Генетические алгоритмы понижения значности признаков

A Preprint

Сорокин Олег Владиславович
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Пронозирования
ovs02@mail.ru

Abstract

Рассматривается проблема логических подходов к задаче классификации, связанная с возможным наличием в датасете признаков высокой значности, в частности, признаков, принимающих вещественные значения. Вводится понятие перекодировки таких признаков и условие корректного перекодирования относительно данной обучающей выборки. Приводится постановка задачи поиска особого покрытия булевой матрицы (т.н. кодирующего покрытия), соответствующего конкретному перекодированию. Рассматривается возможность применения стандартных жадных и генетических алгоритмов к решению поставленной задачи. Предлагаются обобщения и модификации таких алгоритмов. Приводятся результаты экспериментов с предложенными обобщениями, демонстрирующие их работоспособность.

Keywords Классификация · Понижение значности · Генетический алгоритм · Кодирующее покрытие

1 Введение

Одной из центральных задач машинного обучения является задача классификации по прецедентам. Под прецедентной (обучающей) информацией понимается совокупность примеров изучаемых объектов, представленных в виде числовых векторов, обычно полученных на основе некоторых наблюдений. Для выборок малых размеров с признаками небольшой значности достаточно эффективен логический подход к упомянутой задаче (1; 2). В рамках этого подхода предполагается, что каждый признак принимает ограниченное число допустимых целочисленных значений. Часто анализ прецедентов сводится к построению т.н. элементарных классификаторов, разделяющих объекты из разных классов (3; 4).

Однако часто реальные объекты описываются признаками высокой значности (например, категориальными признаками с большим числом допустимых значений или вещественнозначными признаками). В таком случае построение элементарного классификатора оказывается неэффективным или даже невозможным, и необходимо перекодирование исходных данных (5).

Ю.И. Журавлёвым предложена методика корректного перекодирования исходных данных (6). Показано, что задача построения корректной перекодировки может быть сведена к построению специального вида покрытия булевой матрицы, которая строится по обучающей выборке. Кроме того, из специфики построения этой булевой матрицы следует, что для эффективной перекодировки это покрытие должно содержать как можно меньшее число столбцов.

Известно, что задача о минимальном покрытии множества может рассматриваться как обобщение задачи о минимальном вершинном покрытии, которая лежит в классе NP (а потому также является NP-сложной) (7). Верхние оценки временной сложности известных точных методов (метод ветвей и границ, целочисленный метод отсечения и др.) сопоставимы с оценками для переборных алгоритмов (8; 9). В связи с этим использование точных алгоритмов решения для матриц большого размера может быть невозможным на практике. Вместо этого применяют алгоритмы поиска приближённых решений.

В 1979 году В. Хватал предложил модификацию жадного алгоритма для задачи о минимальном покрытии множеств с произвольными весами (10). Главным недостатком такого алгоритма является долгое время работы для задач с разреженными матрицами, поскольку количество итераций перебора быстро возрастает при уменьшении "густоты"единиц на строку матрицы. Существуют теоретические работы, в которых показана неулучшаемость некоторых оценок временной сложности такого алгоритма для ряда задач фиксированного размера (11).

К гораздо более часто используемым алгоритмам поиска приближённых решений относится генетический алгоритм, впервые в общем виде описанный Д. Холландом в 1975 году (12). Одним из преимущество такого подхода является общность используемой в нём схемы, поэтому в алгоритм достаточно просто вносить модификации. Для задачи о покрытии множества наиболее сложным вопросом при выборе конкретного генетического алгоритма является устойчивость к падению в локальные минимумы (13). Для предотвращения преждевременной сходимости обычно используют ряд эвристик, основанных на специфике решаемой оптимизационной задачи (13; 14; 15; 16; 17).

Таким образом, для матриц произвольной конфигурации известны многие алгоритмы приближённого решения задачи о покрытии. Однако в рамках метода Ю.И. Журавлёва рассматривается поиск только покрытий из особого класса (т.н. кодирующих покрытий) (6). Классы кодирующих и минимальных покрытий для одной и той же булевой матрицы в общем случае, очевидно, не совпадают. Соответственно, известные оптимизационные алгоритмы поиска минимального покрытия не всегда будут находить кодирующее покрытие. Это означает, что значность некоторых признаков может остаться неизменной после перекодирования. Для решения этой проблемы требуется вносить модификации в имеющиеся алгоритмы поиска минимальных покрытий.

В данной работе приведены обобщения градиентного (жадного) алгоритма и генетического алгоритма из (14) на случай кодирующих покрытий. В частности, предложен новый генетический алгоритм, частично использующий идеи из (13) и (14), но учитывающий специфику рассматриваемого класса задач. Проведён ряд вычислительных экспериментов с реальными данными, демонстрирующих работоспособность предложенных обобщений.

2 Постановка задачи

В общем случае задачу о покрытии множества можно сформулировать следующим образом (13). Пусть имеется конечное множество $S = \{\sigma_1, \sigma_2, ..., \sigma_m\}$ и система некоторых его подмножеств $S_j \in S$, $j = 1, 2, ..., n$, таких что

$$\bigcup_{j=1}^{n} S_j = S.$$

Каждому из подмножеств S_j поставлен в соответствие вес $c_j > 0$. Требуется найти набор подмножеств S_j, такой, что каждый элемент множества S принадлежит хотя бы одному из множеств этого набора, и сумма весов множеств, входящих в этот набор, минимальна среди всех таких наборов.

Задача также может быть сформулирована в терминах покрытий булевых матриц (13; 14). Введём матрицу $A = (a_{ij})_{m \times n}$, где

$$a_{ij} = \begin{cases} 1, \sigma_i \in S_j \\ 0, \text{иначе} \end{cases}$$

Здесь предполагается, что каждый элемент σ_i входит хотя бы в одно из подмножеств S_j. Также введём булевы переменные z_j, $j = 1, 2, ..., n$:

$$z_j = \begin{cases} 1, \text{если } S_j \text{ входит в покрытие} \\ 0, \text{иначе} \end{cases}$$

В этих обозначениях задача о покрытии множества может быть записана в виде следующей задачи целочисленного программирования:

$$\begin{cases} \sum\limits_{j=1}^{n} c_j z_j \to \min\limits_{z_1, z_2, ..., z_n} \\ \sum\limits_{j=1}^{n} a_{ij} z_j \geq 1, i = 1, 2, ..., m \\ z_j \in \{0, 1\}, j = 1, 2, ..., n \end{cases}$$

Множеству введённых булевых переменных однозначно соответствуют столбцы матрицы A. Будем считать, что столбец входит в покрытие тогда и только тогда, когда соответствующая булева переменная равна 1. В этом смысле в дальнейшем не будем различать переменную и её столбец, а также реализацию набора переменных и соответствующий набор столбцов.

Любой набор столбцов, удовлетворяющий ограничениям задачи, называется покрытием матрицы. Число $\sum_{j=1}^{n} c_j z_j$ называется весом соответствующего покрытия. В случае когда $c_j = 1$ при $j = 1, 2, ..., n$ это число будем также называть длиной покрытия.

В силу предположения о вхождении каждого σ_i хотя бы в одно из подмножеств S_j решение поставленной задачи дискретной оптимизации существует. Пусть некоторый набор булевых переменных $(\hat{z}_1, \hat{z}_2, ..., \hat{z}_n)$ является решением задачи. Соответствующий набор столбцов назовём минимальным покрытием булевой матрицы A.

Пусть теперь каждому столбцу z_j $(j = 1, 2, ..., n)$ булевой матрицы A поставлена в соответствие уникальная пара (k, t), где $k = 1, 2, ..., d,\ t \in \mathbb{R}$. Такое соответствие будем обозначать записью $z_j \rightsquigarrow (k, t)$. Без ограничения общности будем считать, что $\forall k = 1, 2, ..., d\ \exists\ z_j\ (j = 1, 2, ..., n),\ t \in \mathbb{R} :\ z_j \rightsquigarrow (k, t)$. Кодирующим покрытием булевой матрицы M называется набор столбцов $z_{j_1}, z_{j_2}, ..., z_{j_r}$, такой что

1. $z_{j_1}, z_{j_2}, ..., z_{j_r}$ является покрытием матрицы A;
2. $\forall k = 1, 2, ..., d\ \exists\ z_{j_l}\ (l = 1, 2, ..., r),\ t \in \mathbb{R} :\ z_{j_l} \rightsquigarrow (k, t)$

Второе условие из этого определения также в дальнейшем будем называть условием корректного перекодирования.

Введём обозначение для групп столбцов $G(k) = \{z_j \mid \exists t \in \mathbb{R} :\ z_j \rightsquigarrow (k, t)\}$. Число $\max_k |G(k)|$ называется значностью перекодирования.

В введённых обозначениях задача для поиска минимального кодирующего покрытия имеет вид

$$\begin{cases} \sum_{j=1}^{n} z_j \to \min\limits_{z_1, z_2, ..., z_n} \\ \sum_{j=1}^{n} a_{ij} z_j \geq 1, i = 1, 2, ..., m \\ \left(\sum\limits_{z \in G(k)} z \right) + [G(k) = \emptyset] \geq 1,\ k = 1, 2, ..., d \\ z_j \in \{0, 1\}, j = 1, 2, ..., n \end{cases}$$

3 Теоретическое описание обобщённых алгоритмов

3.1 Обобщение градиентного алгоритма для перекодирования признаков

Рассмотрим жадный алгоритм для поиска минимального покрытия. Будем считать, что решается задача поиска покрытия минимальной длины, то есть все веса столбцов c_j положим равными 1. Такой алгоритм не обязательно удовлетворяет условию корректного перекодирования, то есть в общем случае полученное покрытие может быть не кодирующим.

Пусть градиентный алгоритм получает некоторое покрытие H булевой матрицы M, не содержащее столбцы из групп $G(k_1), G(k_2), ..., G(k_l)$. Произвольным образом выберем $j_{k_1} \in G(k_1), j_{k_2} \in G(k_2), ..., j_{k_l} \in G(k_l)$. Тогда множество столбцов

$$K = H \cup \{j_{k_1}, j_{k_2}, ..., j_{k_l}\}$$

является покрытием (т.к. H - покрытие) и, очевидно, удовлетворяет условию корректного перекодирования.

При добавлении столбцов таким образом длина полученного покрытия часто значительно увеличивается, поэтому необходимо провести дополнительную оптимизацию решения.

Назовём кодирующее покрытие неприводимым кодирующим, если при удалении из него любого столбца полученное множество столбцов не является кодирующим покрытием. Введём дополнительный оптимизационный шаг, состоящий в выделении кодирующего неприводимого покрытия. Обозначим U_j число

строк, которые покрываются только столбцом с номером j. Для каждого столбца $j = 1, 2, ..., n$: если выполнены оба условия

1. $U_j \geq 2$
2. $\nexists i \in \{1, 2, ..., d\} : \ G(i) = \{j\}$

то столбец с номером j исключается из решения с последующим обновлением соответствующих $G(i)$ и U_j.

Отметим, что данный подход в силу неулучшаемости теоретических оценок на число итераций стандартного жадного алгоритма оказывается аналогичным образом малоэффективен для разреженных по числу единиц матриц. Кроме того, по построению допускается наличие большого числа групп, в которых содержится малое число столбцов (1, если группа была покрыта столбцом только при восстановлении решения). При этом, напротив, другие группы содержат большое число столбцов. Такое неэффективное распределение приводит к большой значности перекодирования, что обычно сильно отражается на качестве классификации.

Описанную проблему высокой значности решений можно частично решить, если упорядочить столбцы по мощностям соответствующих им групп, а затем начать их последовательное исключение при выполнении условий. Однако большинство матриц, строимых в процессе перекодирования обычно сильно разрежены по числу единиц (т.к. лишь малая часть из всех определённых порогов разделяет выбранную пару объектов), а проблема большой временной сложности остаётся неразрешимой в классе жадных алгоритмов.

3.2 Обобщение генетического алгоритма для перекодирования признаков

Сначала в общем виде рассмотрим генетический подход к решению произвольной задачи дискретной оптимизации. Пусть для определённости это будет задача минимизации некоторой заданной функции Fitness(x). Популяцией будем называть некоторое множество целочисленных или бинарных векторов одной размерности, а элементы популяции будем называть особями или индивидами. Пусть определены функции:

- $Init(N)$ (возвращает некоторую популяцию размера N);
- $ChooseParents(P)$ (возвращает пару особей из популяции P);
- $Crossover(x1, x2)$ (возвращает нового индивида-потомка);
- $Mutation(x)$ (возвращает изменённую копию индивида x);
- $Selection(P, N)$ (выбирает из произвольной популяции P, $|P| > N$, подмножество размера N);
- $StopCriterion(f1, f2, ..., fN)$ (критерий досрочного останова).

Algorithm 1 Общая схема работы генетического алгоритма

```
function GeneticAlgorithm(N_i, i_max)
  P_0 := Init(N)
  for i = 0..(i_max − 1) do
      for j = 1..N do
          f_j := Fitness(P_j^i)
      end for
      if StopCriterion(f_1, f_2, ..., f_n) then
          break
      end if
      p_1, p_2 := ChooseParents(i)
      child := Crossover(p_1, p_2)
      child := Mutation(child)
      P_i := P_i ∪ {child}
  end for
  return P^i_{argmin(f_1,f_2,...,f_N)}
```

Будем рассматривать только алгоритмы, в которых особи являются бинарными векторами. Примерами эффективных бинарных генетических алгоритмов для поиска минимальных покрытий являются (13) и

(14), причём (14) использует многие идеи из (13), но также задействует удачную эвристику для числа мутаций:

$$k(t) = K\left(1 - \frac{1}{1 - Ct}\right),$$

где t - номер итерации, K и C - гиперпараметры. Таким образом, этот алгоритм при правильном подборе параметров не застревает в локальных минимумах и лучше решает задачу.

Для того, чтобы потомок, получаемый с помощью генетического алгоритма, удовлетворял условию корректности перекодирования, необходимо модифицировать процедуру восстановления потомка из (14). Обновлённая процедура для потомка H и матрицы M будет выглядеть следующим образом:

1. Положим $w_i = |N_H \cup N_i|$, где N_H - множество столбцов потомка (H как набора столбцов), $N_i = \{j \in \{1, 2, ..., n\} \mid M_{ij} = 1\}$

2. Положим $M' = \{i | w_i = 0,\ i = 1, 2, ..., m\}$. Упорядочим индексы по возрастанию.

3. Для каждого $i \in M'$ найдём столбец, являющийся решением задачи

$$|M' \cap M_j| \to \max_{j \in N_i},$$

   где $M_j = \{i \in \{1, 2, ..., m\} \mid M_{ij} = 1\}$ Столбец с номером, равным найденному решению задачи, включается в множество столбцов особи-потомка.

4. Положим $w_i = w_i + 1,\ i \in M_j,\ M' = M' \backslash M_j$. Произведём переход к шагу 3, если множество столбцов, соответствующее потомку, всё ещё не является покрытием исходной матрицы.

5. По аналогии с градиентным алгоритмом, достроим полученное покрытие до кодирующего покрытия.

6. Для каждого столбца j: если выполнены условия, аналогичные условиям, указанным в модификации жадного алгоритма, удалить столбец из решения.

4 Вычислительные эксперименты

Проведём сравнение обобщение бинарного генетического алгоритма Р.М. Сотнезова и жадного алгоритма на ряде задач. Для сравнения будем использовать сгенерированные матрицы различных размеров и конфигураций, а также задачи из библиотеки Beasley's OR-Library (18). Из указанной библиотеки рассмотрим задачи scpe1, scpe2, scpe3 с матрицами размера 50×500. Группы столбцов, соответствующие парам признаков и порогов, сформируем случайным образом.

4.1 Исследование сходимости

Генетический алгоритм Сотнезова имеет гиперпараметры K и C, определяющие динамику числа мутаций. Покажем, что при различном выборе этих параметров результаты экспериментов могут сильно различаться, то есть что число мутаций очень существенно для поставленной задачи.

Для иллюстрации рассмотрим модельную задачу с матрицей размера 1.000×10.000, в каждой строке которой в среднем присутствуют 3 единицы (но не менее одной единицы в каждой строке). Эта же частота появления единиц в строке будет задана и для других сгенерированных матриц.

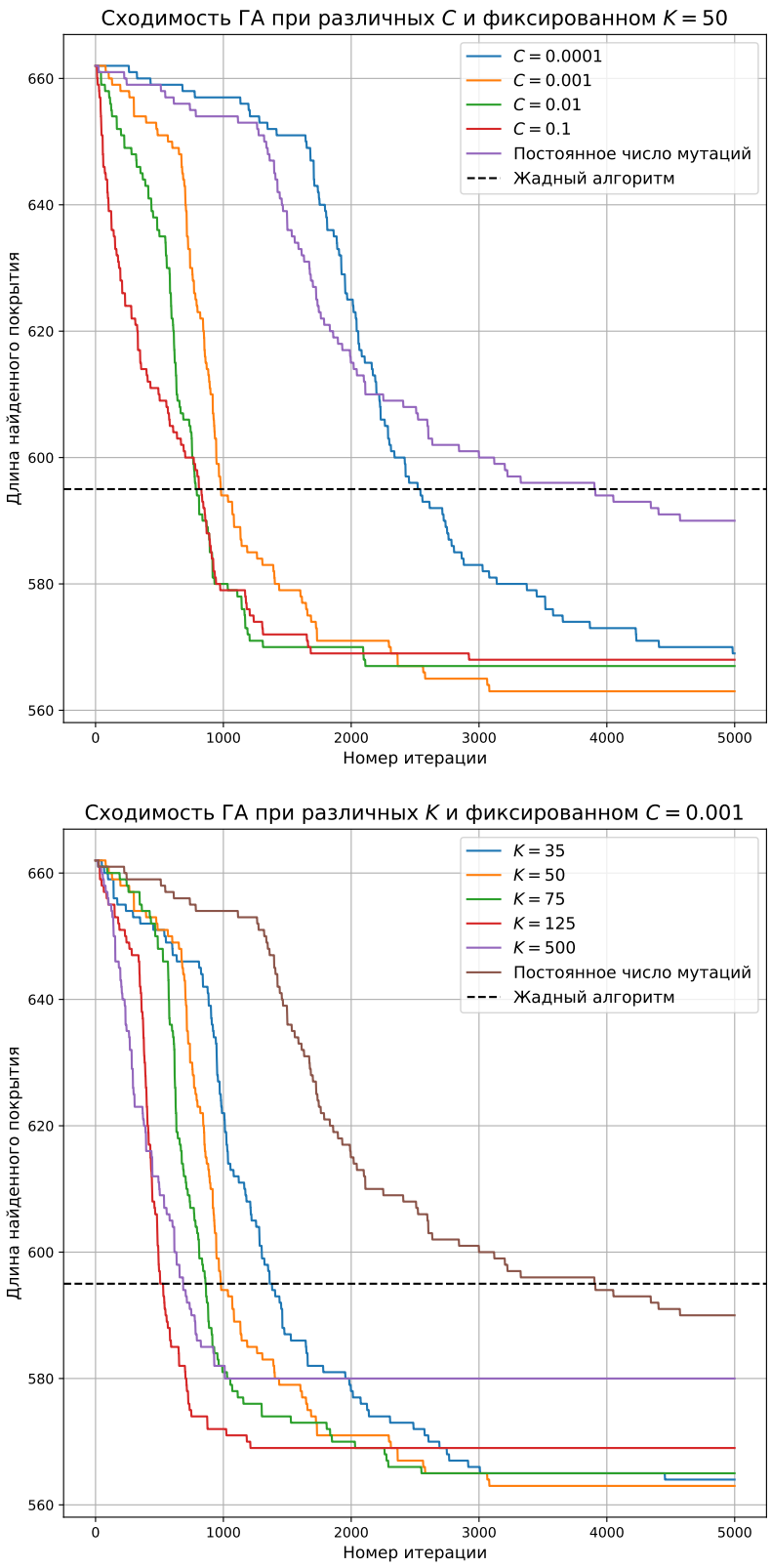

Рис. 1: Сравнение сходимости ГА при различном выборе его параметров на одной задаче

На рис. 1 отображено влияние параметров на сходимость генетического алгоритма. Видно, что при постоянном числе мутаций алгоритм сходится очень медленно, гораздо эффективнее он работает при постепенном увеличении числа мутаций. При этом заметим, что возникновение слишком большого число мутаций (что соответствует большим значениям K) на поздних итерациях алгоритма приводит к ухудшению результата. Это связано с тем, что алгоритм становится слишком случайным, то есть не может стабилизироваться в окрестностях точек минимума. В таком случае наилучшим оказывается решение, полученное в тот момент времени, когда количество мутаций ещё не слишком велико. Аналогично, слишком малые значения C приводят к медленной сходимости, а при слишком больших значениях наблюдается тот же эффект, что и при рассмотрении параметра K.

4.2 Сравнение времени работы

Проведём сравнение градиентного и генетического алгоритмов по времени работы. Обозначим через l длину покрытия, найденного жадным алгоритмом для конкретной задачи.

В таблице для каждого из алгоритмов и соответствующей задачи представлены две характеристики: время поиска покрытия длины l и вес минимального найденного покрытия. Время работы алгоритмов усреднялось в каждом из случаев по 10 запускам.

Задача	Жадный алгоритм	ГА Сотнезова
Сгенерированная (1.000×10.000)	19.6 / 595	26.3 / 564
Сгенерированная (10.000×1.000)	29.8 / 998	8.7 / 998
Сгенерированная (10.000×10.000)	997.1 / 2559	144.7 / 2322
scpe1	0.012 / 5	0.087 / 5
scpe2	0.012 / 5	0.085 / 5
scpe3	0.012 / 5	0.025 / 5

Видно, что для разреженных матриц небольшого размера время работы жадного и генетического алгоритмов оказывается сопоставимым, но при увеличении размеров матрицы жадный алгоритм начинает работать сильно дольше, чем генетический. Кроме того, для всех представленных задач генетический алгоритм находит покрытие веса, не превышающего вес решения жадного алгоритма. Это подтверждает гипотезу о большей эффективности генетического алгоритма для разреженных матриц большого размера.

Список литературы

[1] Н.В. Песков Е.В. Дюкова, А. Инякин. Комбинаторный (логический) анализ данных в распознавании образов. .

[2] П.А. Прокофьев Е.В. Дюкова, Г.О. Масляков. О выборе частичных порядков на множестве значений признаков в задаче классификации. .

[3] Н. В. Федорова. Базисные классификаторы формальной теории классификации технических систем: иерархии, векторы и матрицы, ленты.

[4] Н.В. Песков Е.В. Дюкова. Построение распознающих процедур на базе элементарных классификаторов. .

[5] И.Л. Карнеева Е.В. Дюкова. Модели распознающих алгоритмов, основанные на различных способах перекодировки исходной информации. .

[6] Н.В. Песков Е.В. Дюкова, Ю.И. Журавлёв. Обработка веществennозначной информации логическими процедурами распознавания. .

[7] М. Гэри. Вычислительные машины и труднорешаемые задачи.

[8] A.G. Doig A.H. Land. An autmatic method of solving discrete programming problems.

[9] Ю.Ю. Финкельштейн А.А. Корбут. Дискретное программирование.

[10] V.A. Chvatal. A greedy heuristic for the set-covering problem.

[11] С.А. Ложкин. Лекции по основам кибернетики.

[12] J.H. Holland. Adaptation in natural and artificial systems.

[13] М.Х. Нгуен. Применение генетического алгоритма для решения одной задачи планирования производства.

[14] Р.М. Сотнезов. Генетические алгоритмы для задач логического анализа данных в дискретной оптимизации и распознавании.

[15] И.С. Коновалов. Применение генетического алгоритма для решения задачи покрытия множеств.

[16] T. Sibahara K. Iwamura, M. Horiike. Input data dependency of a genetic algorithm to solve the set covering problem.

[17] D.E. Goldberg. Genetic algorithms in search, optimization, and machine learning.

[18] J.E. Beasley. Or-library: distributing test problems by electronic mail.