОШНИЧЕНКО И. Д.

БОЛЬШИЕ ДАННЫЕ И ОБЗОР АЛГОРИТМОВ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ

Аннотация

Статья посвящена раскрытию сущности понятия "большие данные" и описанию функционирования алгоритмов обработки больших данных. Рассматриваются методы доступа и алгоритмы для решения плохо формализуемых задач.

ВВЕДЕНИЕ

К понятию *большие данные (Big Data)* относится информация, которую ввиду большого объёма уже невозможно хранить или обрабатывать традиционными способами, – это современное технологическое направление, связанное с обработкой крупных массивов данных, которые к тому же постоянно растут, например, потоков структурированных данных (предварительно обработанных), медиа или случайных объектов. Примерами Big Data могут служить данные метеорологических станций, получаемые из различных точек Земного шара, базы данных мобильных операторов мобильной связи, информация геолокаций камер движения ГБДД и подобные наборы информации

Понятно, что в современных условиях с такими данными и с таким объемом последовательные алгоритмы уже не справляются, на смену им приходят новые решения, базирующиеся на принципах параллельной обработки информации, поэтому для них разрабатываются свои алгоритмы, программные инструменты и даже вычислительные комплексы.

До сих пор четкого определения понятия для Big Data пока не сложилось, однако разработаны признаки, которым обрабатываемая информация и технология должны соответствовать. Все эти признаки начинаются с буквы V, поэтому система обозначается VVV. Аббревиатура такова:

- *Volume* — это объём. Объём информации *измерим*.
- *Velocity* — скорость. Объём информации не статичен, он постоянно увеличивается, причем, энтропия весьма велика, и инструменты обработки должны это учитывать.
- *Variety* — означает многообразие форматов информации: она может быть неструктурированной, частично или полностью структурированной.

К этим основным трём принципам добавляются дополнительные V:

- *Veracity* – достоверность,
- *Value* – ценность и
- *Viability* – жизнеспособность.

Сбор и анализ данных, например, обеспечивается поисковыми системами, мобильными операторами, крупными онлайн-компаниями, банками.

Облачные хранилища как крупные облачные data-центры становятся способом хранения больших объемов информации, доступной в любой момент пользователям в зависимости от уровня доступа при наличии Интернет-связи для различных гаджетов.

Блокчейн технология упрощает транзакции, хорошо справляется с обработкой операций между гигантским количеством контрагентов за счёт своего математического алгоритма, выполняемого на большом количестве параллельных блоков разной мощности.

Роботизация и промышленная автоматизация снижают расходы на ведение бизнеса (программы управления и учета, сбора информации, обнаружение мошенничества) и уменьшают стоимость товаров или услуг.

Искусственный интеллект и машинное обучение на основе нейронных сетей, генетических алгоритмов и нечетких вычислений помогают делать системы, эффективно функционирующие в науке и бизнесе.

Использование этих технологий приводит как к появлению новых сфер деятельности. В качестве таких примеров можно привести следующие.

Умный дом проанализирует обычное поведение человека в доме и сделает комфортной среду проживания, а также сможет спрогнозировать погоду на ближайшее время и оптимальный маршрут поездки.

Система дорожной безопасности повышается за счёт сбора данных о стиле вождения и нарушениях отдельных водителей, а также состояния их машин.

Автоматизированная технологическая линия, завод сам изменит линейку продукции, ориентируясь на анализ спроса, поставок, себестоимости и рыночной ситуации. Либо система принятия решений компании на основе анализа производства и каналов сбыта с учётом изменений реальной обстановки на рынке сможет оценить ситуацию и принять оптимальное решение.

Рассмотрим теперь техники и методы анализа и обработки больших данных, к основным и наиболее важным отнесем следующие.

Data Mining, объединяющий многочисленные методы и математический инструментарий в совокупности информационными технологиями.

Краудсорсинг — технология, которая позволяет получать данные одновременно из нескольких практически не ограниченных источников.

A/B(альфа/бета)-тестирование — такая итерационная технология проведения тестов для контрольной совокупности элементов в сравнении с другими подобными, но измененными по одному параметру, совокупностями, которая помогает определить, колебания какого из параметров оказывают наибольшее влияние на контрольную совокупность.

Аналитический прогноз, машинное обучение (искусственный интеллект) и сетевой анализ — технологии, которые сегодня также активно развиваются.

Рассмотрим одно из перечисленных направлений развития технологий, а именно, *Data mining*[1] (добыча знаний, интеллектуальный анализ данных, глубинный анализ данных). Глобально ставится следующая задача. В исходной достаточно крупной базе данных, содержащей

- *ранее неизвестные,* то есть такие знания, которые должны быть новыми (а не подтверждающими какие-то ранее полученные сведения);
- *нетривиальные,* то есть такие, которые нельзя просто так увидеть (при непосредственном визуальном анализе данных или при вычислении простых статистических характеристик);
- *практически полезные,* то есть такие знания, которые представляют ценность для пользователя *и*
- *доступных интерпретации,* то есть такие знания, которые легко представить в наглядной для пользователя форме и легко объяснить в терминах предметной области

знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Основу методов *Data Mining* составляют всевозможные методы *классификации, моделирования* и *прогнозирования*, основанные на применении таких математических аппаратов как построение *деревьев решений, конструирование искусственных нейронных сетей, генетических алгоритмов,* применения *эволюционного программирования,* методов *ассоциативной памяти, нечёткой логики.* Статистические методы (*корреляционный и регрессионный анализ, факторный анализ, дисперсионный анализ, компонентный анализ, дискриминантный анализ, анализ временных рядов)* и многие другие вероятностные методы также используются в ***Data Mining*** (например, в социологических исследованиях широко применяют язык обработки данных ***R,*** в системах, связанных с управлением, язык ***Python***). Некоторые из математических алгоритмов, представляющих аппарат обработки больших данных, рассмотрим подробнее ниже. Важно также отметить, что программные средства включают разнообразную визуализацию результатов исследований с помощью этих методов обработки больших данных, что дает возможность успешно использовать инструментарий специалистами прикладных областей, то есть людьми, не имеющими специальной математической или программистской подготовки.

Особенность исследования и постановки задач в этих областях заключается в их сложной системной организации реальных систем. Такие задачи относятся, главным образом, к плохо формулируемому уровню организации систем, закономерности которого не могут быть достаточно точно описаны на языке *статистических* или иных *аналитических математических моделей*. Данные неоднородны, гетерогенны, нестационарны и часто отличаются высокой размерностью.

Любая обработка данных, связанная с большими данными и методами data mining, проходит следующие этапы

1. постановка задачи анализа;
2. сбор данных;
3. подготовка данных (этот этап включает фильтрацию и дополнение данных, а также их кодирование);
4. выбор модели (конкретной парадигмы и алгоритма анализа данных, эффективного для выбранного подхода);
5. подбор параметров модели и алгоритма обучения;
6. обучение модели (автоматический поиск остальных параметров модели);
7. анализ качества обучения (если результат обучения модели окажется неудовлетворительным, сначала пытаются изменить выбор параметров обучения или алгоритма обучения и возвращаются к обучению модели (переход на п. 5), если это не помогает, выбирают другую модель (переход на п. 4);
8. анализ выявленных закономерностей (если закономерности выявить не удается, то нужно
   - либо снова изменить выбор параметров обучения или алгоритма обучения и возвратиться к обучению модели (переход на п. 5), если это не помогает, выбрать другую модель (переход на п. 4);
   - либо проверить постановку задачи анализа — уточнить, расширить, сузить (переход на п. 1)

Подготовка набора анализируемых данных нужна для того, чтобы в дальнейшем обнаружить присутствующие в данных закономерности, при этом, исходные данные, с одной стороны, должны иметь достаточный объём, чтобы эти закономерности можно было в них обнаружить, а с другой, — быть достаточно компактными, чтобы анализ занял приемлемое время. В таком случае, исходные данные удобно хранить и извлекать из специальных баз данных или хранилищ данных. В особенности, такая подготовка характерна для анализа многомерных данных для методов кластеризации или интеллектуального анализа данных.

Современные технологии позволяют хранить огромные объёмы (уже Петабайты) на всё меньшем кусочке пространства. Этот факт и развитие разнообразных датчиков приводит к тому, что уменьшаются затраты на сбор данных, однако возникают затраты на хранение.

[1] Термин ***Data Mining*** введён Григорием Пятецким-Шапиро в 1989 году.

Кроме того, большой объём данных вовсе не означает их бóльшую ценность. Объемы данных — это, скорее, характеристика *полноты* данных. Получение больших данных и их ценность — зависимость не линейная. С ростом объёмов видимая ценность на байт данных падает, потому что возникает очень много дублей записей и перезаписей, что приводит к падению *качества* исходного материала. Это происходит из-за роста технических качественных характеристик записывающих устройств. Оказывается, взаимодействие с большими данными открывает не только возможности, но и сталкивается с проблемой качества исходных данных. Еще одна проблема — данных генерируется настолько много в единицу времени, что их не успевают сохранять, приходится их квотировать при сохранении.

Так что проблема сохранения, передачи, обработки данных — сегодня очень важная проблема, например, данных собрали много, но возникают вопросы: какого они качества, как их собирали, были ли в них ошибки. Поэтому, осуществляя сбор данных или пользуясь уже существующими ресурсами (например, хранилищами), важно обращать внимание на следующие возможные особенности рассматриваемого процесса сбора:

- несоответствия цели сбора данных и цели их использования;
- отсутствие должного внимания к качеству собираемых данных;
- множество источников данных с неизвестной степенью истинности.

Когда уже собран большой объем данных, появляется проблема перемещения данных.

Чтобы понять, в чем состоит эта проблема, рассмотрим такой пример. Пусть некое хранилище, полностью используемое для хранения информации, получает 100Тб информации в день по каналу связи, который обеспечивает пропускную способность 1Гб/сек. К концу дня процесс сохранения информации закончится.

Если же необходимо её всю извлечь и в какой-то степени обработать, причем, не за один день, а за год, понадобится новый канал для чтения, так как первый канал постоянно будет занят записью информации следующего дня. Чтобы извлечь всю информацию за год, ширина этого канала для чтения должна быть в 365 раз больше существующего. Эти ограничения наводят на мысль, что нужно обрабатывать информацию прямо на тех серверах, где она была сохранена (например, можно было бы 70% мощности использовать для сохранения / записи информации, а 30% — использовать для предварительной обработки на тех же серверах.).

Такая идея лежит в основе технологий ***Hadoop*** и модели ***MapReduce***[2] (потому что присутствует 2 шага ***Map*** и ***Reduce***). Смысл ее состоит в следующем (рис. 1): для сохранения и обработки больших данных используются одни и те же мощности (принцип локальности).

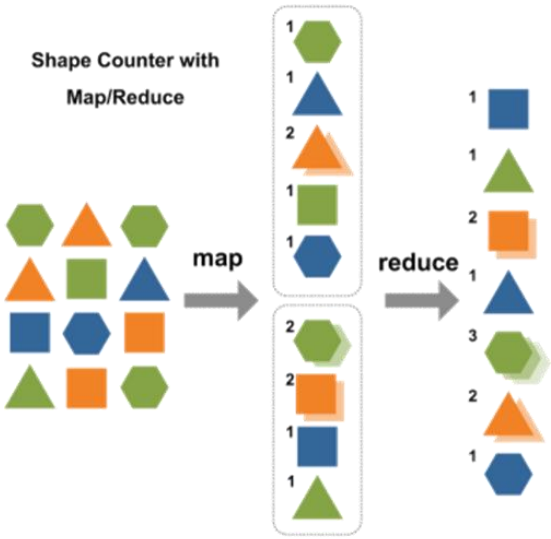


Рис. 1. *http://blog.jobbole.com/1321*

На шаге ***Map*** данные распределяются по процессорам (узлам) блоками и производится предварительная обработка данных. Для этого один из узлов — главный узел (*master node*) получает входные данные задачи, случайным образом разделяет их на части и передает рабочим узлам (*worker*) для предварительной обработки (например, на шаге ***Map*** определяет количество определенных фигурок на каждом узле).

На шаге ***Reduce (***сборка***)*** собираются предварительно обработанные данные. Главный узел получает ответы от рабочих узлов и на их основе формирует ответ (представляет разнообразие фигурок на основном узле). Технология ***Hadoop***[3] – наиболее популярная из технологий, реализующих принцип ***MapReduce.***

Осуществив начальный сбор информации, выполняется фильтрация выборки исходных данных, где они корректируются: удаляются выборки с шумами и удаляются или преобразуются выборки с пропущенными данными. Затем из отфильтрованных данных формируют объекты с требуемым набором признаков (или

[2] Вычислительная модель ***MapReduce*** (технология) была впервые предложена инженерами из Google Джеффри Дином и Сенджейем Гемаватом в 2003 году и изложена в 2004 г. в статье о ***MapReduce***.

3 Проект фонда *Apache Software Foundation*, свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов.

векторам, если алгоритм может работать только с векторами фиксированной размерности), убирая несущественные для данной модели, один набор признаков — на наблюдение.

Выделяют пять стандартных типов закономерностей, которые позволяют выявлять методы *Data Mining*:

- *ассоциация* (имеет место в том случае, если *несколько событий связаны друг с другом);*
- *последовательность* (существует цепочка связанных во времени событий);
- *классификация* (выявляются признаки, характеризующие группу, к которой принадлежит тот или иной объект, затем анализируются уже классифицированные объекты и формулируется некоторый набор правил);
- *кластеризация* (отличается от классификации тем, что сами группы заранее не заданы, в этом методе самостоятельно выделяют различные однородные группы данных);
- *прогнозирование* (основой для этого метода служит историческая *информация*, хранящаяся в БД в виде *временных рядов).*

Задача состоит в том, чтобы построить или найти закономерности, которые бы адекватно отражали динамику поведения целевых характеристик, тогда есть вероятность, что в результате конструирования одной или некоторых моделей можно предсказать и поведение системы в будущем. Часто для подтверждения лучшего решения такие конструируемые системы интегрируют в себе сразу несколько подходов, причем, в каждой системе имеется какая-то ключевая компонента, на которую делается главная ставка.

Следуя классическим определениям, представим типовые постановки этих задач.

Распознавание образов. Задача состоит в отнесении входного набора данных, представляющего распознаваемый объект, к одному из заранее известных классов. В число этих задач входит распознавание рукописных и печатных символов при оптическом вводе в ЭВМ, распознавание типов клеток крови, распознавание речи и другие.

Кластеризация данных. Задача состоит в группировке входных данных по присущей им "близости". Алгоритм определения близости данных (определение расстояния между векторами, вычисление коэффициента корреляции и другие способы) закладывается в нейросеть при ее построении. Сеть кластеризует данные на заранее не известное число кластеров. Наиболее известные применения кластеризации связаны со сжатием данных, анализом данных и поиском в них закономерностей.

Контекстно-адресуемая (ассоциативная) память. Эта память позволяет считывать содержимое по частичному или искаженному представлению входных данных. Основная область применения - мультимедийные базы данных.

Аппроксимация функций. Имеется набор экспериментальных данных $\mathbf{\{(X_1,y_1),..., (X_n,y_n)\}}$, представляющий значения $\mathbf{y_i}$ неизвестной функции от аргумента $\mathbf{X_i=(x_{i1},x_{i2},..., x_{im}),\ i = 1,..., n.}$ Требуется найти функцию, аппроксимирующую неизвестную и удовлетворяющую некоторым критериям. Эта задача актуальна при моделировании сложных систем и создании систем управления сложными динамическими объектами.

Предсказание. Имеется набор $\mathbf{\{y(t_1), y(t_2),..., y(t_n)\}}$ значений **y,** представляющих поведение системы в моменты времени $\mathbf{t_1, t_2,..., t_n}$. Требуется по предыдущему поведению системы предсказать ее поведение $\mathbf{y(t_{n+1})}$ в момент времени $\mathbf{t_{n+r}}$ Эта задача актуальна для управления складскими запасами, логистикой, для систем принятия решений.

Оптимизация. Цель этих задач — найти решение **NP**-полной проблемы, удовлетворяющее ряду ограничений и оптимизирующее значение целевой функции. К числу этих задач относится, например, *задача коммивояжера*. Для оптимизационных задач, относящихся к классу NP-полных, не существует другого *точного* метода решения, кроме полного перебора **n!** возможных вариантов решения, где **n** - размерность задач. Понятно, что с увеличением размерности ***n*** или числа точек рассматриваемого пространства время вычислений и объемы требуемой памяти растут экспоненциально. Поэтому для решения оптимизационных задач (и выше перечисленных) — это плохо формализуемые задачи — используются методы нахождения приемлемого *лучшего* решения в требуемых ограничениях.

Можно привести еще несколько направлений и подходов в решении плохо формализуемых задач, но в данной статье ограничимся некоторыми из выше приведенных, то есть, очень кратко рассмотрим представления о нейронных сетях, генетических алгоритмах и нечетких вычислениях

Теория искусственных нейронных сетей в течение лет 30 проходила этап формирования, а в настоящее время испытывает мощное развитие в самых разнообразных областях (в большой степени обработке больших данных с применением распараллеливания), что обусловливает разнообразие основных определений и постановок проблем исследований. Искусственные нейроны можно наделить разнообразными свойствами, которые позволяют получать решение актуальных для исследователя задач.

В рамках статьи рассмотрим только общие формулировки основных понятий нейронных сетей, (они вводятся без какой-либо связи со свойствами биологических нейронных сетей). Нейронная сеть — это не универсальный алгоритм, не жесткий алгоритм для конкретной задачи, это конструируемый алгоритм, причем, он может быть различен у различных разработчиков в виду разных причин: выбора основных характеристик для решения задачи, способа построения нейронной сети, методов обучения этой сети.

Нейронные сети.

Искусственный нейрон, далее просто *нейрон* j, $j \in \{1, 2, ...\}$, задается совокупностью своих входов $x_{j\,i}$ $i \in \{1, 2, ...\}$, весами входов $w_{j\,i}$, функцией состояния $S=\{s_j\}$, значением порога *a* (или порогов a_i) и функцией активации *f*.

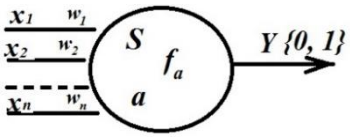


Рис. 2. Схема нейрона

Функция *состояния* определяет состояние нейрона в зависимости от значений его входов, весов входов и, возможно, предыдущих состояний. Наиболее часто используются функции состояния, не зависящие от предыдущего состояния и вычисляемые одним из двух способов: либо как сумма произведений значений входов на веса соответствующих входов по всем входам $s_j = \sum_{i=1}^{n(j)} x_{ij} w_{ji}$, (где n(j) - число входов нейрона j), либо как расстояние между вектором входов $\mathbf{X_j} = \{x_{j\,i}\}$ и вектором весов входов $W_j = \{w_{j\,i}\}$, измеряемое в какой-либо метрике, например, $s_j = \sum_{i=1}^{n(j)} \left| w_{ji} - x_{ij} \right|$.

Функция активации $y = f(s)$ определяет выходной сигнал нейрона как функцию от его состояния s. В качестве функций активации чаще всего берут наиболее распространенные, а именно, пороговую, линейную, линейную пороговую, ступенчатую, сигмоидную, гиперболический тангенс, арктангенс, а также гауссиану.

Нейронная сеть конструируется из слоев одинаковых нейронов путем соединения ориентированными взвешенными весами ребер выходов одного слоя нейронов со входами другого слоя. При этом граф межнейронных соединений может быть как ациклическим, так и произвольным циклическим. Вид графа служит одним из классификационных признаков типа нейронной сети, различают

сети с прямыми связями (рис. 3),
сети с перекрестными связями (рис.4) и
сети с обратными связями (рис. 5 и 6).

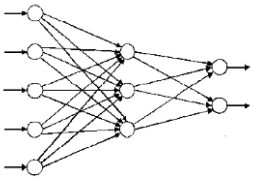


Рис. 3. Сети с прямыми связями

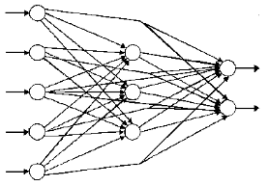


Рис. 4. Сети с перекрестными связями

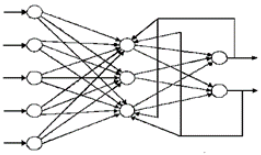


Рис. 5. Сети с обратными связями

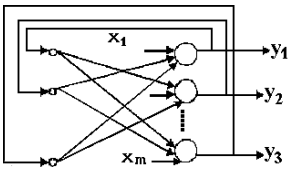


Рис. 6. Сеть Хопфилда

На этих рисунках видно, что сети с прямыми связями имеют соединения только между соседними слоями нейронов, сети с перекрестными связями могут иметь соединения между нейроном любого предыдущего слоя и нейроном любого последующего слоя, а сети с обратными связями имеют соединения между нейронами последующих и предыдущих слоев.

Приняв некоторое соглашение о тактировании сети (времени срабатывания нейронов), получим аппарат для задания алгоритмов посредством нейронных сетей. Следует, однако, понимать, что

- форматы представления значений
  - значений весов связей,
  - значений входов и
  - точность вычислений значений функций состояния и
- значения активации и виды функций активации нейронов сети

накладывают ограничения на классы решаемых задач. Поэтому при использовании *малоразрядных* форматов с фиксированной точкой, например, возможно эффективное решение задач распознавания, но, скорее всего, нецелесообразно решение задач *математической физики*.

Сети могут быть *конструируемыми* (сеть Хопфилда, рис. 6) или *обучаемыми*.

Суть функционирования конструируемой сети в том, что изначально некоторым способом определяется набор устойчивых состояний сети. Затем при подаче на входы частичной или ошибочной входной последовательности сеть через какое-то время сеть переходит в одно из устойчивых состояний, предусмотренных при ее конструировании, а *на входах* сети появляется набор *из запомненных устойчивых наборов, признаваемых сетью как наиболее близкий* к изначально поданному на вход.

В обучаемых сетях графы межнейронных связей и веса входов изменяются при выполнении алгоритма обучения. Алгоритмы обучения сети делятся на *наблюдаемые (с учителем), ненаблюдаемые (без учителя) и смешанные (гибридные).*

Первые при обучении сравнивают *заранее известный выход* с получившимся значением, вторые обучаются, *не зная заранее правильных выходных значений*, но группируя "близкие" входные векторы так, чтобы они формировали один и тот же выход сети. В обоих случаях обучение основывается на сравнении результатов, полученных сетью на исходной выборке примеров, с результатами, полученными экспериментально на той же выборке. При неудовлетворительном результате итеративно вводится весьма

малая поправка в алгоритме обучения, где подправляются значения входов (или выходов) промежуточных слоев в архитектуре графа, до получения приемлемого результата.

Обозначим обучающий набор примеров $\mathbf{X_1, X_2, \ldots, X_m}$ $\mathbf{(X_1=(x_{11}, .., x_{1n}), X_2=(x_{21}, \ldots, x_{2n}), ..., X_m=(x_{m1}, .., x_{mn}),}$ где $\mathbf{X_j = (x_{j1}, ..., x_{jn})}$ входные значения **j**-го примера, а $\mathbf{D_j}$ — соответствующее $\mathbf{X_j}$, *экспериментальное* выходное значение при подаче на входы **j**-го примера, **j=l,...,m.** Считается, что сеть правильно обучена, если выполняется *критерий окончания обучения.* В качестве этих критериев обычно используют следующие (линейный или квадратичный соответственно), хотя могут быть и другие:

1. $max \mid D_j - Y_j \mid < \delta,$ для всех j где **δ -** заданная величина ошибки;
   Y_j **-** выходное значение, выдаваемое сетью при подаче на ее входы *j*-го примера, *j=l, ..., m.*
2. $\sqrt{\sum_j (D_j - Y_j)^2} \le \delta.$

В задачах, эффективно решаемых нейросетями, *точки многомерного пространства*, в котором сформулирована задача, образуют *области точек, обладающих одним и тем же свойством*, например, принадлежащих одному классу объектов, имеющих одинаковое значение заданной на них некоторой функции, и так далее. Нейронные сети *запоминают подобные области, а не отдельные точки,* представляющие предъявленные при обучении примеры. В ходе функционирования *сеть относит предъявленный на ее входы набор значений к той или иной области*, что и является искомым результатом.

У нейросетевого подхода все же есть и недостатки. Основной их них — необходимость иметь очень большой объем обучающей выборки. Другой существенный недостаток в том, что даже натренированная нейронная сеть представляет собой черный ящик: знания, зафиксированные как веса нескольких сотен межнейронных связей, совершенно не поддаются анализу и интерпретации человеком.

Генетические алгоритмы.

Data Mining не основная область применения генетических алгоритмов, область применения генетических алгоритмов достаточно обширна. Но методы, используемые в генетических алгоритмах, входят в стандартный инструментарий методов ***Data Mining***, они успешно используются для решения ряда больших и экономически значимых задач в бизнесе и инженерных разработках, с их помощью были разработаны промышленные проектные решения, позволившие сэкономить многомиллионные суммы (в финансовых компаниях). Генетические алгоритмы обычно используются

- для оценки значений непрерывных параметров моделей большой размерности,
- для решения комбинаторных задач,
- для оптимизации моделей, включающих одновременно непрерывные и дискретные параметры,
- для использования в системах извлечения *новых знаний из больших баз данных*,
- при создании и обучении стохастических сетей, обучении нейронных сетей, в системах принятия решений
- при оценке параметров в задачах многомерного статистического анализа,
- при получении исходных данных для работы других алгоритмов поиска и оптимизации.

Генетические алгоритмы имеют целью нахождение лучшего по сравнению с имеющимся, а не оптимальным решением задачи. Можно выделить основные преимущества применения генетических алгоритмов по сравнению с применением традиционных методов.

1. Генетические алгоритмы работают *с кодами,* в которых *представлен набор параметров, напрямую зависящих от аргументов целевой функции* (причем интерпретация этих кодов происходит только перед началом работы алгоритма и после завершения его работы для получения результата). Код рассматривается просто как битовая строка, и в процессе работы преобразования кодов происходит совершенно независимо от их интерпретации.
2. В процессе поиска лучшего решения *генетический алгоритм* использует *несколько точек поискового пространства одновременно*, а не *переходит* от точки к точке, как это делается в традиционных методах. Процесс использования нескольких точек одновременно позволяет преодолеть опасность попадания в локальный экстремум целевой функции, если она имеет несколько экстремумов.
3. *Генетические алгоритмы* в процессе работы *не используют никакой дополнительной информации:* единственной используемой информацией является *область допустимых значений параметров* и *целевой функции* в произвольной точке. Это повышает скорость работы.
4. *Генетический алгоритм* использует как *вероятностные правила* для порождения новых точек анализа, так и *детерминированные правила* для перехода от одних точек к другим. Одновременное использование элементов случайности и детерминированности дает значительно больший эффект, чем раздельное.

Генетический алгоритм работает *с кодами, не учитывая их смысловую интерпретацию*. Поэтому сам код и его структура описываются понятием *генотип*, а его интерпретация, с точки зрения решаемой задачи, понятием — *фенотип*. Каждый код представляет, по сути, точку пространства поиска. Можно привести такой пример: нужно закодировать результаты анкеты, где каждый вопрос представляет либо ответ «да, нет», либо гораздо большую градацию, которая представлена выбором номера ответа. Ответы на каждую анкету можно представить строкой либо двоичного кода (в первом случае), либо десятичных цифр (во втором случае). С

целью максимально приблизиться к биологическим терминам, экземпляр кода (строку в примере) называют *хромосомой, особью* или *индивидуумом*.

На каждом шаге работы генетический алгоритм использует несколько точек поиска одновременно. *Совокупность этих точек* является набором особей, который называется ***популяцией***. *Количество особей в популяции* называют *размером популяции*. В классических генетических алгоритмах размер популяции является фиксированным и представляет одну из характеристик генетического алгоритма. На каждом шаге работы генетический алгоритм обновляет популяцию, создавая новые особи и уничтожая ненужные с помощью специальных функций. Популяцию на каждом из шагов и сами эти шаги помечают номером и называют *поколениями*. Например, популяция, полученная из исходной популяции после первого шага работы алгоритма, будет первым поколением, после следующего шага - вторым и т. д.

Генерация новых особей происходит на основе *моделирования процесса размножения*. Порождающие особи называются *родителями*, а порожденные — *потомками* (родительская пара, как правило, порождает пару потомков). Непосредственная генерация новых кодовых строк из двух выбранных (родительских) происходит за счет работы *оператора скрещивания*. При порождении новой популяции оператор скрещивания может применяться не ко всем парам родителей, часть этих пар может переходить в популяцию следующего поколения непосредственно. Насколько часто будет возникать такая ситуация, зависит от значения *вероятности применения* оператора скрещивания, который является одним из параметров генетического алгоритма.

Моделирование процесса *мутации новых особей* осуществляется за счет работы *оператора мутации*. Основным параметром оператора мутации также является *вероятность* мутации. Так как ввели соглашение о том, что размер популяции фиксирован, то порождение потомков должно сопровождаться уничтожением других особей. Выбор пар родителей из популяции для порождения потомков производит *оператор отбора*, а выбор особей для уничтожения — *оператор редукции*. Основным параметром их работы является, как правило, качество особи, которое определяется значением *целевой функции* в точке пространства поиска, описываемой этой особью.

Для окончания работы генетического алгоритма, как и нейросети, нужно сформировать критерий останова работы генетического алгоритма. Этим критерием может быть одно из трех событий:

1. сформировано заданное пользователем число поколений,
2. популяция достигла заданного пользователем качества (например, значение качества всех особей превысило заданный порог),
3. достигнут некоторый уровень сходимости (то есть, особи в популяции стали настолько подобными, что дальнейшее их улучшение происходит чрезвычайно медленно).

Отметим, что характеристики генетического алгоритма выбираются таким образом, чтобы обеспечить малое время работы, с одной стороны, и поиск как можно лучшего решения, с другой.

Немного конкретизируем процесс функционирования генетического алгоритма для лучшего понимания. Для формирования исходной популяции используется, как правило, какой-либо случайный закон, на основе которого выбирается нужное количество точек поискового пространства (исходная популяция может быть результатом работы какого-либо другого алгоритма оптимизации, все здесь зависит от разработчика конкретного генетического алгоритма).

В основе *оператора отбора*, который служит для выбора родительских пар и уничтожения особей, лежит принцип *"выживает сильнейший"*. В качестве примера можно привести следующий оператор: выбор особи для размножения производится случайно, а вероятность участия особи в процессе размножения вычисляется по формуле: $P_i = f_i / \sum_{j=1}^{n} f_j$, где P_i — вероятность участия особи в процессе размножения, i — номер особи, n — размер популяции, f_i — значение целевой функции для n-ной особи. Целью поиска в нашем случае будет являться нахождение максимума целевой функции. Одна особь может быть задействована в нескольких родительских парах.

Аналогично решается вопрос *уничтожения особей*. Вероятность уничтожения, соответственно, должна быть обратно пропорциональна качеству особей. Однако обычно происходит просто уничтожение особей с наихудшим качеством. Таким образом, выбирая для размножения наиболее качественные особи и уничтожая наиболее слабые, генетический алгоритм постоянно улучшает популяцию, приводя к формированию лучших решений.

Теперь, собственно о моделировании процесса *размножения*. *Оператор скрещивания*, моделируя природный процесс наследования, представляет обеспечение передачи свойств родителей потомкам.

Приведем пример простейшего *оператора скрещивания*, который выполняется в два этапа. Пусть особь представляет собой строку из ***m*** элементов. Сначала равновероятно выбирается натуральное число k от 1 до $m-1$, представляющий место разбиения строки на две части (это число называется точкой разбиения), и обе исходные (родительские) строки разбиваются на две подстроки в соответствии с точкой разбиения. Затем родительские строки обмениваются своими подстроками, лежащими после точки разбиения, то есть элементами с $k+1$-го по m-й: получаются две новые строки, которые наследовали частично свойства обоих родителей. Этот процесс проиллюстрирован ниже.

строка1	родитель 1: $X_1X_2...X_kX_{k+1}.. .X_m$	потомок 1:$X_1X_2...X_k\boldsymbol{Y_{k+1}...Y_m}$
строка2	родитель 2: $Y_1Y_2...Y_kY_{k+1}...Y_m$	потомок 2: $Y_1Y_2...Y_k\boldsymbol{X_{k+1}...X_m}$

Вероятность применения *оператора скрещивания* обычно выбирается достаточно большой, в пределах от 0,9 до 1, чтобы обеспечить постоянное появление новых особей, расширяющих пространство поиска. При значении вероятности меньше 1 часто используют *элитизм*: особую стратегию, которая предполагает переход в популяцию следующего поколения лучших особей текущей популяции, без каких-либо изменений. Применение *элитизма* способствует сохранению общего качества популяции на высоком уровне. В таком случае, обычно элитные особи участвуют еще и в процессе отбора родителей для последующего скрещивания. Количество элитных особей определяется обычно по формуле: $K = (1 - P) * N$, где K — количество элитных особей, P — вероятность применения оператора скрещивания, N — размер популяции. При *элитизме* все выбранные родительские пары подвергаются скрещиванию, несмотря на то, что вероятность применения оператора скрещивания меньше 1(это позволяет сохранять размер популяции постоянным).

Оператор *мутации* нужен для *моделирования* природного *процесса мутации* и применяется в генетических алгоритмах по следующим соображениям. Ввиду того, что исходная популяция (какой бы большой она ни была) охватывает ограниченную область пространства поиска. Оператор скрещивания расширяет эту область, но все же до определенной степени, так как использует ограниченный набор значений, заданный исходной популяцией. Необходимо внесение случайных изменений в особи (как и в реальной жизни): это позволяет преодолеть указанное ограничение и иногда значительно сократить время поиска и улучшить качество результата.

Как правило, *вероятность мутации*, в отличие от *вероятности скрещивания*, выбирается достаточно малой (0,01 – 0,1). Сам *процесс мутации* заключается в *замене одного из элементов кода строки на другое значение*. Это может быть

- перестановка двух элементов кода в строке,
- замена элемента кода строки значением элемента кода из другой строки,
- в случае битовой строки может применяться инверсия одного из битов и тому подобное.

В процессе работы алгоритма все операторы применяются многократно и ведут к постепенному изменению исходной популяции. Все операторы (отбора, скрещивания, мутации и редукции) по своему построению направлены на улучшение каждой отдельной особи, поэтому результатом их работы оказывается постепенное улучшение популяции в целом, что и приводит к *улучшению популяции решений по сравнению с исходной популяцией*.

По завершению работы генетического алгоритма из конечной популяции выбирается та особь, которая дает экстремальное (максимальное или минимальное) значение целевой функции и является, таким образом, результатом работы генетического алгоритма (полученный результат представляет собой улучшенное решение, потому что конечная популяция лучше исходной).

Генетические алгоритмы удобны тем, что их легко *распараллеливать*. Например, можно разбить поколение на несколько групп и работать с каждой из них независимо, обмениваясь время от времени несколькими хромосомами. Существуют также и другие методы распараллеливания генетических алгоритмов.

Генетические алгоритмы имеют, конечно, и недостатки. Критерий отбора хромосом и используемые операторы являются эвристическими и не всегда гарантируют нахождения “лучшего” решения. Как и в реальной жизни, эволюцию может “зациклить” на какой-либо непродуктивной ветви. И, наоборот, можно привести примеры, когда два неперспективных родителя, которые могут быть исключены из эволюции генетическим алгоритмом, оказываются способными произвести высокоэффективного потомка. Это особенно становится заметно при решении высокоразмерных задач со сложными внутренними связями.

Нечеткие вычисления.

Нечеткие вычисления базируются на аппарате нечеткой логики, свойствах нечетких множеств, операциях и отображениях над ними и представляют строгий и мощный математический аппарат. Подробное и точное описание этого математического аппарата и способов применения его с конкретными примерами требует значительного объема, строгих математических определений и вряд ли интересно в этой обзорной статье. Тем не менее, попытаемся доступно показать важные особенности функционирования объектов нечетких вычислений.

Прежде всего, покажем, чем отличается множество от нечеткого множества на примере отрезка и нечеткого отрезка и понятия человек высокого (маленького) роста. Нам известно классическое представление отрезка на прямой (все точки прямой между b и c, включая сами граничные точки b и c). Понятие нечеткого отрезка определяет у отрезка bc в качестве границ не конкретные точки, а линии, описываемые, так называемыми, функциями принадлежности (на рисунке 7 левая граница — это линия с основанием ab, правая граница — линия с основанием cd), аналогично, число b классически представлено точкой на прямой, а нечеткое число b представляет собой треугольник с левой ab и правой cd нечеткими границами.

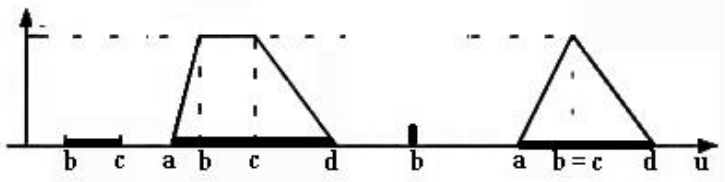


Рис 7. Схематичное представление о нечетких объектах (отрезка и числа)

Попытаемся определить на обыденном уровне понятия множеств маленький и большой рост человека, интерпретированные на рисунке 8.

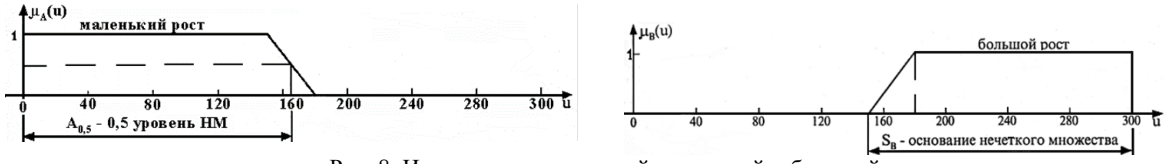


Рис. 8. Интерпретация понятий маленький и большой рост.

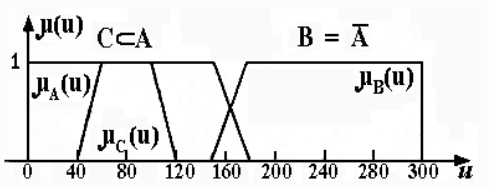
В соответствии с рисунком 8 все люди точно (с вероятностью 1) считают человеком маленького роста человека, если его рост от 0 до 150см, аналогично, все люди точно не считают человеком маленького роста человека, рост которого от 180 см и выше. А вот человека, рост которого находится в промежутке от 150 до 180 см человеком маленького роста считает разное количество человек (а поэтому имеет правую нечеткую границу), аналогично человеком высокого роста считает также разное количество человек (а поэтому имеет левую нечеткую границу).

К таким множествам можно применить все операции, которые определены для классических множеств: дополнение, объединение, разность, пересечение. Определяются эти операции с использованием функций принадлежности (аналогом вероятности). Например, (упрощенно) *объединением нечетких множеств* ***A*** и ***C*** называется *нечеткое множество* $D = A \cup C$, имеющее *функцию принадлежности:* $\mu_D(u)=max\ (\mu_A(u), \mu_C(u))$.

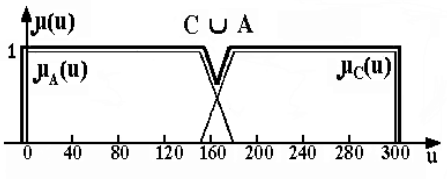
При этом существует несколько альтернативных вариантов реализации *объединения* нечетких множеств, использующих суммирование (выбор остается за разработчиком проекта):

$\mu_D(u) = min\ (1, \mu_A(u) + \mu_C(u))$,

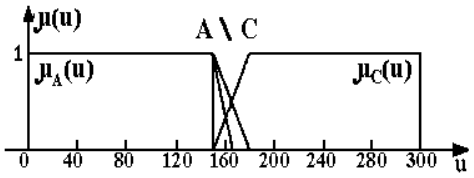
$\mu_D(u) = \mu_A(u) + \mu_C(u) - \mu_A(u) * \mu_C(u)$



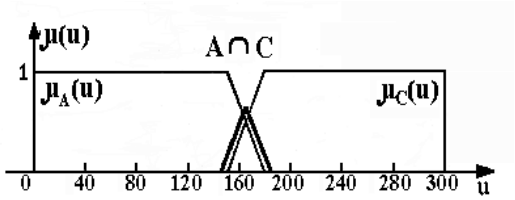
а) *Подмножество и дополнение нечеткого множества*



а) *Объединение нечетких множеств*



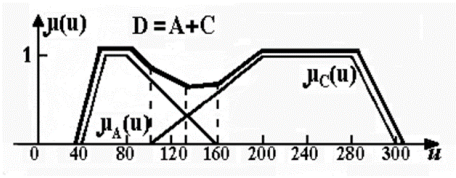
а) *Разность нечетких множеств*



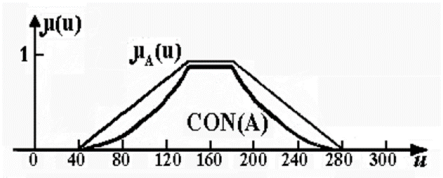
а) *Пересечение нечетких множеств*

Рис. 9. *Операции с нечеткими множествами*

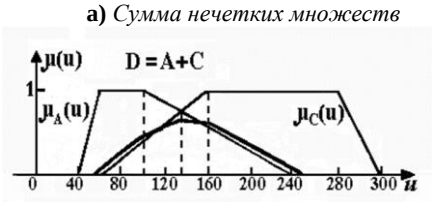
А также некоторые дополнительные операции: сумма, скалярное произведение, концентрирование и расширение нечетких множеств.



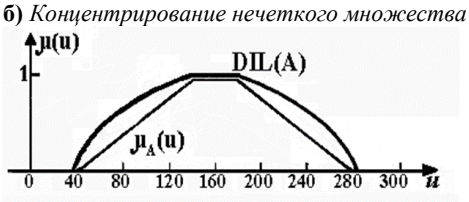
а) *Сумма нечетких множеств*



б) *Концентрирование нечеткого множества*



в) *Скалярное произведение нечетких множеств*



г) *Расширение нечетких множеств*

Рис. 10. *Специальные операции с нечеткими множествами*

Суммой *нечетких множеств* ***A*** и ***C*** называется *нечеткое множество* $D = A + C$, имеющее *функцию принадлежности:* $\mu_D(u) = min\ (1, \mu_A(u) + \mu_C(u))$,

Скалярным произведением *нечетких множеств* ***A*** и ***C*** называется *нечеткое множество* $D = A \bullet C$, имеющее *функцию принадлежности:* $\mu_D(u) = \mu_A(u) \bullet \mu_C(u)$,

Степенью α нечеткого множества ***A*** называется *нечеткое множество* $D = A^{\alpha}$, имеющее *функцию принадлежности:* $\mu_D(u) = (\mu_A(u))^{\alpha}$,

Концентрированием нечеткого множества ***A*** называется *нечеткое множество* $D = CON(A) = A^2$.

*Расширением нечеткого множества **A** называется нечеткое множество* $\boldsymbol{D = DIL(A) = A^{0,5}}$.

Далее определяются *нечеткие отношения*, исследования свойств которых лежат в основе *нечеткой алгебры* или *нечетких вычислений*, когда арифметические операции производятся не с обычными числами, а с *особой формы нечеткими множествами — нечеткими числами* (о которых говорили выше).

Для нечетких чисел разработаны алгоритмы выполнения следующих операций: вычисления одномерной функции, изменения знака, сложения, получения обратного числа, вычитания, умножения, деления, сравнения, сопоставления.

Можно показать, что операции сложения и умножения *нечетких чисел* обладают свойствами *коммутативности, ассоциативности* и *дистрибутивности умножения* относительно *сложения*. Для *сложения* и *умножения* действуют правила, на правила для "обычных" нуля и единицы.

Одной из важнейших операций с нечеткими числами является их *сравнение*. При этом результат будет очевидным лишь в том случае, если основания сравниваемых нечетких чисел M и N не пересекаются. В этом случае большим будет считаться то нечеткое число, чье основание на действительной оси расположено правее. Однако в большинстве случаев основания нечетких чисел пересекаются, поэтому их сравнение происходит путем вычисления специальной меры для каждого из чисел, которые затем сравниваются. Чаще всего используются два наиболее важных способа вычисления мер:

- поиск точки максимума;
- поиск "центра тяжести".

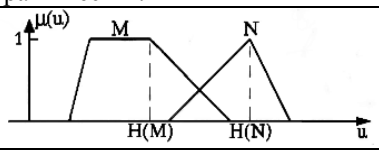


Рис. 11. *Сравнение по максимуму функций принадлежности*

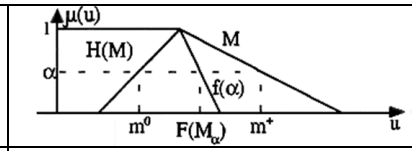


Рис. 12. *Сравнение по "центру тяжести" функций принадлежности*

Системы построения *нечетких управляющих систем* базируются на описанном аппарате нечетких множеств (чисел) и нечетких отношений. Число элементов *нечеткой управляющей системы* бесконечно велико, поэтому невозможно задать правила *нечеткого вывода* соответствующими парами точек. Однако они могут быть описаны в терминах *теории нечетких множеств*. Например, вполне работоспособная система кондиционирования может быть описана правилами:

"если температура в комнате высокая, то скорость вращения вентилятора высокая" и

"если температура в комнате низкая, то скорость вращения вентилятора низкая".

Подобные *правила вывода* используются в нечетких системах управления.

Нечеткие правила вывода образуют *базу правил*. Важно то, что в *нечеткой управляющей системе* в отличие от традиционной работают все правила *одновременно*, но степень их влияния на выход может быть различной. Принцип вычисления *суперпозиции* многих влияний на окончательный результат лежит в основе *нечетких управляющих систем*.

Процесс обработки нечетких правил вывода в управляющей системе состоит из 4 этапов:

1. Вычисление степени истинности левых частей правил (между "если" и "то"): определение *степени принадлежности* входных значений *нечетким подмножествам*, указанным в *левой* части *правил вывода*.
2. Модификация *нечетких подмножеств*, указанных в *правой* части *правил вывода* (после "то"), в соответствии со значениями истинности, полученными на первом этапе.
3. *Объединение (суперпозиция)* модифицированных подмножеств. Традиционно *суперпозиция* нескольких функций принадлежности нечетких множеств определяется как их *объединение* или *суммирование*.
4. *Скаляризация* результата *суперпозиции*: переход от *нечетких подмножеств* к скалярным значениям. Процесс преобразования *нечеткого множества* в единственное значение называется *"скаляризацией"*. Чаще всего в качестве такого значения используется *"центр тяжести"* функции принадлежности нечеткого множества, другой распространенный подход — использование *максимального* значения *функции принадлежности*. Конкретный выбор методов *суперпозиции* и *скаляризации* осуществляется в зависимости от желаемого поведения нечеткой управляющей системы.

ЗАКЛЮЧЕНИЕ

В статье рассмотрено современное представление о больших данных и его основных характеристиках методы и направления анализа и обработки. Особое место отведено направлению развитию технологий Data Mining, рассмотрены особенности функционирования методов этой технологии на различных этапах, представлены типовые постановки задач, в которых реализуются методы Data Mining.

Более подробно описаны методы, особенности, преимущества и недостатки наиболее мощно развивающихся направлений обработки больших данных: нейронные сети, генетические алгоритмы и нечеткие вычисления (представляющие базу для нечетких управляющих систем).

Все это позволит обучающимся разных уровней расширить свой кругозор в представлении о развитии современных информационно-коммуникационных технологий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мэтью Рассел, Михаил Классен. Data Mining. Извлечение информации из Facebook, Twitter, LinkedIn, Instagram, GitHub. Изд.-во: Питер.2020, 464 с.

2. В. Дюк. Data Mining – интеллектуальный анализ данных. https://blog.iteam.ru/data-mining-intellektualnyj-analiz-dannyh/