Эволюционные вычисления, нейросети, генетические алгоритмы — формальные постановки задач

Л. Н. КОРОЛЁВ
Московский государственный университет им. М. В. Ломоносова

УДК 519.95

Ключевые слова: автоматическая классификация, распознавание образов, нейросети, эволюционные алгоритмы, генетические алгоритмы, синтез алгоритмов.

Аннотация

В статье сделана попытка формализовать постановки некоторых задач, решение которых связано с применением эволюционных и нейросетевых алгоритмов, и обсудить возможность обосновать достоверность решений, полученных в результате применения таких алгоритмов.

Abstract

L. N. Korolev, Evolutional computations, neuronet, genetic algorithms — formal statements, Fundamentalnaya i prikladnaya matematika, vol. 15 (2009), no. 3, pp. 119—133.

We try to formalize the definition of some problems which deal with application of evolutional and neuronet algorithms. One of our goals is to discuss the correctness of solutions obtained with application of evolutional methods.

1. Введение

Задачи, которые решаются с применением эволюционных и нейросетевых алгоритмов, сводятся к классификации и кластеризации [3] элементов некоторых множеств, в которых тем или иным способом закодированы свойства реальных объектов и особенности предметной области исследований.

Существенно то, что в качестве исходных данных для решения задач классификации выступают выборки ограниченного объёма кодов элементов, представляющих реально существующие классы объектов. Выборки, принадлежащие тому или иному классу, как правило, формируются экспертами на основе опыта и интуиции и могут содержать неточности и ошибки, связанные с принятой методикой представления данных и способов кодирования.

Фундаментальная и прикладная математика, 2009, том 15, № 3, с. 119—133.

При обработке данных любого происхождения на компьютерах мы имеем дело с множествами, состоящими из элементов, представляющих слова над конечным алфавитом. Эти слова можно интерпретировать как числа, графы, текст и т. п. В дальнейшем, говоря о множествах, мы чаще всего будем подразумевать множества двоичных слов — в конечном итоге компьютер может обрабатывать элементы только таких множеств.

В задачах кластеризации требуется группировать схожие между собой элементы множества, представляющего реальные объекты, в условиях, когда формально не определена метрика, позволяющая оценивать взаимные отношения между реальными объектами, поэтому в определении степени похожести часто приходится опираться на интуицию и здравый смысл. Формализация понятия здравого смысла сводится к выработке ряда необходимых условий, которым должны удовлетворять результаты, и к доказательствам, что полученные решения дают требуемый результат.

Основная проблема при решении задач классификации и кластеризации состоит в создании алгоритмов, удовлетворяющих этим плохо формализуемым условиям. Эволюционные и нейросетевые алгоритмы [2, 4] предоставляют некоторые общие концепции обработки данных, требующие конструктивного наполнения в каждом конкретном случае. Отличительной чертой этих алгоритмов является принцип обучения — итерационного процесса настройки параметров этих алгоритмов.

Конструирование алгоритмов — сложный творческий процесс, требующий определённой интуиции и привлечения эвристик. Желательно в связи с этим попытаться формализовать в этих задачах и алгоритмах их решения всё то, что не зависит от предметной области исследований, интуиции и эвристик.

2. Постановки задач классификации и кластеризации

Начнём с формальной постановки задачи классификации. Пусть некоторое множество V разбито на N непересекающихся подмножеств $K_1, K_2, K_3, \ldots, K_n$, которые мы назовём классами.

Пусть явно заданы непересекающиеся конечные подмножества $E_i \subset K_i$ $(i = 1, \ldots, N)$ множества V, которые мы назовём выборками представителей классов $K_1, K_2, K_3, \ldots, K_n$ соответственно. При этом

$$E_i \subset K_i \subset V \quad (i = 1, \ldots, N), \quad K_i \cup K_j \neq \varnothing \quad (i \neq j). \qquad (*)$$

Требуется найти такие алгоритмы (функции), которые определяли бы для любого элемента v пространства V принадлежность его одному и только одному классу. Идентификатор класса, которому принадлежит элемент множества, можно считать целым числом, тогда задача сводится к построению функции $\chi(v)$, которая отображает множество V на множество целых чисел — номеров классов принадлежности, на которые разбиты элементы исходного множества. При этом

должно соблюдаться условие, что все элементы выборок должны оставаться в своих классах.

Если не налагать никаких других условий на отношения элементов, принадлежащих множествам классов $K_1, K_2, K_3, \ldots, K_n$, кроме условий $(*)$ и условия, что выборки остаются в своих классах, то такая постановка не имеет практического смысла, так как существует множество алгоритмов, решающих такую задачу. В самом деле, в реальных задачах классификации мы всегда имеем дело с векторным пространством. Любое векторное пространство V всегда можно лексикографически упорядочить и, следовательно, многими произвольными способами разбить V на непересекающиеся классы, например деля интервал изменения значений некоторых выбранных компонент вектора на N интервалов. Следовательно, можно конструктивно построить функцию $\chi(v)$, проверяющую номер интервала, в который попал наш вектор, и тем самым определяющую принадлежность любого вектора, не принадлежащего ни к одной из выборок, к тому или иному классу.

Беря за основу любое произвольное деление множества на классы, мы можем построить алгоритм, который решает поставленную задачу. Алгоритм определения принадлежности произвольного вектора v к одному из классов K_i, заданных своей выборкой E_i, прост и состоит в следующем:

- проверим, не совпадает ли наш вектор v с одним из векторов, принадлежащих выборкам E_i;
- если наш вектор совпал с одним из векторов, принадлежащих выборке E_i, мы полагаем, что этот вектор принадлежит классу E_i.
- если вектор не совпал ни с одним из векторов выборок, то подставим исследуемый вектор в нашу функцию $\chi(v)$ и вычислим её значение k, которое будет номером класса принадлежности вектора v.

Формально при таком алгоритме будут соблюдены требования $(*)$ исходной задачи. В то же время очевидно, что такой алгоритм, основанный на произвольном выборе деления множества V на классы, практически не пригоден, хотя формально он решает поставленную задачу.

При решении практических задач классификации и кластеризации явно или неявно предполагается, что существует некоторый критерий «похожести» друг на друга элементов, принадлежащих одному классу. В метрических пространствах таким критерием можно считать малость расстояния. Но во многих практически важных задачах приходиться иметь дело с пространствами, в которых метрика не введена, или даже с пространствами, в которых невозможно ввести приемлемую метрику, отвечающую интуитивному понятию похожести.

Задача классификации приобретает смысл, если найдено подобие метрики, которое делает все векторы, принадлежащие одному классу, близкими по «похожести», а классы в некотором смысле «компактными».

Заметим, что любой конструктивный алгоритм, умеющий распределять элементы множества на непересекающиеся классы, можно считать способом формального введения понятия «похожести», или «подобия», или «близости» эле-

ментов. Так понимаемая «похожесть» не удовлетворяет классическим аксиомам расстояния. Например, неравенство треугольника в этом случае, как правило, не работает.

В конкретных задачах множество V представляет собой векторы признаков некоторых реальных объектов, которые мы хотим классифицировать, опираясь на интуитивные или экспертные представления о близости одних объектов другим. Классификация производится на основе анализа реальных признаков объектов или процессов. Например, животные могут классифицироваться по таким признакам, как размер, вес, среда обитания, цвет шерсти и т. д., которые тем или иным образом формируют цифровой образ конкретного индивидуума. Выборки элементов классов формируются специалистами и экспертами, владеющими предметом исследований. Почти всегда существует интуитивная или экспертная возможность определить степень похожести одного реального объекта на другой.

В задачах автоматической классификации и кластеризации считается, что алгоритмы формирования выборок нам неизвестны. Именно в таких условиях необходимо решать задачи. В качестве начальных данных мы можем использовать только сами выборки, на основе анализа которых должны конструироваться гипотезы структуры классов и алгоритмы их построения.

3. Формальный анализ выборок

Рассмотрим простой пример, демонстрирующий возможность построения алгоритма классификации на основе анализа выборок.

Пусть исходное множество V представляет собой множество двоичных слов (векторов) заданной длины $n = 6$. Пусть нам заданы две выборки слов, принадлежащих двум классам A и B:

$$E_A \subset A \subset V: \quad 010111,\ 111001,\ 101101,\ 010001,$$
$$E_B \subset B \subset V: \quad 111100,\ 100010,\ 010110,\ 101010.$$

Анализ содержимого этих двух выборок показывает, что все элементы выборки A являются нечётными двоичными числами, а элементы выборки B — чётными. Это, в свою очередь, позволяет нам высказать интуитивно обоснованное предположение, что весь класс A состоит только из нечётных чисел, а весь класс B — только из чётных. Алгоритм искомого отображения, если верно наше предположение, очевиден: проверяется младший разряд слова, если он равен 1, то слово относится к классу A, в противном случае — к классу B. Это позволяет построить булеву функцию, разделяющую классы. Она совершенно проста:

$$F(x1, x2, x3, x4, x5, x6) = x6.$$

Степень похожести двух векторов (слов) также определяется по их младшим разрядам: если они одинаковы, то элементы похожи, если различны, то непохожи. Это мало соответствует тому, что мы понимаем под метрикой в классическом

понимании, но, тем не менее, любым двум векторам может быть поставлено в соответствие число 0 или 1, определяющее их близость. Формально нашу догадку о том простом алгоритме, который позволяет классифицировать все элементы данного множества по двум классам, можно интерпретировать как некоторый метод анализа выборок для конструктивного построения алгоритма.

Рассмотрим другой пример выборок, принадлежащих двум классам множества 6-разрядных двоичных слов:

$$E_A \subset A \subset V: \quad 010101,\ 111011,\ 101110,\ 010101,$$
$$E_B \subset B \subset V: \quad 111100,\ 100010,\ 010010,\ 101001.$$

Эти выборки составлены таким образом, что в выборке A в трёх младших разрядах всегда только по две двоичных единицы, а в выборке B в трёх младших разрядах всегда по одной двоичной единице. Это позволяет нам регулярным образом строить логические функции, различающие классы. В этом примере они таковы:

$$\varphi_1(x_1, x_2, x_3, x_4, x_5, x_6) = \bar{x}_4\bar{x}_5x_6 \vee \bar{x}_4x_5\bar{x}_6 \vee x_4\bar{x}_5\bar{x}_6,$$
$$\varphi_2(x_1, x_2, x_3, x_4, x_5, x_6) = \bar{x}_4x_5x_6 \vee x_4\bar{x}_5x_6 \vee x_4x_5\bar{x}_6.$$

В этом примере переменными, по которым определяется принадлежность к классам, оказались три младших разряда. Поиск сочетания значимых переменных, по которым можно построить функции, определяющие принадлежность к классам, можно осуществлять направленным перебором. Расчёты показывают, что прямой перебор для реальных задач неприемлем. Отыскание таких функций — одна из задач дискретной математики, относящихся к минимизации булевых функций.

Анализ содержания выборок, относящихся к различным классам, состоит в оценке статистики, в анализе частоты встречаемости значений компонент (координат) векторов, включённых в выборки. В частности, в нашем простом примере мы обнаружили, что частота встречаемости единицы в последнем разряде выборки E_A равна 1, а частота встречаемости единицы в последнем разряде выборки E_B равна 0. Тем самым были обнаружены два взаимоисключающих признака, однозначно характеризующих эти классы, и алгоритм классификации был однозначно определён.

Этот простой пример подсказывает один из возможных путей поиска определяющих признаков принадлежности объекта к одному из классов на основе формального анализа состава выборок. Эту задачу мы будем рассматривать применительно к исходному множеству, состоящему из двоичных векторов.

Пусть задано множество V двоичных слов одинаковой длины n. Очевидно, что такое множество слов конечно и его мощность равна 2^n. Рассмотрим некоторое строгое подмножество $K \subset V$ — класс K.

Пусть $E_K \subset K$ — некоторая выборка элементов, принадлежащих классу K. Введём $(2 \times n)$-матрицу Q_{E_K} частотных характеристик выборки E_K, которая будет содержать частоты появления букв двоичного алфавита 0 и 1 в каждой из

позиций n-разрядного слова:

$$Q_{E_K} = \begin{pmatrix} a_{01} & a_{11} \\ a_{02} & a_{12} \\ \dots & \dots \\ a_{0n} & a_{1n} \end{pmatrix}.$$

Элемент a_ij обозначает частоту встречаемости i-й буквы нашего алфавита на j-й позиции слова, принадлежащего выборке E_K.

Любое слово из V длины n с помощью такой матрицы можно отобразить в некоторое векторное пространство V_{E_K} размерности n, состоящее из векторов размерности n, координаты которых состоят из чисел, обозначающих частоты появления соответствующих букв в столбцах матрицы Q_{E_K}. Множество V_{E_K} векторов, полученное таким образом, мы назовём множеством, индуцированным выборкой E_K.

Можно взять другую выборку $E_B \subset E$ из того же класса $E \subset V$ и индуцировать с помощью соответствующей матрицы Q_{E_B} множество V_{E_B}. Вообще говоря, множества, индуцируемые разными выборками из одного и того же класса, будут различны.

Однако если класс построен по какой-то логике, т. е. существуют функции или алгоритмы, генерирующие рассматриваемый класс, то правомочна гипотеза, утверждающая что V_{E_K} и V_{E_B} совпадают. Косвенным признаком существования функции, различающей класс, может служить близость индуцируемых выборками характеристических матриц или «компактность» объединения множеств $V_{E_K} \cup V_{E_B}$. Здесь слово «компактность» взято в кавычки, так как определение понятие компактности может зависеть от предметной области, возможна, в частности, интерпретация компактности в смысле пространства А. Н. Тихонова.

Отображение $E_K \to V_{E_K}$ не является взаимно-однозначным. Если, например, все a_{ij} одинаковы, то все слова множества V отобразятся только на один вектор пространства образов V_{E_K}.

Если частота появления какой-либо буквы на какой-либо позиции слова равна 1, то данная буква может служить характеристикой, определяющей класс K. Правомочна гипотеза, утверждающая, что наличие данной буквы в данной позиции любого слова однозначно определяет принадлежность его к классу K. Вероятность того, что такая ситуация в выборке E_K возникла чисто случайно, даже при небольших объёмах выборки, очень мала.

Анализ частоты встречаемости букв алфавита на различных позициях слова согласуется с идеей поиска логических функций, разделяющих на классы.

В каждом столбце можно найти букву алфавита, частота появления которой максимальна, можно также вычислить среднюю частоту встречаемости буквы на заданной позиции. По этим частотам можно построить слова, которые мы назовём типичными характеристическими словами класса K или центром класса K по заданной выборке E_K.

Очевидно следующее утверждение. Если в каждом из столбцов все частоты a_{ij} различны, то отображение, полученное с использованием такой матрицы, взаимно-однозначно.

Переход от букв к векторам позволяет нам вводить понятие расстояния между двумя словами $\rho(s_1, s_2)$. Используя аппарат классической векторной алгебры в евклидовом пространстве, это можно сделать, например, так:

$$\rho(s_1, s_2) = \rho(v_1, v_2) = \sqrt{\sum_1^n (x_{i1} - x_{i2})^2},$$

где x_{i1}, x_{i2} — соответствующие элементы векторов v_1, v_2.

Можно также ввести понятие расстояния слова от подмножества K как минимальное расстояние до центра подмножества.

Анализ выборок в задачах классификации, связанный с поиском логической функции, разделяющей исходное множество на классы, требует выполнения огромного перебора с применением методов минимизации булевых функций. Задачи кластеризации, связанные с поиском в исходном множестве групп «похожих», близких друг к другу элементов, также требует огромного перебора вариантов группирования, соответствующих интуитивному понятию близости. Большинство реальных задач, возникающих на практике при отыскания экстремумов плохо формализованных функций или функционалов, также требуют огромного числа переборов для точного решения. Способ решения перечисленных выше плохо формализованных, плохо определённых задач — это направленный перебор. В настоящее время для этих целей широко используются так называемые эволюционные алгоритмы.

4. Эволюционные алгоритмы

Эволюционные алгоритмы [8,12,13] можно отнести к известному методу проб и ошибок, связанному с решением переборных задач, возникающих при поисках правильных решений или экстремальных значений в случае отсутствия знания или теории о поведении изучаемого объекта, процесса или функции. В непрерывной математике широко используется метод градиентного спуска для поиска экстремумов функций, в дискретной математике используется метод ветвей и границ, позволяющий отсечь большое число вариантов, которые заведомо не приведут к нахождению правильного решения. В том и в другом случае отыскания экстремума направленным перебором необходимы сведения о характеристиках функции или функционала, экстремум которых надлежит отыскать. При решении практических задач эти сведения, как правило, заранее не известны. Замечательным отличаем эволюционных алгоритмов от многих других методов решений является возможность уточнения характеристик исследуемых объектов в ходе самого решения.

В математике давно известны итерационные методы, позволяющие находить локальные экстремумы функций. Первый шаг таких численных методов поиска экстремума состоит в том, что на некотором отрезке определения функции «разбрасываются» случайным образом несколько точек, в которых вычисляются значения функции. Следующие шаги состоят в том, что выбираются точки с наибольшим (с наименьшим) значением функции и вблизи от них отыскивается новые «перспективные» точки для дальнейших проб с использованием, например, идей градиентного спуска. Выбор точек, на основе которых определяются перспективные интервалы для дальнейших проб на поиск экстремума, можно назвать *селекцией*, получение новых точек — процессом генерации новых поколений точек. Подобные численные методы по своей сути являются итерационными, и каждый шаг итерации состоит в селекции объектов и получении новой их генерации для дальнейшего анализа. Алгоритмы такого типа называют, следуя Дарвину, эволюционными.

Возникло множество эволюционных алгоритмов, использующих в разных видах механизмы типа естественного отбора. Для обозначения некоторых проблем, возникающих при использовании эволюционных алгоритмов, рассмотрим простой пример задачи поиска глобального экстремума некоторой функции одного переменного. В представленном ниже примере мы будем использовать классический алгоритм, принадлежащий к классу эволюционных алгоритмов и получивший название *генетического алгоритма*. В генетических алгоритмах [8,12,13] используется терминология, заимствованная из эволюционной теории Дарвина, которая перекочевала в чисто математические исследования, не имеющие ничего общего с биологией.

Пусть о функции не известно ничего, кроме того, что нам доступен алгоритм, позволяющей вычислить её значение в любой произвольно выбранной точке.

- Возьмём наугад n значений $x_i^{(1)}$, принадлежащих области определения. Вычислим (отыщем) значения $y_i^{(1)} = f\big(x_i^{(1)}\big)$ во всех этих точках, а далее будем рассуждать и действовать следующим образом.
- Расположим $x_i^{(1)}$ в порядке возрастания соответствующих значений $y_i^{(1)}$.
- Будем считать, что значения $x_i^{(1)}$, соответствующие бóльшим значениям $f\big(x_i^{(1)}\big)$, находятся ближе к точке искомого экстремума нашей функции, чем другие, и будем считать их «хорошими» (отметим, что это предположение есть гипотеза, основанная только на интуиции).
- Сформируем новую последовательность $x_j^{(2)}$ той же длины n, но уже не совсем случайным образом, а так, чтобы часть новых $x_j^{(2)}$ находились поблизости от «хороших» точек $x_j^{(2)}$, а другая часть, во избежание попадания в локальный экстремум, охватывала бы другие, «далёкие» области определения функции.
- Получив новую последовательность $x_j^{(2)}$, которая в сложившейся терминологии называется «популяцией», мы будем генерировать таким же образом всё новые и новые популяции.

Этот процесс построения все новых и новых популяций $x_j^{(k)}$ будем продолжать в надежде, что на каком-то шаге мы найдём точку, в которой функция $f(x)$ достигнет своего экстремума. Чем может быть подтверждено, что наша надежда оправдалась, что мы действительно на некой итерации получили значение координаты искомого экстремума? Например, тем, что новые итерации не дают уже улучшения результатов, или тем, что с точки зрения эксперта вся область определения функции, экстремум которой мы ищем, достаточно «плотно» обследована. Эти и многие другие критерии завершения процесса итераций эволюционного алгоритма, очевидно, не имеют ничего общего с формальными строгими математическими доказательствами полученных результатов.

Для получения точек новой популяции на основе анализа точек предыдущей популяции используются операции *скрещивания* и *мутации*. Выбор «хороших» точек и отбрасывание плохих называется *селекцией*.

Обратимся к нашему простому примеру. Заметим, что если мы будем каждый раз выбирать для получения новой популяции только хорошие точки — геномы предыдущей популяции, то велика вероятность «зациклиться» около точки локального экстремума и не найти популяцию, содержащую глобальный экстремум. Во избежание такого явления живая природа придумала мутацию — случайное изменение какого-то генома, приводящее к изменчивости отдельной особи либо в лучшую, либо в худшую сторону. В соответствии с этим механизмом изменчивости в генетических алгоритмах используют операцию мутации, которая обеспечивает, как правило, выход из зацикливания около «хороших» геномов. Более сложную операцию скрещивания можно интерпретировать как попытку сформировать новую особь, смешением признаков двух других особей (родителей).

Общую ситуацию, в которой действует генетический алгоритм в нашей простой задаче отыскания экстремума функции, следует представлять следующим образом.

Имеется «чёрный ящик», устройства которого мы не знаем, но который, получая на вход некоторые данные, на выходе выдаёт некоторые значения, которые можно сравнивать между собой, ранжируя их по величине или значимости. На вход могут поступать характеристики некоторых объектов, которые мы будем называть геномом особи. Геном должен быть закодирован некоторой последовательностью символов. С вычислительной точки зрения любой код для компьютера представляет собой структурированную последовательность двоичных разрядов.

Двоичные элементы такой последовательности часто называют генами, а смысловые поля последовательности иногда называют хромосомами. Сразу же отметим, что в формальной постановке задач, решаемых генетическими алгоритмами, эти термины не связаны с их исходным биологическим смыслом. В общем случае можно считать, что на вход «чёрного ящика» поступают данные в двоичном коде, на выходе мы получаем закодированные в двоичном виде результаты,

с помощью которых можно упорядочить геномы популяции, генерируемой на каждом шаге работы алгоритма.

Операция скрещивания формально состоит в том, что из популяции выбираются геномы двух «родителей» и из них формируются геномы нескольких «потомков». По некоторым правилам, напоминающим перекрёстное опыление (для потомков часть хромосом (блоков генов) берётся от одного родителя, часть хромосом — от другого), они объединяются, но так, чтобы их общее число, а иногда и место расположения, при этом оставалось таким же, как у родителей.

Одноместная операция мутации состоит в том, что у *случайно* выбранного представителя популяции *случайно* меняется один двоичный разряд (ген) на противоположный, так получают значение нового представителя. Слово «случайно» означает, что этот процесс производится с некоторой заранее заданной вероятностью.

Операция селекции состоит в выборе геномов, полученных в результате скрещиваний и мутаций, для формирования новой популяции следующего шага итерации генетического алгоритма. При организации селекции также используется вероятностный подход, т. е. задаются вероятности выбора новых геномов и оставления в популяции старых особей.

Перечисленные выше вероятности являются параметрами генетического алгоритма, и от их выбора в значительной степени зависит качество работы самого алгоритма и успех в решении поставленной задачи. Входными параметрами генетического алгоритма являются также размер генома, заданный размер популяции и число шагов итерационного процесса, необходимых для получения приемлемого решения.

В 1975 году Дж. Холланд [9] доказал теорему, известную под названием теоремы схем (schemata theorem). В очень грубом приближении содержательно теорема схем утверждает, что с увеличением числа итерационных шагов генетического алгоритма вероятность приближения к экстремуму функции качества возрастает. Теорема исходит из того, что геном представляет собой двоичную последовательность, у которой каждый ген (двоичный разряд) может изменяться независимо от других в результате действий операций скрещивания и мутации. Теорема утверждает, что от поколения к поколению операции селекции стабилизируют группу разрядов в геноме, которая не меняется от поколения к поколению.

Поясним утверждение теоремы на примере. Пусть геном представляет собой последовательность n двоичных разрядов. Рассмотрим популяцию некоторого поколения, состоящую из M особей, обладающих самыми хорошими значениями функций качества. Каждому разряду генома поставим в соответствие частоту повторяемости его значения. Например, значение 0 генома с номером i повторяется в популяции k раз, значение 1 этого гена повторяется $M - k$ раз. Если $k \approx M - k$, то будем считать такой ген «случайным», или неинформативным, и обозначим его неопределённое значение символом $*$, если же $k \gg M - k$, то будем считать, что вероятность закрепления за геномом некоторого значения

пропорциональна частоте его появления в популяции. Это позволяет нам построить «схему» генома, отражающую степень закрепления определённых значений генов в популяции. Схема будет выглядеть, например, так:

0	0	*	*	1	0	1	1	*	0

.

Значимые переменные схемы можно интерпретировать как определение значимых переменных булевой функции, выделяющих класс точек пространства, в которых находится искомый экстремум.

Геном с заданными значениями всех его разрядов можно интерпретировать как одну вершину n-мерного куба. Множество вершин, удовлетворяющих схеме, в которой закреплены значения лишь за несколькими разрядами, представляет собой несколько вершин, образующих аналог гиперплоскости в обычном n-мерном пространстве.

Прямой перебор для решения задачи отыскания экстремума функции потребует огромного числа вычислений. Практические эксперименты на задаче о рюкзаке показывают, что для достижения заданной цели требуется около 200 итераций при размере популяций порядка 40 геномов в каждой. Ускорение, даваемое генетическим алгоритмом по сравнению с прямым перебором, равно 1012, работа ускоряется в миллиарды раз.

При некоторых искусственных предположениях математически точно доказывается сходимость генетического алгоритма к точному решению поставленной задачи.

Для некоторых реальных задач, как уже говорилось, доказательство того, что алгоритм выдал нам точное (или близкое к точному) решение, получить трудно, и это один из недостатков генетических вычислений.

Эволюционные алгоритмы нашли широкое применение для решения практических задач, связанных с минимизацией функций, заданных на перестановках. Формальная постановка таких задач состоит в следующем.

Пусть нам задано конечное множество из N уникальных элементов. Уникальность элементов означает, что каждый из них имеет своё, присущее только ему, имя и обладает некоторым набором приданных ему характеристик. Каждому элементу этого множества может быть приписан порядковый номер. Если в нашем множестве N элементов, то мы можем присвоить им порядковые номера $N!$ различными способами. Пусть с каждой нумерацией связаны некоторые значения (число, вектор, слово и т. п.) и существует алгоритм вычисления этих значений. Функцию $F(p)$, зависящую от способа нумерации элементов множества, мы будем называть функцией, заданной на перестановках, или просто функцией перестановок.

В качестве имён элементов можно выбрать номера от 1 до N и назвать эту последовательность начальной перестановкой $p_1 = (1, 2, 3, \ldots, N)$. Все перестановки можно перенумеровать от 1 до $M = N!$.

Многие задачи сводятся к поиску экстремумов функций перестановок. Классическим примером функции, заданной на перестановках, является задача коммивояжёра. Если перенумерованы все города, которые должен посетить ком-

мивояжёр, и заданы расстояния между любой парой городов, то перестановку можно считать порядком обхода городов, а сумму пройденных расстояний можно считать значением функции на выбранной перестановке.

При небольшом числе городов эту задачу можно решить прямым перебором. Но если N большое, например 300, то с прямым перебором не справится даже очень мощный современный суперкомпьютер!

Другим примером задачи, сводящейся к поиску экстремума функции на перестановках, является хорошо известная задача о рюкзаке. Дано N предметов, каждый из которых обладает своим весом и стоимостью. Рюкзак может быть нагружен только до определённого веса. Следует из имеющихся N предметов положить в рюкзак наиболее ценные, не превысив допустимый вес. Здесь с каждым предметом связано уже два значения: вес и стоимость (ценность).

Обе эти задачи можно пытаться решать, выбирая одну перестановку за другой наугад, в надежде случайно наткнуться на удачную перестановку, доставляющую искомый экстремум функции или приближающую к экстремуму. Для решения задач описанного типа с успехом применяются генетические алгоритмы.

5. О генетическом программировании

В модели с «чёрным ящиком» возможна и другая постановка задачи. Пусть мы знаем реакцию чёрного ящика на входные данные, которые мы также будем называть геномами. Нас интересует, по какой программе работает этот аппарат, выдавая значения некоторой вычисленной им функции.

Задача генетического программирования [10,11] как раз сводится к конструированию программы, выполняющей аналогичное «чёрному ящику» преобразование данных, с использованием идей генетических алгоритмов. Речь идёт о конструировании реальной вычислительной программы, которую можно задать вычислительной машине для её реального исполнения, т. е. о программе, закодированной на одном из формальных алгоритмических языков, доступных для понимания компьютера.

Формальный механизм генетического программирования состоит в следующем [8, 11]. Любое вычисление арифметического выражения можно интерпретировать как поток вычислений, определяемый соответствующим графом. Например, рассмотрим простое выражение

$$y = \big((a + b) \cdot c + (a - d) \cdot x\big) \cdot r.$$

Последовательность вычислений, выраженных таким графом, может быть записана в виде программы — последовательности символов, включающей в свой состав переменные, символы операций и управляющие конструкции. Эту последовательность символов, отображающих граф вычислений, мы будем считать геномом, точнее набором хромосом, несущих на себе соответствующую семантическую нагрузку.

Теперь построим следующий процесс. Соблюдая некоторые очевидные правила, случайно выберем последовательность этих трёх типов символов, образующую программу, которая что-то может вычислять. Подставим в эту случайно выбранную программу входные значения, воспринимаемые нашим «чёрным ящиком». Наверное, результат вычислений по такой случайно сформированной программе будет значительно отличаться от ожидаемого значения. Ошибку в вычислениях мы будем считать значением функции качества генетического алгоритма, конструирующего нужную машинную программу. Тем самым мы определили вид генома и функцию качества, и для реализации генетического алгоритма нам нужно разумно определить операции скрещивания, мутации, селекции, размер популяции и критерии окончания итерационного процесса по выбору правильной программы, которая минимизирует ошибку в вычислении отклика «чёрного ящика» на входные данные.

Перейдём к более формальной постановке задачи генетического программирования для простого примера, связанного с конструированием программы вычисления функции одного переменного.

Пусть нам вместо «чёрного ящика» задана конечная таблица пар чисел (x_i, y_i), где x_i обозначает аргумент, а y_i — значение функции от этого аргумента. Будем искать программу, представленную древовидным графом, которая будет вычислять по значению x_i значение z_i, минимально отличающееся от y_i, т. е. будем искать программу, минимизирующую функцию

$$F(p) = \sum_{1}^{n} |y_i - z_i|.$$

Аргументом функции F является программа-геном p. Положим размер популяции программ равным, например, 10 и случайным образом построим 10 графов-программ с небольшим числом узлов и связей между ними. Для каждого графа i вычислим значение функции $F(p_i)$ и упорядочим p_i по значениям функции $F(p_i)$.

Далее мы будем следовать классическому генетическому алгоритму для получения популяции программ на следующем шаге. Однако операцию скрещивания мы будем проводить своеобразным способом, а именно выберем два графа из популяции, в каждом из них выделим какие-то ветви дерева и поменяем их местами. Из двух родителей мы получаем двух потомков, которые могут оказаться в следующей популяции, если их выберет алгоритм селекции.

Операция мутации графа состоит в том, что к дереву либо добавляется один узел, либо убирается какая-либо ветвь или узел. При выполнении операций мутации и скрещивания соблюдается правила сохранения правильного синтаксиса новых графов, которые составят новую популяцию.

В отличие от обычного генетического алгоритма геномы алгоритма могут быть разной длины с разным числом узлов и связей между ними, что создаёт дополнительные сложности в организации алгоритма конструирования хороших программ.

Представление программ в виде деревьев и других графовых структур распространено в практике генетического программирования. Развитая теория генетического программирования позволяет различным образом интерпретировать понятие программы. Вместо машинных программ, используя генетическое программирование, можно конструировать электрические сети с заданными характеристиками, функции алгебры логики, в принципе любые автоматические устройства. На основе использования идей генетического программирования сконструированы и даже запатентованы [10, 11] некоторые автоматические радиотехнические устройства!

Основными элементами любой генетической программы являются терминалы и функции. При использовании генетического программирования для решения оптимизационной задачи в какой-либо предметной области следует выбрать алфавит терминальных символов и множество используемых функций. В области конструирования машинных программ в качестве набора функций, как правило, выбирают стандартный набор арифметических операций, элементарных функций, операторов ветвлений и циклов. В качестве терминальных символов — идентификаторы начальных данных и результатов промежуточных вычислений. Последние иногда играют роль функций обращения к памяти по записи и чтению. Набор функций может включать и более сложные элементы, такие как используемые в данной предметной области готовые библиотечные программы.

Процессы, происходящие в природе на разных уровнях, начиная с биомолекулярного и кончая макропроцессами развития живой материи, дают богатую пищу для создания всё новых и новых алгоритмов, подражающих природным процессам. Достаточно привести название некоторых алгоритмов: «иммунные сети», «муравьиный алгоритм» (он воспроизводит логику поведения муравьев по отысканию кратчайшего пути к пище), алгоритм «птичьей стаи» (он решает некоторые навигационные задачи) и т. п. Но в этой сфере исследований и приложений слишком много эвристик, адекватных здравому смыслу, но не получивших точных математических доказательств полученных решений. Совпадение правдоподобия и правильности полученных решений обосновывается экспериментом на некоторых базах данных, принятых в качестве мировых стандартов, на которых проверяется эффективность и точность многочисленных эвристик по распознаванию образов, классификации, кластеризации и прогнозированию.

6. Заключение

Правдоподобие результатов, полученных нейросетевыми и эволюционными алгоритмами, часто доказывается вычислительным экспериментом, сравнением результатов на тестовых выборках. Однако во многих статьях, описывающих результаты экспериментов этого типа, очень часто отсутствуют статистические обоснования чистоты постановки таких экспериментов. Очень важно привлечь классические методы теории вероятности и математической статистики к анализу выборок с точки зрения доказательства их «неслучайности», наличия неко-

торого закона их построения. В частности, это могло бы дать обоснованный анализ выборок малого объёма, существенного для минимизации перебора, и числа шагов в итерационных процессах.

Нейросетевые вычисления, эволюционные вычисления и генетическое программирование можно отнести к многогранному кругу задач распознавания образов [6]. Успешным прорывом в направлении математизации задач распознавания образов, которые принадлежат к классу плохо формализованных, следует отнести работы группы Ю. И. Журавлёва [1,5,7]. Эти работы позволяют применить алгебраический подход к анализу алгоритмов решения задач такого типа и тем самым навести некоторый порядок в сфере задач классификации и прогнозирования. Рассмотренные в данной статье алгоритмы обладают своей частной спецификой, надлежащий математический порядок ещё не установлен. Будем надеяться, что это когда-нибудь произойдёт.

Литература

[1] Журавлёв Ю. И. Об алгоритмах распознавания с представительными наборами (о логических алгоритмах) // ЖВМ и МФ. — 2002. — Т. 42, № 9. — С. 1425—1435.

[2] Минский М., Пейперт С. Персептроны. — М.: Наука, 1971.

[3] Осовский С. Нейронные сети для обработки информации. — М.: Финансы и статистика, 2002.

[4] Розенблатт Ф. Принципы нейродинамики. — М.: Наука, 1965.

[5] Рудаков К. В., Чехович Ю. В. О проблеме синтеза обучающих алгоритмов выделения трендов (алгебраический подход) // Прикладная математика и информатика. Вып. 8. — М.: Изд. отдел ф-та ВМиК МГУ, 2001. — С. 97—113.

[6] Ту Дж., Гонсалес Р. Принципы распознавания образов. — М.: Мир, 1978.

[7] Чехович Ю. В. Применение алгебраического подхода к задачам выделения трендов // Математические методы распознавания образов (ММРО-10). Докл. 10-й Всероссийской конф. ВЦ РАН. — М.: АЛЕВ-В, 2001. — С. 315—316.

[8] Forrest S. Genetic algorithm: Principles of natural selection applied to computation // Sciense. — 1993. — Vol. 261. — P. 872—878.

[9] Holland J. Adaptation in Natural and Artificial Systems. — Cambridge: MIT Press, 1992.

[10] Koza J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. — Cambridge: MIT Press, 1992.

[11] Koza J. Genetic Programming. II: Automatic Discovery of Reusable Programs // Cambridge: MIT Press, 1994.

[12] Filho J. R., Alippi C., Treleaven P. Genetic algorithms programming environments // Computer. — 1994. — Vol. 27, no. 6. — P. 28—43.

[13] Srinivas M., Patnaik L. Genetic algorithms, a survey // Computer. — 1994. — Vol. 27, no. 6. — P. 17—26.