Московский авиационный институт
(государственный ТЕХнический университет)

Факультет прикладной математики

Кафедра вычислительной математики и программирования

Реферат по курсу «Искусственный интеллект»

«Data Mining. Генетические алгоритмы»

Преподаватель: Д. В. Сошников
Студент: И. К. Никитин

Москва, 2010

Содержание

Введение

Data mining — технология выявления *скрытых* **внутри** больших информационных объемов Термин получил своё название из двух понятий:

- поиска ценной информации в большой базе данных (data);
- добычи горной руды (mining).

Как и в случае с горнодобывающей промышленностью, Data mining требует «просеивания огромного количества сырого».

> Data Mining часто переводится как добыча данных, извлечение информации, раскопка данных, интеллектуальный анализ данных, средства поиска закономерностей, извлечение знаний, анализ шаблонов, «извлечение зерен знаний из гор данных», раскопка знаний в базах данных, информационная проходка данных, «промывание» данных.

Само понятие появилось во последней четверти XX века. На данный момент понятие приобрело высокую популярность и часто употребляется, иногда и без всякого осмысления. Развитию Data Mining значительно помогло, то что его как компоненту стали включать в себя многие коммерческие корпоративные информационные системы. До информационного бума конца XX века[1] обработка и анализ данных осуществлялся в рамках прикладной статистики, на маленьких базах данных. Часто Data Mining используется в областях:

- промышленное производство;
- электронная коммерция;
- торговля;
- игра на бирже и инвестиции;
- лингвистические исследования;
- медицина.

[1]Согласно оценке профессоров университета Berkley в только 2002 объем информации в мире увеличился в $5 \cdot 10^{12}$ байт. Согласно другим исследованиям объем информации утраивается каждые 2 года.

1. Свойства знаний

Для полноты картины приведем классическое определение[2]:

> « **Data mining** — исследование и обнаружение «машиной» (алгоритмами, средствами искусственного интеллекта) в сырых данных скрытых знаний, которые ранее не были известны, нетривиальны, практически полезны, доступны для интерпретации человеком. »

Знания должны быть новые, ранее неизвестные. Затраченные усилия на поиск знаний, которые уже известны, не окупаются. Ценность представляют только новые, ранее неизвестные знания.

Знания должны быть нетривиальны. Результаты анализа должны отражать неочевидные, неожиданные закономерности в данных, составляющие называемые *скрытые знания*. Если результаты, получены более простыми способами (например, визуальным просмотром), они не оправдывают привлечение мощных методов Data Mining.

Знания должны быть практически полезны. Полезность заключается в том, чтобы эти знания могли принести определенную выгоду при их применении. Полученные знания должны быть применимы, в том числе и на новых данных, с достаточно высокой степенью достоверности.

Знания должны быть доступны для понимания человеку. Найденные закономерности должны быть логически объяснимы, в противном случае существует вероятность, что они являются случайными. Обнаруженные знания должны быть представлены в понятном для человека виде.

[2] Пятецкий-Шапиро, (1996 г.) один из основателей этого направления.

2. Методы Data Mining

К базовым методам Data Mining принято относить прежде всего алгоритмы, основанные на переборе. Простой перебор всех исследуемых объектов требует $O(2^N)$ операций, где N — количество объектов. Следовательно, с увеличением количества данных объем вычислений растет экспоненциально, что при большом объеме делает решение любой задачи таким методом практически невозможным.

Для сокращения вычислительной сложности в таких алгоритмах, как правило, используют разного вида эвристики, приводящие к сокращению перебора. Оптимизация подобных алгоритмов сводится к приведению зависимости количества операций от количества исследуемых данных к функции линейного вида. В то же время, зависимость от количества атрибутов, как правило, остается экспоненциальной. При условии, что их немного (в подавляющем большинстве случаев их значительно меньше, чем данных), такая зависимость является приемлемой.

Основным достоинством данных алгоритмов является их простота, как с точки зрения понимания, так и реализации. К недостаткам можно отнести отсутствие формальной теории, на основании которой строятся такие алгоритмы, а следовательно, сложности, связанные с их исследованием и развитием.

К базовым методам можно отнести также и подходы, использующие элементы теории статистики. В связи с тем, что Data Mining является развитием статистики, таких методов достаточно много. Основная их идея сводится к корреляционному, регрессионному и другим видам статистического анализа. Основным недостатком является усреднение значений, что приводит к потере информативности данных. Это в свою очередь приводит к уменьшению количества добываемых знаний.

Большинство методов Data Mining были разработаны в рамках теории искусственного интеллекта. Рассмотрим основные из них.

- **Ассоциативные правила**

  $$A \mapsto B, \quad \text{объем, частота, периодичность}$$

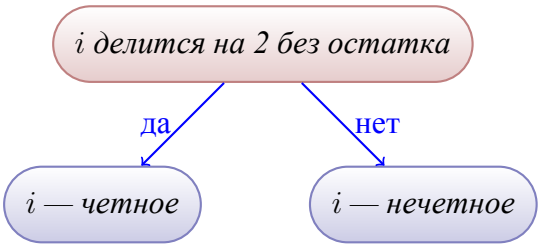
  При поиске ассоциативных правил целью является нахождение частых зависимостей (или ассоциаций) между объектами или событиями. Найденные зависимости представляются в виде правил и могут быть использованы как для лучшего понимания природы анализируемых данных, так и для предсказания появления событий.

- **Деревья решений.** Деревья решений позволяют строить модель зависимости T набора решений (Y) от набора характеристик (Y).

  $$Y = T(X)$$

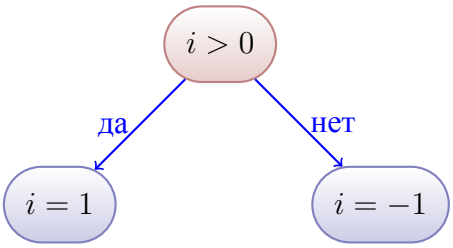
  Деревья решений бывают:

  – *классификационные*. Например:

  Задача классификации сводится к определению класса объекта по его характеристикам. Необходимо заметить, что в этой задаче множество классов, к которым может быть отнесен объект, заранее известно.

  – *регрессионные*[3]. Например для множества $I = \{-1, 0, 1\}$:

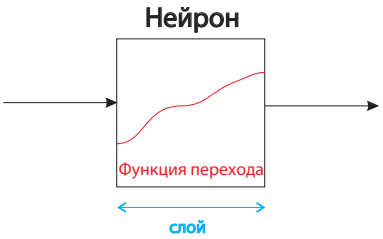
[3]Каждой конечной вершине дерева (листу) приписано решение — некоторое значение. В случае задачи распознавания образов данное значение — это определенный класс, а в случае регрессионного анализа — некоторое число.

Задача регрессии, подобно задаче классификации, позволяет определить по известным характеристикам объекта значение некоторого его параметра. В отличие от задачи классификации значением параметра является не конечное множество классов, а множество действительных чисел.

- **Нейронные сети**

  Различают:

  – нейронные сети с учителем
  – нейронные сети без учителя



- **Генетические алгоритмы**
  Все генетические алгоритмы описывают итерационный процесс эволюции системы с заданными операциями:

  – Отбор, *например:*

    Вход: $\{a_1, ..., a_i, ..., a_n\}$
    Выход: $10 < a_i < 12$

    Выход определяется заранее заданным правилом.

  – Скрещивание, *например:*

    Вход: $\{a_1, ..., a_i, ..., a_n\}$
    Выход: $a_i + a_j$

  – Мутация, *например:*

    Вход: $\{a_1, ..., a_i, ..., a_n\}$
    Выход: $\sin(a_i)$

  Генетические алгоритмы относятся к числу универсальных методов оптимизации, позволяющих решать задачи различных типов (комбинаторные, общие задачи с ограничениями и без ограничений) и различной степени сложности. При этом генетические алгоритмы характеризуются возможностью как однокритериального, так и многокритериального поиска в большом пространстве, ландшафт которого является негладким.

- **Сопоставление с примером**
  - Возникла некоторая ситуация — получим опыт

    $$\text{опыт} = V_{\text{информации после фильтрации}}$$

  - Выясним были ли опыт положительным или отрицательным
  - Запомним
  - Следующую ситуация. Ее мы сопоставляем с полученным опытом, учитывая его знак.
- **Кластерный анализ** Выделение групп объектов.

  Вход: $\{a_1, ..., a_i, ..., a_n\}$
  Выход: $\{a_{i_1}, ..., a_{i_m}\}, \{a_{j_1}, ..., a_{j_l}\}, \{a_{t_1}, ..., a_{t_k}\}$

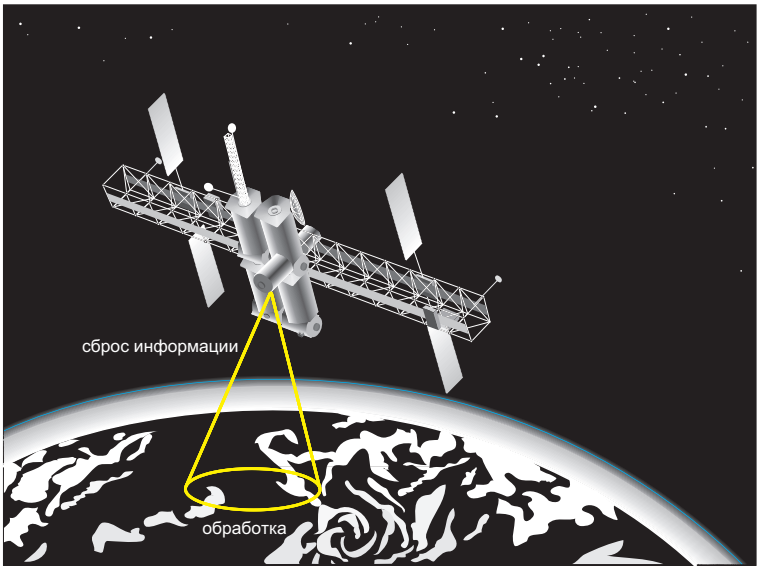
  Кластерный анализ позволяет ускорить поиск. Это своего рода индексация.

  > «Что такое кластеризация? Это деление по группам некоторого количества объектов. Например, у нас в огромной куче смешались чебурашки, велосипеды и роботы-убийцы-детей. Если производить кластеризацию по признаку, то чебурашек мы отнесем в одну группу, велосипеды — в другую, а роботов-убийц-детей в третью. Признак, по которому мы относим один предмет в одну группу, а другой в другую называется метрикой. Грубо говоря, метрика — это просто способ отделить один предмет от другого, используя какие-то точные расчеты. Самый простой вариант — это когда метрикой является расстояние. Кластеризация (деление на кластеры) может быть и немного другой. Если в первом примере с 3 кучами вещей нам сказано разделить их на четыре кластера, то наверняка велосипеды попадут в кучу с чебурашками, а для каких-то детей не найдутся их роботы-убийцы.»[4]

  Задача кластеризации заключается в поиске независимых групп (кластеров) и их характеристик во всем множестве анализируемых данных. Решение этой задачи помогает лучше понять данные. Кроме того, группировка однородных объектов позволяет сократить их число, а следовательно, и облегчить анализ.

Пример data mining: Над Землей проходит спутник. В определенный момент он сбрасывает всю информацию на Землю. На Земле проходит ее обработка. С помощью кластерного анализа выделяется военная, новостная информация, погода. Дальше каждый вид информации отрабатывается нейронной сетью.

[4] Со слов Ольги Владимировны Захаровой, преп. каф. Экономической Кибернетики факультета Экономической Информатики Харьковского Национального Экономического Университета.



Ниже мы подробнее рассмотрим генетические алгоритмы.

Генетические алгоритмы

3. Немного истории

Генетические алгоритмы возникли в результате наблюдения и попыток копирования естественных процессов, происходящих в мире живых организмов, в частности, эволюции и связанной с ней селекции (естественного отбора) популяций живых существ. История эволюционных вычислений началась с разработки ряда различных независимых моделей.

Основную идею генетических алгоритмов высказал Дж. Холланд в конце шестидесятых — начале семидесятых годов XX века. Она получила всеобщее признание после выхода в свет книги, ставшей классикой в этой области, «Адаптация в естественных и искусственных системах»[5].

> Он заинтересовался свойствами процессов естественной эволюции (в том числе фактом, что эволюционируют хромосомы, а не сами живые существа). Холланд был уверен в возможности составить и реализовать в виде компьютерной программы алгоритм, который будет решать сложные задачи так, как это делает природа — путем эволюции. Поэтому он начал трудиться над алгоритмами, оперировавшими последовательностями двоичных цифр (единиц и нулей), получившими название **хромосом**. Эти алгоритмы имитировали эволюционные процессы в поколениях таких хромосом. В них были реализованы механизмы селекции и репродукции, аналогичные применяемым при естественной эволюции. Так же, как и в природе, генетические алгоритмы осуществляли поиск «хороших» хромосом без использования какой-либо информации о характере решаемой задачи. Требовалась только некая оценка каждой хромосомы, отражающая ее приспособленность. Механизм селекции заключается в выборе хромосом с наивысшей оценкой (т.е. наиболее приспособленных), которые репродуцируют чаще, чем особи с более низкой оценкой (хуже приспособленные). Репродукция означает создание новых хромосом в результате рекомбинации генов родительских хромосом.

В семидесятых годах в рамках теории случайного поиска Растригиным Л.А. был предложен ряд алгоритмов, использующих идей бионического поведения особей. Развитие этих идей нашло отражение в цикле работ Букатовой И.Л. по эволюционному моделированию. Развивая идеи Цетлина М.Л. о целесообразном и оптимальном поведении стохастических автоматов, Неймарк Ю.И. предложил осуществлять поиск глобального экстремума на основе коллектива независимых автоматов, моделирующих процессы развития и элиминации особей. Большой вклад в развитие эволюционного программирования внесли Фогел и Уолш. Несмотря на разницу в подходах, каждая из этих «школ» взяла за основу ряд

[5] «Adaptation in Natural and Artifical Systems», 1975.

принципов, существующих в природе, и упростила их до такой степени, чтобы их можно было реализовать на компьютере.

Для этого используются две операции:

- скрещивание, позволяющее создать две совершенно новые хромосомы потомков путем комбинирования генетического материала пары родителей;
- мутация, которая может вызывать изменения в отдельных хромосомах.

В генетических алгоритмах применяется ряд терминов, заимствованных из генетики: *гены*, *хромосомы*, *популяция*, *особь*, *аллель*, *генотип*, *фенотип*. Генетические алгоритмы применяются при разработке программного обеспечения, в системах искусственного интеллекта, оптимизации, искусственных нейронных сетях и в других отраслях знаний.

4. Представление генетической информации

Из школьного курса биологии известно, что хромосомный материал представляет собой последовательность комбинаций четырех нуклеотидов:

- А — аденин;
- Ц — цитозин;
- Т — тимин;
- Г — гуанин.

В генетических алгоритмах вектора переменных также записывают в виде цепочек символов. Обычно используются достаточно бедные алфавиты. Наиболее часто используются бинарный алфавит — проще реализация алгоритмов.

Будем считать, что каждая переменная x представлена фрагментом хромосомы, состоящим из фиксированного количества генов. В любой позиции фрагмента может стоять как ноль, так и единица. Рядом стоящие фрагменты не отделяют друг от друга какими-либо маркерами.

> Важно заметить, (и заметить именно тут), что фенотип однозначно определяется генотипом, а не наоборот. Таким образом наше «кодирование» является первичным по отношению к фенотипу.

Для приведения генетической информации из бинарной формы к десятичному виду подходит любой двоично-десятичный код, но обычно исходят из того, что она представлена в коде Грея. От кода Грея переходим к двоично-десятичному коду, а от него — к натуральным целым числам. Код Грея обладает интересным свойством: соседние числа в двоичном представлении отличаются ровно на символ. Это очень важно для поиска оптимальной точки. В таком представлении поиск происходит быстрее и реже «сваливается» в локальные экстремумы.

Двоичное кодирование			Кодирование по коду Грея		
десятичное	двоичное	шестнадца-теричное	десятичное	двоичное	шестнадца-теричное
0	000	0h	0	0000	0h
1	0001	1h	1	0001	1h
2	0010	2h	3	0011	3h
3	0011	3h	2	0010	2h
4	0100	4h	6	0110	6h
5	0101	5h	7	0111	7h
6	0110	6h	5	0101	5h
7	0111	7h	4	0100	4h
8	1000	8h	12	1100	Ch
9	1001	9h	13	1101	Dh
10	1010	Ah	15	1111	Fh
11	1011	Bh	14	1110	Eh
12	1100	Ch	10	1010	Ah
13	1101	Dh	11	1011	Bh
14	1110	Eh	9	1001	9h
15	1111	Fh	8	1000	8h

Если привлечь геометрические интерпретации Код Грея гарантирует, что две соседние, принадлежащие одному ребру, вершины гиперкуба, на котором осуществляется поиск, всегда декодируются в две ближайшие точки пространства вещественных чисел отстоящие друг от друга на одну единицу точности. Двоично-десятичный код подобным свойством не обладает.

5. Генетические операторы

Механизмы передачи наследственности живой природы моделируются генетическими операторами

Генетические операторы могли быть заимствованы не только из микробиологических исследований, но и из анализа лингвистических явлений.

Генетические операторы:

- кроссовер (в отечественной литературе его называют оператором скрещивания);
- мутация;
- инверсия.

Все операторы воздействуют на генотипы родительских особей. На выходе получается генотип потомка. Смысл операторов:

- передача потомству жизненно важных признаков;
- поддержание уровня изменчивости.

5.1. Кроссовер

Кроссовер, описывает механизм гаметогенеза в диплоидных популяциях организмов. Холландом стал использовать его для моделирования эволюции гаплоидных популяций. Это изобретение привело к тому, что хромосома потомка включает два фрагмента, один из которых принадлежал ранее, «отцовской» хромосоме, другой — «материнской».

Можно принцип кросовера описать как:

- из популяции выбираются две особи, которые будут родителями;
- определяется (обычно случайным образом) точка разрыва;
- потомок определяется как конкатенация части первого и второго родителя.

> Благодаря кроссоверным обменам особи популяции обмениваются между собой генетической информацией, то есть поиск приобретает действительно коллективный характер.

5.2. Оператор мутации

Оператор мутации с вероятностью p_m изменяет значение гена в хромосоме на противоположное. Оператор мутации похож по своей сути на точечные мутации в живой природе. Очевидно, что в зависимости от того, в каком разряде фрагмента, кодирующего переменную, произойдет мутация, зависит величина расстояния, отделяющего потомка от родителя. Оператор предназначен для того, чтобы поддерживать разнообразие особей с популяции.

5.3. Оператор инверсия

Инверсия приводит к нарушению порядка следования фрагментов хромосом у потомка по сравнению с родительской хромосомой. Инверсию тоже можно воспринимать как мутацию, Но она не имеет случайной составляющей и применяется ко все й хромосоме, а не к отдельной аллели.

5.4. Модифицированные операторы

Для функционирования генетического алгоритма достаточно только кросовера, мутации и инверсии, но на практике применяют еще и некоторые дополнительные операторы или модификации этих двух операторов. Например, кроссовер может быть не с одной точкой разрыва, а многоточечный, когда формируется несколько точек разрыва. Кроме того, для кросовера можно использовать и число родителей большее двух. (Правда при таких можификациях поведение алгоритма не всегда предсказуемо.) Оператор мутации иногда применяют сразу к нескольким битам хромосомы. Так же существуют:

6. Классический алгоритм

6.1. В общих чертах

Классический алгоритм[6] можно в первом приближении записать так:

```
1: Инициализация — выбор исходной популяции хромосом
2: Пока  Хромосомы недостаточно приспособлены  выполняем
3:   Выбор хромосом
4:   Применение генетических операторов
5:   Создание новой популяции
6: Выбор наилучшей хромосомы
```

6.1.1. Выбор родителей для скрещивания

Отличительной чертой репродуктивных планов Холланда является право более приспособленных дать большее количество потомков.

Панмиксия

Обе особи, которые составят родительскую пару, случайным образом выбираются из всей популяцию. Любая особь может стать членом нескольких пар. Подход универсален. Но эффективность алгоритма, реализующего такой подход, снижается с ростом численности популяции. В этом мы убедились на практике.

Селекция

«Родителями» могут стать особи с приспособленностью не ниже среднего. Такой подход обеспечивает более быструю сходимость алгоритма. Алгоритм сходится «слишком быстро». Это не всегда хорошо.

- Когда ставиться задача определения нескольких экстремумов, быстро сходится к одному из решений.
- Для класса задач со сложным ландшафтом быстрая сходимость может превратиться в преждевременную сходимость к субоптимальному решению.

Этот недостаток может быть отчасти компенсирован использованием подходящего механизма отбора, который бы «тормозил» слишком быструю сходимость алгоритма. Отчасти проблему решает и использование диплойдных популяций, но подобный прием сильно усложняет исходный алгоритм и рассматривать мы его не будем.

[6] Репродуктивный план Холланда

Инбридинг

Метода построен на формировании пары на основе близкого «родства». Под «родством» здесь понимается расстояние между членами популяции как в смысле геометрического расстояния особей в пространстве параметров так и хемингово расстояние между генотипами. Потому различают генотипный и фенотипный (или географический) инбридинг.

Первый член пары для скрещивания выбирается случайно, а вторым с большей вероятностью будет максимально близкая к нему особь.

Аутбридинг

Формировании пары на основе дальнего «родства», для максимально далеких особей.

> Использование генетических инбридинга и аутбридинга оказалось более эффективным по сравнению с географическим для всех тестовых функций при различных параметрах алгоритма. Наиболее полезно применение обоих представленных методов для многоэкстремальных задач. Однако два этих способа по-разному влияют на поведение генетического алгоритма. Так инбридинг можно охарактеризовать свойством концентрации поиска в локальных узлах, что фактически приводит к разбиению популяции на отдельные локальные группы вокруг подозрительных на экстремум участков ландшафта, напротив аутбридинг как раз направлен на предупреждение сходимости алгоритма к уже найденным решениям, заставляя алгоритм просматривать новые, неисследованные области.

6.2. Подробно

1: Инициализировать начальную популяцию $\mathbb{B}(0)$
2: Ввести точку отсчета эпох $t = 0$.
3: Инициализировать случайным образом M генотипов особей
4: Сформировать начальную популяцию $\mathbb{B}(0) = (\mathbb{A}(0)...\mathbb{A}_M(0))$
5: Оценить приспособленность каждой особи $\mathbb{B}(0)$
6: Вычислить среднюю приспособленность популяции.
7: **Пока** Хромосомы недостаточно приспособлены **выполняем**
8: $t \leftarrow t + 1$
9: Вычислить среднюю приспособленность $\mathbb{B}(t)$
10: **Пока** $t+1$-ая популяция не заполнена **выполняем**
11: Выбрать родительскую особь из $\mathbb{B}(t)$ с учетом ее приспособленности
12: **Если** Используются генетические операторы **то**
13: **Если** Используются кроссовер **то**
14: Выбрать вторую особь из $\mathbb{B}(t)$
15: Выбрать точку кроссовера
16: Провести кроссовер
17: **иначе**
18: **Если** Используются мутация **то**
19: Применить мутацию
20: **иначе**
21: Применить инверсию
22: **иначе**
23: Копировать особь
24: Выбрать особь, которую заменить потомок
25: Поместить потомка в $\mathbb{B}(t+1)$
26: Выбор наилучшей хромосомы

7. Механизм отбора

Метод создания родительских пар сильно связан с реализуемым механизмом отбора при формировании нового поколения. Наиболее эффективные два механизма отбора элитный и отбор с вытеснением.

7.1. Отбор с вытеснением

Из всех особей с одинаковыми генотипами предпочтение отдается тем, чья приспособленность выше. Таким образом, достигаются две цели:

- не теряются лучшие найденные решения, обладающие различными хромосомными наборами,
- в популяции постоянно поддерживается достаточное генетическое разнообразие.

Вытеснение формирует новую популяцию из далеко расположенных особей, вместо особей, группирующихся около текущего найденного решения. Этот метод применяют для многоэкстремальных задач. Есть возможность определения не только глабального экстремума, но и локальных.

7.2. Элитный отбор

Элитные методы отбора гарантируют, что при отборе обязательно будут выживать лучший или лучшие члены популяции совокупности. При этом часть самых лучших особей без каких-либо изменений переходит в следующее поколение. Быстрая сходимость, обеспечиваемая элитным отбором, может быть компенсирована подходящим методом выбора родительских пар. В данном случае часто используют аутбридингом. Именно такая комбинация «аутбридинг — элитный отбор» является одной из наиболее эффективной.

7.3. Турнирный отбор

Турнирный отбор реализует n турниров, чтобы выбрать n особей. Каждый турнир построен на выборке k элементов из популяции, и выбора лучшей особи среди них. Наиболее распространен турнирный отбор с $k = 2$.

8. Пример реализации

8.1. Диофантово уравнение

Рассмотрим простой пример иллюстрирующий работу генетических алгоритмов. Идея примера взята с algolist.manual.ru. Дано диофантово[7] уравнение:

$$a + 2b + 3c + 4d = 15, (a, b, c, d) \in (\{0\} \cup \mathbb{N})^4$$

Применение генетического алгоритма за очень короткое время находит искомое решение (a, b, c, d). Можно было бы подставить все возможные значения a, b, c, d

$$1 \leqslant a, b, c, d \leqslant 15$$

Архитектура генетических алгоритмов позволяет найти решение быстрее. Перебора, грубо говоря, никакого нет. Приближение решения идет от случайно выбранных решений к лучшим. Для решения этой задачи мы реализовали простую программу, полный ее листинг будет приведен в конце примера. В ходе изложения будем приводить листинги функций, которые реализуют заданное действие.

Первое поколение

Для начала выберем 5 случайных решений: $1 \leqslant a, b, c, d \leqslant 15$.

Хромосома	$(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$
0	$(12, 14, 14, 4)$
1	$(9, 5, 14, 13)$
2	$(14, 14, 1, 15)$
3	$(2, 10, 11, 11)$
4	$(5, 4, 3, 1)$

```
1  def _make_first_generation(self,):
2      for i in xrange(self.population_size):
3        self.population[i] = Gene();
4        for j in xrange(4):
5          self.population[i].alleles[j] = random.randint(1, self.param.res);
```

[7] Только целые решения.

Коэффициенты выживаемости

Чтобы вычислить коэффициенты выживаемости, подставим каждое решение в выражение $a+2b+3c+4d$. Расстояние от полученного значения до 30 и будет нужным значением.

Хромосома	Коэффициент выживаемости
0	83
1	98
2	90
3	84
4	11

```
1  def _count_fitness(self, gene):
2      total = self.param.a * gene.alleles[0] \
3          + self.param.b *   gene.alleles[1] \
4          + self.param.c *   gene.alleles[2] \
5          + self.param.d *   gene.alleles[3];
6      gene.fitness = abs(total − self.param.res)
7      return gene.fitness
```

В нашем случае (как это ни странно звучит), чем меньше коэффициент выживаемости тем, лучше.

Выбор родителя

В живой природе, у приспособленных родителей получается приспособленное потомство. Приспособленность родителя можно считать в долях или процентах от общего числа.

```
1  def _inverse_sum(self):
2      sum = 0;
3      for i in xrange(self.population_size):
4          sum += 1.0 / self.population[i].fitness;
5      return sum;
6  def _count_parent_probability(self):
7          inverse_sum = self._inverse_sum();
8          procent_factor = 100.0; # may be 1.0
9          for i in xrange(self.population_size):
10             self.population[i].probability = \
11                 ((1.0/self.population[i].fitness) / inverse_sum) \
12                 * procent_factor;
```

Хромосома	Вероятность стать родителем (в %)
0	8.84743509191
1	7.49323584315
2	8.15930125143
3	8.74210848367
4	66.7579193298

Кроссовер

Каждый потомок содержит информацию о генах и отца и матери. Вообще говоря, это можно обеспечить различными способами. Мы решили реализовать метод «кроссовер»[8]. Допустим гены распределены так:

- *Отец:* a_1, b_1, c_1, d_1;
- *Мать:* a_2, b_2, c_2, d_2.

Тогда для потомка возможны варианты:

Отец	Мать	Потомок
$a_1, \updownarrow b_1, c_1, d_1$;	$a_2, \updownarrow b_2, c_2, d_2$	a_1, b_2, c_2, d_2 или a_2, b_1, c_1, d_1
$a_1, b_1, \updownarrow c_1, d_1$;	$a_2, b_2, \updownarrow c_2, d_2$	a_1, b_1, c_2, d_2 или a_2, b_2, c_1, d_1
$a_1, b_1, c_1, \updownarrow d_1$;	$a_2, b_2, c_2, \updownarrow d_2$	a_1, b_1, c_1, d_2 или a_2, b_2, c_2, d_1

Мы не будем приводить конкретной таблицы для рассматриваемого случая. Приведем лучше код нашего «Кросовера». Для разнообразия можно добавить немного мутации.

```
1  def _make_child(self, p_1, p_2):
2      crossover = random.randint(0, 3);
3      everything = 128
4      trigger = random.randint(0, everything);
5      for i in self.population[p_1].alleles:
6          child.alleles[i] = self.population[p_1].alleles[i];
7      initial = 0; final = 3;
8      if (everything / 2 > trigger):
9          initial = crossover;
10     else:
11         final = crossover + 1;
12     for i in xrange(initial, final):
13         child.alleles[i] = self.population[p_2].alleles[i]; # crosover
14         if (random.randint(0, 666) < 5):
15             child.alleles[i] = random.randint(0, self.param.res); # mutation
16     return child;
```

[8] crossover

Второе поколение

После генерации второго поколения получим:

Хромосома	(**a**, **b**, **c**, **d**)
0	$(5, 4, 3, 1)$
1	$(12, 14, 14, 1)$
2	$(5, 4, 3, 1)$
3	$(5, 4, 14, 1)$
4	$(12, 14, 14, 1)$

Хромосома	**Коэффициент выживаемости**
0	11
1	71
2	11
3	44
4	71

Причем, важно отметить, что :

- Округленный средний коэффициент выживаемости родителей $= 73$
- Округленный средний коэффициент выживаемости потомков $= 41$

Для диофантова уравнения $a+2b+3c+4d = 15$, мы не сразу получили правильный ответ. Но решения лежали весьма близко и конечная сумма падала очень близко от числа 15. При увеличении количества итераций и/или количества популяции близкие числа выпадали чаще. Возможно проблемы связаны, с ошибкой в вычислениях. Мы не сильно из-за этого сейчас беспокоимся. Правильное решение самого уравнения не было целью описания примера. Пример иллюстрирует исключительно работу генетического алгоритма.

> Безусловно, для задач подобного рода генетические алгоритмы не могут проявить себя в полной мере, однако они даже тут эффективнее наивного подхода.

Полный листинг

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
# TIME OF CREATION Fri Jun 01 17:02:34 2010
import random
# ========================================
class Param():
  def __init__(self, a = 0, b = 0,
               c = 0, d = 0, res = 0,
               iterations_limit = 50):
    # There is no type check, but all parameters is Long
    self.a = a
    self.b = b
    self.c = c
    self.d = d
    self.res = res
    self.iterations_limit = iterations_limit
# ========================================
class Gene():
  def __init__(self):
    self.alleles = {};
    self.alleles[0] = 0
    self.alleles[1] = 0
    self.alleles[2] = 0
    self.alleles[3] = 0
    self.fitness = 0;
    self.probability = 0.0;
# ----------------------------------------
  def __eq__(self, other):
    for i in xrange(len(self.alleles.keys())):
      if(self.alleles[i] != other.alleles[i]):
        return False;
    return True;
# ========================================
class Solver():
  def __init__(self, param):
    self.param = param
    self.population_size = 500;
    self.population = {}
    pass;
# ----------------------------------------
  def solve(self):
```

```
42    global_fitness = None
43    self._make_first_generation()
44    global_fitness = self._count_fitnesses();
45    if(0 != global_fitness):
46       return global_fitness;
47    iterations = 0;
48    while ((0 != global_fitness)
49          or ( self.param.iterations_limit != iterations)):
50       self._count_parent_probability();
51       self._make_generation();
52       global_fitness = self._count_fitnesses()
53       if(0 != global_fitness):
54          return global_fitness;
55       iterations += 1;
56    return None
57 #--------------------------------------
58  def _make_first_generation(self,):
59    for i in xrange(self.population_size):
60      self.population[i] = Gene();
61      for j in xrange(4):
62        self.population[i].alleles[j] = random.randint(1, self.param.res);
63 #--------------------------------------
64  def _count_fitness(self, gene):
65    print "(%s, %s, %s, %s)" %(gene.alleles[0],
66                          gene.alleles[1],
67                          gene.alleles[2],
68                          gene.alleles[3]);
69    total = self.param.a * gene.alleles[0] \
70       + self.param.b *   gene.alleles[1] \
71       + self.param.c *   gene.alleles[2] \
72       + self.param.d *   gene.alleles[3];
73    gene.fitness = abs(total - self.param.res)
74    print "| %s - %s | =%s" %(total,  self.param.res,gene.fitness)
75    return gene.fitness
76 #--------------------------------------
77  def _count_fitnesses(self):
78    avgfit = 0;
79    fitness = 0;
80    for i in xrange(self.population_size):
81      fitness = self._count_fitness(self.population[i])
82      sum += fitness
83      if(0 == fitness):
84        print 'avgfit ===>' , sum / self.population_size
85        return i;
```

```
86      print 'avgfit ===>' , sum / self.population_size
87      return 0;
88  # ------------------------------------
89    def _inverse_sum(self):
90      sum = 0;
91      for i in xrange(self.population_size):
92          sum += 1.0 / self.population[i].fitness;
93      return sum;
94  # ------------------------------------
95    def _count_parent_probability(self):
96        inverse_sum = self._inverse_sum();
97        procent_factor = 100.0; # may be 1.0
98
99        for i in xrange(self.population_size):
100         self.population[i].probability = \
101               ((1.0/self.population[i].fitness) / inverse_sum) \
102               * procent_factor;
103       print "%s & %s" %(i,  self.population[i].probability)
104 # ------------------------------------
105   def _get_active_index(self):
106     val = random.randint(0, 101)
107     last = 0;
108     for i in xrange(self.population_size):
109       if (last <= val <= self.population[i].probability):
110           return i;
111       else:
112           last = self.population[i].probability;
113     return 4;
114 # ------------------------------------
115   def _make_child(self, p_1, p_2):
116     crossover = random.randint(1, 3);
117     first = random.randint(0, 128);
118     child = Gene();
119     for i in self.population[p_1].alleles:
120         child.alleles[i] = self.population[p_1].alleles[i];
121     initial = 0;
122     final = 3;
123     if (64 > first):
124         initial = crossover;
125     else:
126         final = crossover+1;
127     for i in xrange(initial, final):
128         child.alleles[i] = self.population[p_2].alleles[i];
129         if (random.randint(0, 666) < 5):
```

```
130             child.alleles[i] = random.randint(0, self.param.res);
131       print 'p_1' , self.population[p_1].alleles.values()
132       print 'p_2' , self.population[p_2].alleles.values()
133       print 'child' , child.alleles.values()
134       return child;
135   #----------------------------------------
136     def _make_generation(self, iterations_limit = 25):
137       tmp = {};
138       for i in xrange(self.population_size):
139         parent_1 = 0;
140         parent_2 = 0;
141         iterations = 0;
142         while(parent_1 == parent_2
143               or self.population[parent_1] == self.population[parent_2]):
144           parent_1 = self._get_active_index();
145           parent_2 = self._get_active_index();
146           iterations += 1
147           if (iterations_limit < iterations):
148             break;
149         tmp[i] = self._make_child(parent_1, parent_2);
150       for i in xrange(self.population_size):
151         self.population[i] = tmp[i];
152   #----------------------------------------
153     def __str__(self):
154       res = "";
155       fitness = self.solve()
156       if(None == fitness):
157         return "No solution found :("
158       gene = self.population[fitness]
159       res += "The solution set to %s a + %s b + %s c + %s d = %s is: \n" \
160         %(self.param.a,
161           self.param.b,
162           self.param.c,
163           self.param.d,
164           self.param.res);
165       res += "a = %s "%gene.alleles[0];
166       res += "b = %s "%gene.alleles[1];
167       res += "c = %s "%gene.alleles[2];
168       res += "d = %s "%gene.alleles[3];
169       return res + "\n"
170   if (__name__ == '__main__'):
171     print  Solver(Param(1,2,3,4,15))
```

9. Интеграция с другими методами

В последние годы резко возросло число работ, посвященных развитию теории ГА и вопросам их практического использования.

Результаты данных исследований показывают, в частности, что генетические алгоритмы могут получить более широкое распространение при интеграции с другими методами и технологиями. Появились работы, в которых доказывается эффективность интеграции генетических алгоритмов и методов теории нечеткости, а также нейронных вычислений и систем.

Эффективность такой интеграции нашла практическое подтверждение в разработке соответствующих инструментальных средств(ИС). Например:

- Фирма Attar Software включила компоненту генетических алгоритмов, ориентированную на решение задач оптимизации, в свои ИС, предназначенные для разработки экспертной системы.
- Фирма California Scientific Software связала ИС для нейронных сетей с компонентой генетических алгоритмов, обеспечивающими автоматическую генерацию и настройку нейронной сети.
- Фирма NIBS Inc. включила в свои ИС для нейронных сетей, ориентированные на прогнозирование рынка ценных бумаг, компоненты генетических алгоритмов, которые, по мнению финансовых экспертов, позволяют уточнять прогнозирование.

Несмотря на известные общие подходы к такой интеграции генетических алгоритмов и нечеткой логики, по-прежнему актуальна задача определения наиболее значимых параметров операционного базиса генетических алгоритмов с целью их адаптации в процессе работы генетических алгоритмов за счет использования нечеткого продукционного алгоритма (НПА).

Перечисленные ниже причины коммерческого успеха инструментальных средств в области искусственного интеллекта могут рассматриваться как общие требования к разработке систем анализа данных, используемых генетических алгоритмов:

- интегрированность — разработка ИС, легко интегрирующихся с другими информационными технологиями и средствами;
- открытость и переносимость — разработка ИС в соответствии со стандартами, обеспечивающими возможность исполнения в разнородном программно-аппаратном окружении и переносимость на другие платформы без перепрограммирования;
- использование языков традиционного программирования. Переход к ИС, реализованным на языках традиционного программирования (C, C++ и т. д.), упроща-

ет обеспечение интегрированности, снижает требования приложений к быстродействию ЭВМ и объемам оперативной памяти;

- архитектура «клиент-сервер» — разработка ИС, поддерживающих распределенные вычисления в архитектуре «клиент-сервер», что позволяет снизить стоимость оборудования, используемого в приложениях, децентрализовать приложения и повысить их производительность.

Эти требования обусловлены необходимостью создания интегрированных приложений, приложений, объединяющих в рамках единого комплекса традиционные программные системы с системами искусственного интеллекта и генетические алгоритмов в частности.

Интеграция **генетических алгоритмов и нейронных сетей** позволяет решать проблемы поиска оптимальных значений весов входов нейронов, а интеграция **генетических алгоритмов и нечеткой логики** позволяет оптимизировать систему продукционных правил, которые могут быть использованы для управления операторами ГА (двунаправленная интеграция).

Одним из наиболее востребованных приложений генетических алгоритмов в области Data Mining является поиск наиболее оптимальной модели (поиск алгоритма, соответствующего специфике конкретной области).

Заключение

В современном обществе развитие информационных технологий способствует эффективному информационному взаимодействию людей и организаций, обеспечивая широкоформатную коммуникацию и свободный доступ к глобальным информационным ресурсам. Информация во многих областях динамично изменяется. В связи с этим огромную роль начинают играть технологии, позволяющие ускорить ее обработку. Это требует новых инструментов добычи, анализа и представления данных в удобном формате для повторного использования. При рассмотрении генетических алгоритмов как метода Data Mining первое что бросается в глаза, что они вроде как и не методы искусственного интеллекта вообще. Может показаться, что эти методы не обладают собственной ценностью, кроме как для задач оптимизации. Отчасти это так. Но используя подходы тех же генетических алгоритмов нельзя про них забывать, потому что рано или позже они проявят себя в полной мере. Даже сейчас в совокупности с другими методам генетические алгоритмы значительно повышают мощность вычислители тельной системы. Генетические алгоритмы можно применять к задачам, которые бы без них решались бы полным перебором. Даже если генетический алгоритм будет спроектирован не удачно, он в любом случае даст результаты лучше полного перебора. Возвращаясь к вопросам оптимизации, нужно сказать, что в настоящее время не существует универсального метода, который мог бы вытеснить все остальные.

В общем случае сложность задач оптимизации обусловлена видом целевой функции, которая может иметь как глобальные. так и локальные экстремумы. Большинство задач можно классифицировать:

- комбинаторные задачи;
- общие задачи без ограничений;
- общие задачи с ограничениями;
- линейные задачи;
- полиномиальные задачи;
- экспоненциальные задачи.

Но есть много задач, которые не попадают не в один из классов. Задача оптимизации, относится к таким. Подобные задачи удобно решать стохастическими методами. Они требуют жесткой формализации. Генетические алгоритмы как раз являются стохастическими. Для задач с одним единственным экстремумом генетические алгоритмы являются наиболее универсальным средством решения. Для задач с глобальным и несколькими локальными экстремумами они все же показывают не самые лучшие результаты.

Огромный минус генетических алгоритмов — это слабая адаптируемость к похожим задачам. То есть для каждой конкретной задачи приходится с самого начала проектировать алгоритм, настраивать его.

Словарь терминов

Приведем термины, которые так или иначе связаны с этой работой.

Аллель — (биол.) значение конкретного гена, значение свойства или вариант свойства.

Гаметы — (биол.) клетки, которые несут генетическую информацию о родительской особи. У животных и человека мужские гаметы называются сперматозоидами, а женские — яйцеклетками. Гаметы содержат *гаплоидный* (одиночный) набор хромосом. В эволюционном моделировании категория «пол особи» не применяется. Объектом моделирования являются панмиктические популяции бесполых организмов, то есть популяции, в которых отсутствует запрет на скрещивание любых двух особей. Разнополовость в природных популяциях как раз и играет роль естественного ограничения на скрещивание — ни пара из двух самцов, ни пара из двух самок не могут дать потомства.

Ген — основная единица наследственности, представляющая собой фрагмент ДНК, кодирующий один или несколько фенотипических признаков и занимающий фиксированный *локус* хромосомы. Ген представляет собой скорее феноменологическую категорию и понятийно противостоит фену — единичному, доступному наблюдению признаку организма, позволяющему дифференцировать его от других особей того же биологического вида. В эволюционном моделировании — фрагмент хромосомы, кодирующий значение одного из искомых параметров.

Генетический алгоритм (репродуктивный план Холланда) — раздел эволюционного моделирования, заимствующий методические приемы из теоретических положений популяционной генетики. Представляет собой своего рода модель машинного исследования поискового пространства, построенную на эволюционной метафоре. Характерные особенности:

- использование строк фиксированной длины для представления генетической
- информации, работа с популяцией строк, использование генетических
- операторов для формирования будущих поколений.

Глобальная оптимизация — процесс поиска экстремума или экстремумов функционала, который в эволюционном моделировании соответствует приспособленности особи, интерпретируемой как ее способность решать поставленную задачу.

Дарвинизм — разработанная Ч. Дарвиным теория, объясняющая эволюцию как результат взаимодействия двух факторов — случайных изменений характеристик особей в че-

реде поколений и естественного отбора (выживание наиболее приспособленных).

Диплойдия — способ хранения наследственной информации у высших организмов, когда в ядрах соматических клеток содержится двойной набор хромосом, одна половина которого досталась особи от отца, а другая — от матери.

Локус — (биол.) позиция гена в цепочке (хромосоме). Множество позиций — локи.

Изменчивость — (биол.) разнообразие признаков и свойств у особей любой степени родства. Термин «изменчивость» применяется также для характеристики преобразования форм живых организмов в процессе их эволюции. Различают изменчивость наследственную (генотипическую) и ненаследственную (паратипическую). Изменчивость, обусловленную возникновением мутаций, называют мутационной, а обусловленную перестановками генов в результате скрещивания — рекомбинационной.

Инверсия — рекомбинационный оператор, который воздействует на фрагмент хромосомы, изменяя в нем порядок следования генов на обратный.

Кросовер, кроссинговер (от англ. cross-over, перекресток) — (биол.) одна из стадий процесса гаметогенеза, присущего только организмам, размножающимся половым путем. В фазе кроссовера хромосомы потенциального родителя, участвующие в выработке гамет, выстраиваются друг напротив друга, скрещиваются в некоторой средней точке и расходятся, обменявшись порциями генетического материала. При последующем оплодотворении мужская и женская гаметы (сперматозоид и яйцеклетка) сливаются в зиготу, диплоидную клетку, дающей начало новому организму. В эволюционном моделировании под кроссовером понимают оператор, который формирует хромосому потомка, собирая ее из фрагментов родительских хромосом. Если речь идет о гаплоидной популяции, результирующая хромосома сразу интерпретируется как самостоятельная особь.

Ламаркизм — эволюционная теория, предшествовавшая дарвинизму.

> Ламарк верил, что эволюционное развитие идет путем наследования потомками индивидуальных приспособлений, приобретенных родителями в течение жизни. Хотя ламаркизм не нашел подтверждения, идея наследования результатов прижизненной адаптации успешно применяется в эволюционном моделировании.

Мутация — оператор, вносящий изменения в структуру копии родительской хромосомы, модифицируя значения отдельных генов в рамках разрешенного аллельного алфавита.

Оптимизация — итеративный процесс улучшения решения задачи, сформулированной в постановке поиска экстремума целевой функции.

Особь — представитель определенного биологического вида, который характеризуется неразрывным единством генотипа и фенотипа. В эволюционном моделировании особь также обладает генотипом, однозначно определяющим приспособленность, и имеет смысл одного из возможных решений рассматриваемой задачи.

Парадигма — самодостаточная совокупность основополагающих идей и принципов, позволяющая на собственной мировоззренческой основе формировать цельную по внутреннему единству картину объектного мира. Сегодня можно говорить о существовании в науке двух равноправных парадигм — классической и эволюционной.

Популяция — группа особей одного вида, то есть обладающих одинаковой структурой генотипа и поэтому способных взаимодействовать друг с другом, например, скрещиваться и давать потомство.

Приспособленность — (биол.) интегральная характеристика реализованных организмом способностей противостоять окружающей среде. Часто оценивается как коэффициент размножения, то есть количество доживших до репродуктивного возраста потомков, оставленных данной особью. В эволюционном моделировании — количественная характеристика, показывающая, насколько успешно особь решает поставленную задачу, и позволяющая сопоставить ее в этом отношении с другими особями.

Селекция — процесс, при помощи которого некоторые особи из популяции отбираются для получения от них потомства. Обыкновенно — на базе предпочтения по величине индивидуальной приспособленности.

Фенотип — набор значений соответствующий данному генотипу. В биологии, набор внешних признаков особи.

Хромосома — (биол.) одна из цепочек ДНК, обнаруженных в клетках. Хромосомы присутствуют во всех клетках организма, хотя только небольшая их часть активна в какой-то конкретной клетке. В эволюционном моделировании под хромосомой понимают фрагмент данных, содержащий искомые параметры. Он может быть представлен в виде бинарной строки или целочисленного массива.

Элитизм — принцип формирования следующего поколения в популяции, при котором хромосомы наиболее приспособленных особей текущего поколения копируются в следу-

ющее поколение, не подвергаясь действию генетических операторов. *Элитизм* гарантирует сохранение в популяции сверхиндивидов, переходящих из поколения в поколение, но, как правило, ускоряет вырождение популяции, иногда преждевременное.

Эволюционное моделирование — направление в математическом моделировании, объединяющее компьютерные методы моделирования эволюции, а также близкородственные по источнику заимствования идей (теоретическая биология, если таковая существует) другие направления в эвристическом программировании. Включает в себя как разделы генетические алгоритмы, эволюционные стратегии, эволюционное программирование, искусственные нейронные сети, нечеткую логику.

Эволюционная система — система, динамика развития которой опирается на принципы воспроизводства, изменчивости, соревнования и отбора.

Эпоха — одна итерация в вычислительном процессе.

Список литературы

[1] Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. «Куприянов М.С. и др. Технологии анализа данных. Data Mining, Visual Mining, Text Mining, OLAP»

[2] Вороновкий Г.К., Махотило К. В., Петрашев С. Н., Сергеев С. А. «Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности»

[3] Рутковская М., Плинский Л. «Нейронные сети, генетические алгоритмы и нечеткие системы»

[4] Батищев Д.И., Неймарк Е.А., Старостин Н.В. «Применение генетических алгоритмов к решению задач дискретной оптимизации»

[5] Holland J.H. «Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence»

[6] Марасанов А.М. «Лекции по корпоративным информационным системам»

[7] Захарова О.В., Никитин И.К. «Использование программного обеспечения Weka Floss для решения задач Data Mining» — Тезисы докладов Второй Международной Научно-Практической Конференции.