А.Г. Дьяконов

МЕТОДЫ РЕШЕНИЯ ЗАДАЧ КЛАССИФИКАЦИИ С КАТЕГОРИАЛЬНЫМИ ПРИЗНАКАМИ[1]

Введение

В работе рассматриваются задачи классификации с категориальными признаками. Категориальный признак – это признак, значения которого обозначают принадлежность объекта к какой-то категории. Примеры таких признаков: национальность, гражданство, профессия, должность, идентификационный номер, номер группы студента, тарифный план, издательство, область науки и т.п. Сами значения признаков для алгоритмов анализа данных бесполезны: на практике разные категории кодируют разными целыми числами, но использовать на таких данных классические методы машинного обучения, ориентированные на вещественные признаки (и операции над ними), нельзя. Для категориальных признаков имеет смысл лишь операция сравнения (совпадают или нет категории). Категориальные признаки также называют «номинальными» [1], среди прикладников больше распространён термин «факторные», поскольку так эти признаки называются в некоторых системах анализа данных (например, в R [2] для их задания используется функция factor).

В прикладных задачах преобладают вещественные признаки. Если в задаче есть категориальные признаки, то их, как правило, немного, поэтому удаётся подобрать подходящую кодировку (ниже мы опишем один из способов кодирования) в признаки, над которыми уже можно выполнять различные арифметические операции. В последние годы появились задачи, в которых почти все или даже все признаки категориальные. Пример подобной задачи – простейшая коллаборативная фильтрация [3]. Есть множество пользователей и услуг, дан перечень, какие пользователи какими услугами пользовались и их оценки «насколько эти услуги им понравились». Необходимо конкретному пользователю предложить новую услугу (т.е. ту, которой он, скорее всего, потом поставит высокую оценку). В такой простейшей постановке – только два категориальных признака: номер пользователя и номер услуги.

Есть задачи и с б***о***льшим числом признаков. В данной работе при описании качества алгоритмов мы будем использовать данные задачи [4]. В этой задаче работники компании просят доступ к ресурсам. Запрос

[1] Работа выполнена при поддержке гранта РФФИ №14-07-00965.

представлен признаковым описанием: номером работника, номером его специальности, отдела, номером менеджера обрабатывающего запрос, номером ресурса и т.п. В этой задаче восемь категориальных признаков, целевой признак (девятый) – бинарный (получил ли работник доступ к ресурсу).

1. Обозначения

Будем считать, что исходная обучающая информация задана стандартной матрицей объект-признак (признаковой матрицей):

$$F = \| f_{ij} \|_{m \times n},$$

здесь m – число объектов, n – число признаков, f_{ij} – значение j-го признака на i-м объекте, а также целевым вектором

$$(y_1, \ldots, y_m)^{\mathrm{T}},$$

здесь y_i – значение целевого признака на i-м объекте. В данной работе рассмотрим задачу классификации с двумя непересекающимися классами. В этом случае целевой вектор бинарный и

$$\{y_1, \ldots, y_m\} = \{0,1\}$$

(считаем, что в обучающей выборке есть представители обоих классов). Необходимо разработать алгоритм, который по признаковому описанию нового объекта $(f_1, \ldots, f_n)$ выдаёт значение его целевого признака y (естественно, с некоторой погрешностью). Качество решения будет формализовано ниже. Совокупность объектов, описания которых перечислены в F, называют обучающей выборкой (или просто «обучением»). Если есть объекты с известной классификаций, на которых тестируют качество алгоритмов, то их совокупность называют контрольной выборкой, а сам метод тестирования – «с отложенным контролем» (hold-out). Также нам потребуется оценка качества методом leave-one-out (по одному): когда из обучения временно изымается один объект, алгоритм обучается на новом обучении и контролируется на этом объекте, процедура проводится для всех объектов, качество усредняется по всем объектам.

Будем считать, что все признаки в задаче категориальные. Вообще, в качестве значений категориальных признаков могут использоваться объекты произвольной природы. Единственное требование – на этих объектах должна быть определена операция сравнения (равно и не равно), чтобы отличать, принадлежат ли разные объекты к одной категории или нет. Мы будем пользоваться этим для описания перехода к новым признакам. Категории всегда можно перенумеровать, поэтому для удобства описания алгоритмов будем считать, что $f_{ij} \in \{1,2,\ldots,n_j\}$ и n_j – число разных категорий j-го признака, $j \in \{1,2,\ldots,n\}$. Заметим, что не обязательно

$$\{f_{1j},\ldots,f_{mj}\}=\{1,2,\ldots,n_j\},$$

т.е. в обучении представлены объекты со всевозможными значениями признака.

Отметим также, что универсальным способом перекодировки категориального признака со значениям из $\{1,2,\ldots,n_*\}$ в «стандартные» без потери информации и для использования классических моделей является замена столбца $(h_1,\ldots,h_m)^{\mathrm{T}}$ матрицы F (соответствующего признака) бинарной матрицей $\|\delta_{ij}\|_{m\times n_*}$,

$$\delta_{ij}=\begin{cases}1, & h_i=j,\\ 0, & h_i\neq j,\end{cases}$$

$i\in\{1,2,\ldots,m\}$, $j\in\{1,2,\ldots,n_*\}$. В западной литературе такая кодировка называется «one hot encoding» и она встроена во многие пакеты для машинного обучения (например в [5]), в российской – устоявшегося термина нет. Мы будем использовать термин «one-hot-кодировка», матрицу $\|\delta_{ij}\|_{m\times n_*}$ будем называть матрицей one-hot-кодировки (рассматриваемого признака) или one-hot-кодом. Заметим, что в случае большого числа категорий (n_*) такая перекодировка всех признаков может существенно увеличить число столбцов в матрице «объект-признак».

В современных задачах классификации от алгоритмов требуют не просто выдачу ответа, но и некоторой дополнительной информации, например уверенности в ответе. В данном случае, можно потребовать, чтобы алгоритм по описанию объекта выдавал значения из отрезка $[0,1]$. В качестве функции ошибки можно использовать стандартные отклонения: квадрата разности $(a-y)^2$ или модуля разности $|a-y|$, a – ответ алгоритма, y – истинное значение метки объекта (эти значения усредняют по всем объектам контрольной выборки). Мы рассмотрим случай, когда алгоритм тестируется на контрольной выборке, а в качестве функционала качества используется площадь под ROC-кривой (receiver operating characteristics): AUC (area under the curve, [6]). Пусть верные метки контрольных объектов $y^1,\ldots,y^q$, а алгоритм выдал на них значения $a^1,\ldots,a^q$. ROC-кривая образуется соединением точек $(\mathrm{fp}(c),\mathrm{tp}(c))$,

$$\mathrm{fp}(c)=\frac{|\{t\in\{1,2,\ldots,q\}\mid y^t=0,a^t\geq c\}|}{|\{t\in\{1,2,\ldots,q\}\mid y^t=0\}|},$$

$$\mathrm{tp}(c)=\frac{|\{t\in\{1,2,\ldots,q\}\mid y^t=1,a^t\geq c\}|}{|\{t\in\{1,2,\ldots,q\}\mid y^t=1\}|},$$

при варьировании порога c и лежит внутри квадрата с вершинами из $\{0,1\}\times\{0,1\}$. Ясно, что площадь под ROC-кривой принадлежит отрезку $[0,1]$ и достигает максимума, если ответы алгоритма на объектах из клас-

са 0 меньше некоторого порога c, а ответы на объектах из класса 1 – больше. Также ясно, что при использовании этого функционала качества можно не требовать, чтобы ответы лежали на отрезке $[0,1]$, функционал инвариантен относительно умножений ответов на положительную константу и сдвигов на константу. Важным свойством значения AUC как меры качества классификатора является то, что оно автоматически учитывает диспропорцию в представителях класса (когда объектов одного класса больше, чем другого). AUC имеет простую вероятностную интерпретацию: это вероятность того, что ответ на случайном объекте из класса 1 будет больше ответа на случайном объекте из класса 0 [6].

Опишем операцию получения новых категориальных признаков на базе имеющихся. На основе признаков $(f_{1j},\ldots,f_{mj})$, $(f_{1t},\ldots,f_{mt})$ сформируем новый признак $((f_{1j},f_{1t}),\ldots,(f_{mj},f_{mt}))$, т.е. описания объектов совпадают по новому признаку тогда и только тогда, когда они совпадают по двум данным. По нашей договорённости о нумерации категорий, пары $(f_{1j},f_{1t}),\ldots,(f_{mj},f_{mt})$ надо заменить на значения из начального отрезка натурального ряда (т.е. занумеровать). Назовём новый признак конъюнкцией двух исходных. Аналогично можно ввести конъюнкцию порядка k: формирование признака на основе k категориальных (считаем, что конъюнкция порядка 1 оставляет признак без изменений). Интересно, что матрица one-hot-кодировки конъюнкции порядка k состоит из всевозможных ненулевых покоординатных произведений столбцов матриц one-hot-кодировок соответствующих признаков, при этом из матрицы каждого из этих k признаков в произведение берётся ровно один столбец. Подробнее немного в других терминах это описано в [7].

2. Сингулярное разложение матриц в анализе данных

Некоторые рассмотренные ниже методы опираются на сингулярное разложение матриц (singular value decomposition, SVD, [8]), поэтому вкратце напомним необходимые для описания методов факты. Сингулярное разложение часто применяется на практике, поскольку эффективные алгоритмы для его реализации известны с 1970х годов [9]. Напомним, что разложение заключается в представлении матрицы Z размера $m\times n$ в виде произведения $U\Lambda V$, где $U_{m\times m}$, $V_{n\times n}$ – ортогональные матрицы, $\Lambda_{m\times n}$ – матрица с элементами $\lambda_1,\ldots,\lambda_r,0,\ldots,0$ на главной диагонали, остальные – нули, $r=\mathrm{rank}(Z)$, удобно считать, что $\lambda_1\geq\ldots\geq\lambda_r>0$. Эти элементы называются сингулярными числами и равны квадратным корням собственных значений матрицы ZZ^{T}. Разложение можно переписать в т.н. сокращённом представлении, считая, что матрица U имеет размеры $m\times r$, V – $r\times n$, Λ – $r\times r$, тогда

$$Z = \sum_{i=1}^{r} \lambda_i u_i v_i^{\mathrm{T}},$$

u_i – i-й столбец матрицы U, v_i^{T} – i-я строка матрицы V. Самым полезным свойством разложения является то, что матрица

$$Z_k = \sum_{i=1}^{k} \lambda_i u_i v_i^{\mathrm{T}} \qquad (1)$$

является наилучшим приближением матрицы Z среди всех матриц ранга k в L_2-норме, $k < \mathrm{rank}(Z)$, причём $\| Z - Z_k \|_2 = \lambda_{k+1}$ (теорема Эккарта–Янга, см. [10]). Поэтому мы будем говорить об усечённом сингулярном разложении (1):

$$Z \approx U_{m\times k} \Lambda_{k\times k} V_{k\times n}.$$

Одно из наиболее частых применений SVD – сокращение размерности пространства. Пусть исходная матрица объект-признак $F = \| f_{ij} \|_{m\times n}$ (в данном случае вещественная, и все признаки вещественные) и число признаков n очень велико. Тогда делают усечённое сингулярное разложение матрицы $F \approx U\Lambda V$ и используют матрицу U в качестве новой матрицы объект-признак. Ясно, что линейные комбинации столбцов матрицы U достаточно точно приближают столбцы исходной матрицы (в смысле теоремы Эккарта–Янга), поэтому матрица U годится для описания маломерного пространства, которое сохраняет информацию об исходном пространстве. Первые два столбца матрицы U (которые соответствуют наибольшим сингулярным значениям) часто используют для визуализации данных. Если известны контрольные объекты, на которых будет работать алгоритм, то раскладывают матрицу признаковых описаний для всех объектов. Если же контрольные объекты не известны, то вместо U используют матрицу $FV^{\mathrm{T}}\Lambda^{-1}$, а при классификации объекта с признаковым описанием $f = (f_1, \ldots, f_n)$ его заменяют на $fV^{\mathrm{T}}\Lambda^{-1}$.

О приложениях SVD, в том числе в многомерном случае (разложения тензоров), можно почитать в обзоре [9]. Ниже мы опишем алгоритм для нахождения аналога сингулярного разложения многомерной разреженной матрицы. Отметим, что в многомерном случае не все свойства SVD справедливы для существующих разложений, мы представим т.н. CP-разложение [11].

3. Реальная прикладная задача с категориальными признаками

В качестве тестовой задачи для исследования описанных ниже методов выбрана задача международного соревнования на базе платформы Kaggle [4]. Для воспроизведения экспериментов достаточно скачать данные с сайта соревнования [4]. Файл train.csv записан в обычном csv-

формате и содержит матрицу «объект-признак» размера 32769x10, первый столбец соответствует целевому признаку, а 7й и 10й столбцы совпадают как факторные признаки (при этом категории нумеруются в каждом столбце по-своему). Поэтому в задаче 8 различных признаков, число категорий в каждом из них показано в табл. 1.

Номер признака	1	2	3	4	5	6	7	8
Число категорий	7518	4243	128	177	449	343	2358	67

Табл. 1. Число категорий для различных признаков

Порядок конъюнкции	1	2	3	4
Число признаков	15283	205298	882617	2076324

Табл. 2. Число признаков при пополнении признакового пространства конъюнкциями

Таким образом, в задаче довольно мало признаков, но они категориальные, поэтому после one-hot-кодировки число признаков превышает 15000, см. табл. 2. Число объектов достаточно велико, что накладывает на исследуемые модели дополнительные требования по времени работы.

Все описанные ниже методы обучались на первых 25000 объектах, остальные объекты были контрольными – на них показано качество классификации. Отметим, что 94.21% объектов в задаче – из класса 1, поэтому мера AUC ROC хорошо подходит для оценки качества.

4. Линейные методы

Под линейными методами мы понимаем методы, которые при классификации объекта $(f_1,\ldots,f_n)$ используют значение линейной комбинации

$$L = w_1 f_1 + \ldots + w_n f_n . \qquad (2)$$

Естественно, $(f_1,\ldots,f_n)$ – не обязательно исходные признаки (чтобы линейная комбинация имела смысл, они должны быть вещественные), а могут получаться из исходных с помощью некоторых преобразований. Пример линейного метода – обычный линейный классификатор, результат действия которого получается как индикатор сравнения с порогом

$$\begin{cases} 1, & L \geq w_0, \\ 0, & L < w_0. \end{cases}$$

Учитывая наш функционал качества AUC, можно использовать значение L в качестве ответа, которое можно перевести на отрезок $[0,1]$ преобразованием

$$\frac{1}{1+e^{-L}}$$

(хотя это не влияет на значение AUC). После выполнения one-hot-кодирования всех категориальных признаков можно применять различные линейные алгоритмы. Заметим, что даже при большом числе категорий и, соответственно, при большом числе признаков после кодирования, матрицу можно хранить в sparse-формате (хранятся только ненулевые элементы) и эффективно работать с ней. Многие современные линейные алгоритмы ориентированы на такое хранение. Как правило, такие алгоритмы, например, реализованные в LIBLINEAR [12], могут эффективно находить линейные разделяющие поверхности SVM-подобными методами, но не могут использовать ядра, т.е. переходить к нелинейным поверхностям. Такой переход можно осуществить «вручную», добавив one-hot-кодирование конъюнкций признаков. Однако, при увеличении степени конъюнкций число признаков уже может увеличиваться значительно. На практике часто используют жадную стратегию для наращивания бинарной признаковой матрицы: пытаются добавить one-hot-код очередной конъюнкции, если качество линейной регрессии на отложенном контроле повышается, то этот код оставляют, иначе удаляют. Затем рассматривают следующую конъюнкцию. В данной работе не будем исследовать такие жадные стратегии.

Простейший персептронный алгоритм [1] настаивается достаточно быстро, показывает неплохое качество (0.8285 AUC) и может быть использован в качестве «бенчмарка» (для сравнения с ним более сложных моделей алгоритмов). Напомним, что в нём последовательно перебираются все объекты обучения и вектор весов $(w_1,\ldots,w_n)$ корректируется в случае, если i-й объект классифицируется неверно: к нему прибавляется

$$\lambda(2y_i-1)(f_{i1},\ldots,f_{in}).$$

Если в процессе перебора объектов не было коррекций весов, то классификатор настроен, иначе объекты перебираются снова (в случайном порядке). В задаче [4] параметр λ оптимальнее было выбирать так

$$\lambda=\frac{1}{\log(s+1)},$$

где s – номер итерации (номер прохождения по обучающей выборки), см. рис. 1. После 2000 итераций качество на контроле начинает медленно понижаться.

Аналогично настраивается логистическая регрессия [1], здесь последовательно пересчитываются веса по следующей формуле

$$(w_1,\ldots,w_n) = (w_1,\ldots,w_n) + \lambda\sum_{i=1}^{m}\left(y_i - \frac{1}{1+e^{-(w_1 f_1+\ldots+w_n f_n)}}\right)(f_1,\ldots,f_n),$$

но качество AUC после настройки существенно выше: 0.8713. Для экспериментов был использован пакет LIBLINEAR [12], в котором реализованы алгоритмы логистической регрессии и SVM для больших разреженных матриц (поэтому его часто используют при классификации текстов). В логистической регрессии можно выбрать тип регуляризации: L1 или L2, а в SVM – вид функции потерь: L1 или L2 [13]. На рис. 2 показана зависимость AUC от коэффициента регуляризации.

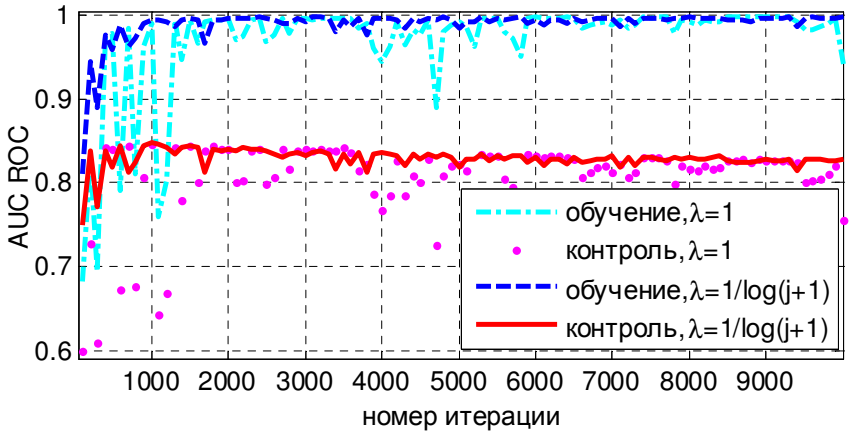

Рис. 1. Качество при настройке простейшего персептронного алгоритма.

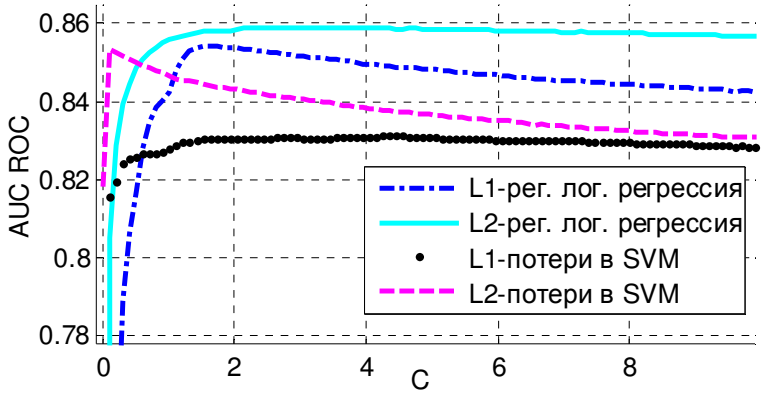

Рис. 2. Качество алгоритмов пакета LIBLINEAR после one-hot-кодирования признаков.

Лучшее качество среди линейных алгоритмов показала логистическая регрессия с L2-регуляризацией. В табл. 3 показано изменение качества AUC при повышении порядка конъюнкции (для каждого $k \in \{1,2,3,4\}$ признаковая матрица состоит из one-hot-кодов всех конъюнкций степени не выше k). Отметим, что здесь мы не проводили селекцию признаков.

Как показало соревнование [4], при отборе признаков логистическая регрессия является лучшим алгоритмом для решения данной задачи.

Порядок конъюнкции	1	2	3	4
ROC AUC	0.8591	0.8703	**0.8713**	0.8704

Табл. 3. Качество логистической регрессии на конъюнкциях разных порядков.

5. Байесовские алгоритмы и их обобщения

От исходного (категориального) признакового описания перейдём к новому (вещественному), для этого значения признаков f_j заменим на g_j – оценки принадлежности к классу 1, полученные по j-му признаку:

$$g_j = \begin{cases} \dfrac{|I_j(f_j) \cap Y_1|}{|I_j(f_j)|}, & f_j \in F_j, \\ \Delta_j, & f_j \notin F_j, \end{cases} \tag{3}$$

где $I_j(f_j) = \{t \in \{1,2,\ldots,m\} \mid f_j = f_{tj}\}$ – номера объектов, у которых значение j-го признака равно f_j, $Y_1 = \{t \in \{1,2,\ldots,m\} \mid y_t = 1\}$ – номера объектов первого класса, $F_j = \{f_{1j},\ldots,f_{mj}\}$ – множество значений j-го признака на обучении. Причина появления константы Δ_j – категория f_j могла не встречаться в значении j-го признака в обучающей выборке, часто на практике полагают

$$g_j = \frac{|I_j(f_j) \cap Y_1| + \Delta_j \cdot c}{|I_j(f_j)| + c},$$

c – коэффициент регуляризации. В наивном байесовском классификаторе [1] для классификации используются значения

$$g_1 \cdot \ldots \cdot g_n. \tag{4}$$

Заметим, что если прологарифмировать полученные новые признаки $g_1,\ldots,g_n$, то значение (4) – это экспонента суммы полученных значений. Поэтому можно перейти от признакового описания $(f_1,\ldots,f_n)$ к описанию

$$(\varphi(g_1),\ldots,\varphi(g_n),\gamma_1,\ldots,\gamma_n), \tag{5}$$

где $\varphi: \mathbf{R} \to \mathbf{R}$ – некоторая функция,

$$\gamma_j = \begin{cases} 0, & f_j \in F_j, \\ 1, & f_j \notin F_j, \end{cases}$$

(признаки-индикаторы неизвестных категорий). При этом можно положить $\Delta_j = 0$ в (3), поскольку, например, при решении задачи в новом признаковом пространстве линейными методами коэффициент при γ_j в линейной комбинации (2) соответствует значению $\varphi(\Delta_j)$ для j-го признака.

Преимущество подобного преобразования признакового пространства в том что

1) в нём можно использовать любые стандартные методы решения задачи,
2) признаковое пространство увеличивается незначительно,
3) получаются легко интерпретируемые кодировки (первая группа признаков – оценки принадлежности к классам с точки зрения отдельных признаков, вторая группа – индикаторы новых категорий),
4) быстро удаётся построить алгоритм неплохого качества (см. ниже).

Основной недостаток в том, что при неумелой реализации этого подхода можно переобучиться (получить алгоритм с низким качеством на независимом контроле). Обычно матрицу F делят на две подматрицы размера $m_1 \times n$ и $(m - m_1) \times n$. По первой осуществляют кодировку (т.е. вычисляют значения g_j), по второй решают задачу в новом признаковом пространстве. Чтобы оценить эффективность алгоритма, нужна ещё и третья матрица (т.н. отложенный контроль [1]). Второй недостаток заключается в том, что g_j – оценка вероятности, поэтому может быть некорректна из-за дефицита информации. Поэтому на практике отбрасывают «небольшие категории»:

$$g_j = \begin{cases} \dfrac{|F_j \cap Y_1|}{|F_j|}, & |I(f_j)| \geq r, \\ 0, & |I(f_j)| < r. \end{cases}$$

Третий недостаток связан с тем, что при кодировке мы учитываем все признаки по отдельности, игнорируя связи между ними. Для устранения этого недостатка изначально признаки пополняют конъюнкциями, однако при этом увеличивается число небольших категорий.

Алгоритм, который в пространстве (5) решает задачу линейным методом, назовём $\varphi(x)$-1-алгоритмом. На практике в качестве $\varphi: \mathbf{R} \to \mathbf{R}$ часто оказывается лучше использовать тождественные функции или функции вида $\varphi(g) = g^k$ (а не логарифм).

Одним из лучших оказался следующий «φ-2-алгоритм»: ответ формируется как

$$L(f_1, \ldots, f_n) = \frac{\sum\limits_{\substack{j=1 \\ f_j \in F_j}}^{n} w_j \varphi(g_j)}{\sum\limits_{\substack{j=1 \\ f_j \in F_j}}^{n} w_j}$$

Данная формула некорректна, если $f_j \notin F_j$ для всех $j \in \{1,2,\ldots,n\}$, понятно, что это происходит в том случае, если категории всех признаков классифицируемого объекта встретились впервые. На практике, как правило, такого не происходит (в этом случае можно в качестве ответа выдавать $(y_1 + \ldots + y_m)/m$).

Веса $w_1,\ldots,w_n \geq 0$ настраиваются методом покоординатного спуска максимизируя AUC на обучающей выборке. На рис. 3 показано, как при этом меняется функционал качества на обучении и контроле. Здесь качество на контроле получилось выше, чем качество на обучении. При использовании конъюнкций признаков происходит наоборот и увеличивается число итераций. Например, при использовании конъюнкций 3й степени качество на обучении 0.8933, на контроле – 0.8864, 276 итераций. Основное время тратится на перекодирование признаков (в оценки вероятностей). В табл. 4 приведена более подробная статистика.

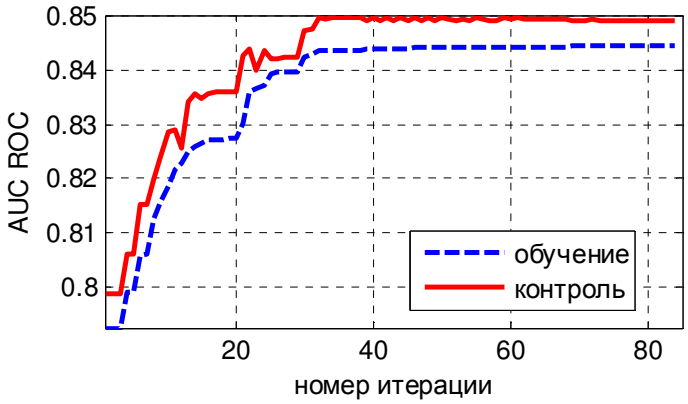

Рис. 3. Качество при настройке байесовского алгоритма методом покоординатного спуска.

Порядок конъюнкции	1	2	3	4	5
x-1-алгоритм	0.8491	0.8814	0.8864	**0.8878**	0.8868
x-2-алгоритм	0.8475	0.8746	0.8801	0.8834	0.8842
$\log(x)$-1-алгоритм	0.8101	0.8234	0.8247	0.8242	0.8246
$\sqrt{x}$-1-алгоритм	0.8465	0.8726	0.8790	0.8809	0.8821

Табл. 4. Качество обобщений байесовских алгоритмов.

6. Методы, основанные на сингулярном разложении матрицы бинарных признаков

В задачах с разреженными матрицами часто используют сингулярное разложение для понижения размерности данных. Например, такой подход даёт неплохие результаты в задачах классификации текстов

(см. [14]). После one-hot-кодирования признаков (в том числе, при использовании конъюнкций) как раз получается сильно разреженная матрица F' «объект-признак». Как было описано выше, матрицу U (или $F'\Lambda^{-1}V^{\mathrm{T}}$) в усечённом сингулярном разложении $F' \approx U\Lambda V$ можно использовать как признаковую матрицу. Дальше задачу часто удаётся решить достаточно простым методом, например гребневой линейной регрессией [1]: решая уравнение

$$Uw = Y,$$

где Y – целевой вектор, т.е.

$$w = (U^{\mathrm{T}}U + \lambda I)^{-1}U^{\mathrm{T}}Y,$$

I – единичная матрица, λ – коэффициент регуляризации. К сожалению, в задаче [4] такой подход не оправдал ожиданий: качество не более 0.8 AUC, см. рис. 4. Заметим, что в данном случае использование регуляризации только ухудшает качество, см. рис. 5 (статистики достаточно, чтобы не было переобучения).

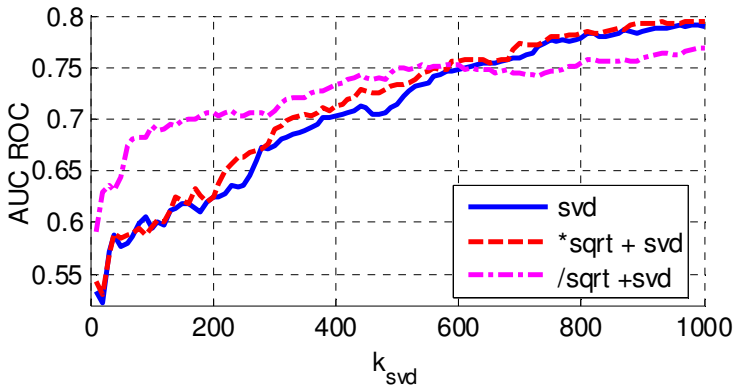

Рис. 4. Качество линейной регрессии после SVD от числа слагаемых в разложении.

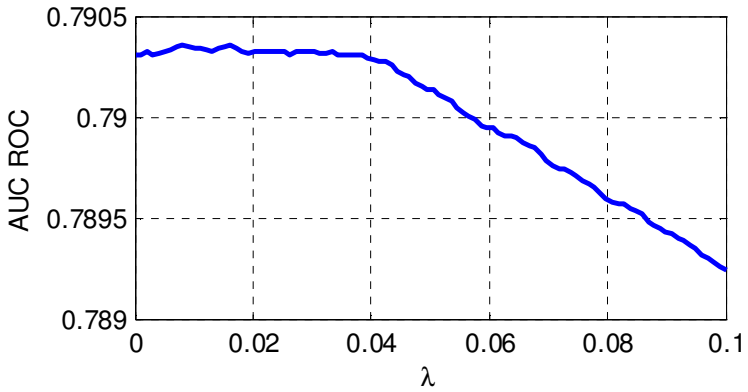

Рис. 5. Зависимость качества классификации от коэффициента регуляризации (при использовании 1000 слагаемых в SVD).

При увеличении числа слагаемых в неполном сингулярном разложении больше 1000 качество повышается медленно, а время вычисления

SVD становится неприемлемым (более 1 часа, эксперименты проводились в среде MATLAB с помощью функции svds). Отметим, что в классификации текстов разложению часто подвергают не исходную матрицу («текст–слово», в которой ij-й элемент – сколько раз в i-м тексте встретилось j-е слово), а матрицу, полученную с помощью tf*idf-преобразования [15]. Смысл преобразования – уменьшить элементы столбцов, в которых много ненулевых элементов (столбцы соответствуют часто встречающимся словам), а также учесть не абсолютные значения, а частоты встречаемости слов в предложениях. Возможно, по аналогии можно делать предварительное преобразование признаковой матрицы после one-hot-кодировки. На рис. 4 показано также качество линейной регрессии после предварительных нормировок вида

$$\eta(\| u_{ij} \|_{m\times n}) = \left\| u_{ij} \Big/ v_j \right\|_{m\times n},$$

$$v_j = \sqrt{\sum_{t=1}^{m} u_{tj}}$$

и

$$\mu(\| u_{ij} \|_{m\times n}) = \left\| u_{ij} v_j \right\|_{m\times n}$$

(деление и умножение на корень суммы элементов в столбце).

7. Методы, основанные на близости

В алгоритмах вычисления оценок [16] (см. также [7]) вычисляются оценки принадлежности к классам

$$\Gamma_y(f_1,\ldots,f_n) = \frac{1}{\mathrm{N}_y} \sum_{\Omega\in\Omega^*} \sum_{i:y_i=y} w^i w_\Omega \mathrm{B}_\Omega((f_1,\ldots,f_n),(f_{i1},\ldots,f_{in})),$$

где Ω^* – система опорных множеств: подмножеств множества признаков $\{1,2,\ldots,n\}$, w_Ω – вес опорного множества Ω, w^i – вес i-го объекта из обучения, N_y – нормирующий множитель, $\mathrm{B}_\Omega(\tilde{f},\tilde{g})$ – функция близости, которая оценивает сходство объектов $\tilde{f}$ и $\tilde{g}$ на опорном множестве Ω. В нашем случае единственная «разумная» функция близости будет обращаться в единицу тогда и только тогда, когда объекты совпадают на данном опорном множестве. Веса всех объектов обучения положим равными единице (в действительности, их сложно настроить по данным без экспертной информации). Поскольку алгоритм должен выдавать вещественное число – будем использовать разность оценок принадлежностей за 1 и 0 классы (возможно, с какими-то коэффициентами, но они могут учитываться в нормировках). Получаем следующий ответ алгоритма:

$$\sum_{i=1}^{m}\left(\frac{2y_i-1}{\mathrm{N}_{y_i}}\sum_{\Omega\in\Omega^*}w_\Omega\prod_{j\in\Omega}I[f_j=f_{ij}]\right),\ I[f_j=f_{ij}]=\begin{cases}1, & f_j=f_{ij},\\ 0, & f_j\neq f_{ij}.\end{cases}$$

При решении задачи [4] выяснилось, что модель можно немного изменить, существенно повысив качество классификации. Чтобы ответ алгоритма был на отрезке $[0,1]$, вычислялась только одна оценка

$$\frac{\sum_{i=1}^{m}r_i y_i}{\sum_{i=1}^{m}r_i},\ r_i=\left(\sum_{\Omega\in\Omega^*}w_\Omega\prod_{j\in\Omega}I[f_j=f_{ij}]\right)^d.$$

Этот метод можно рассматривать и как обобщение взвешенного метода k ближайших соседей [1] (только здесь k совпадает с числом объектов в обучении, и веса зависят от сходства объектов[2]). При реализации метода можно просто перейти к конъюнкциям признаков. Если через f_Ω обозначить конъюнкцию признаков с номерами из $\Omega\subseteq\{1,2,\ldots,n\}$ (значение этого признака у i-го объекта из обучения – $f_{i,\Omega}$), тогда

$$r_i=\left(\sum_{\Omega\in\Omega^*}w_\Omega I[f_\Omega=f_{i,\Omega}]\right)^d,$$

w_Ω естественно проинтерпретировать как вес нового признака. Таким образом, близость между объектом и объектом из обучения вычисляется как нормированная сумма весов всех (новых) признаков, на которых их описания совпадают, в некоторой фиксированной степени. Степень оказалась достаточно удачной эвристикой, как и добавление фиктивного константного признака (по нему все объекты близки).

Параметры модели – веса w_Ω и степень d настраиваются методом покоординатного спуска. Изначально все веса нулевые, степень равна 1, Δ равна 1. Последовательно к каждому весу и степени прибавляем Δ и вычисляем AUC методом leave-one-out (после этого возвращаем старое значение параметра: вычитаем Δ). После перебора всех весов и степени смотрим, при каком изменении было максимальное увеличение AUC – и сохраняем это изменение, если AUC не увеличивался при прибавлении Δ ко всем параметрам, то уменьшаем Δ в два раза. Процесс заканчивается, когда Δ становится меньше некоторого положительного порога.

В задаче [4] этот метод показывал очень хорошее качество, особенно при увеличении степени конъюнкций. При этом стоит отметить, что многие веса после настройки оставались нулевыми, т.е. алгоритм отбирал

[2] Задача [4] – одна из немногих, в которых такие веса максимизируют качество. Обычно в весовых схемах веса не зависят от близости объектов. См., например [17].

значащие сочетания признаков. Однако при повышении степени конъюнкции до 5 время настройки становилось неприемлемым.

Степень конъюнкции	1	2	3	4
Качество	0.8681	0.8884	0.8900	**0.8919**

Табл. 5. Качество метода, основанного на близости

Из недостатков отметим то, что

1) это «ленивая модель» алгоритмов (для классификации необходимо хранить всю обучающую выборку),

2) процедура настройки параметров достаточно долгая.

О целом классе методов, которые являются обобщением ближайшего соседа и успешно работают на практике, см. [18].

8. Методы, основанные на тензорных разложениях

Для начала рассмотрим задачу с двумя категориальными признаками. Обучающие данные

$$\begin{bmatrix} f_{11} & f_{12} \\ \vdots & \vdots \\ f_{m1} & f_{m2} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix},$$

с учётом нашей договорённости, что первый признак принимает значение из $\{1,\ldots,n_1\}$, а второй – из $\{1,\ldots,n_2\}$, можно интерпретировать следующим образом: здесь представлена информация о значениях матрицы $Z = \| z_{ij} \|_{n_1 \times n_2}$:

$$z_{f_{t1},f_{t2}} = y_t$$

для всех $t \in \{1,2,\ldots,m\}$. Значения остальных элементов неизвестны и их надо восстановить, поскольку классификация объекта (f_1, f_2) эквивалентна определению значения элемента z_{f_1,f_2}. Будем искать разложение $Z = UV$, где $U = \| u_{ij} \|_{n_1 \times k}$, $V = \| v_{ij} \|_{k \times n_2}$, минимизируя функционал

$$J = \sum_{t=1}^{m} e_t^2 + \lambda_1 \sum_{s=1}^{k} \sum_{i=1}^{n_1} u_{is}^2 + \lambda_2 \sum_{s=1}^{k} \sum_{j=1}^{n_2} v_{sj}^2 , \qquad (6)$$

где

$$e_t = \sum_{s=1}^{k} (u_{f_{t1},s} v_{s,f_{t2}}) - y_t .$$

Здесь первое слагаемое (сумма e_t^2) отвечает за минимизацию ошибки: элементы $z_{f_{t1},f_{t2}}$ должны быть равны y_t, а остальные слагаемые – регуляризация. Обычно используют два подхода для минимизации (6): метод стохастического градиента (stochastic gradient descent) и чередующиеся

минимизации среднеквадратичной ошибки (alternating least squares). Последний основан на последовательной фиксации одной из матриц: $U = \| u_{ij} \|_{n_1 \times k}$ или $V = \| v_{ij} \|_{k \times n_2}$, тогда относительно параметров, записанных в другой, получаем выпуклую задачу оптимизации. Однако, он обычно медленнее сходится, чем первый. Метод стохастического градиента [19] основан на итерационном изменении настраиваемых параметров в направлении антиградиента. Нетрудно видеть, что

$$\frac{\partial J}{\partial u_{is}} = 2 \sum_{t:\, f_{t1}=i} e_t v_{s,f_{t2}} + 2\lambda_1 u_{is},$$

$$\frac{\partial J}{\partial v_{sj}} = 2 \sum_{t:\, f_{t2}=j} e_t u_{f_{t1},s} + 2\lambda_2 v_{sj}.$$

В качестве начальных приближений задаются случайные матрицы $U = \| u_{ij} \|_{n_1 \times k}$, $V = \| v_{ij} \|_{k \times n_2}$, которые затем пересчитываются по формулам

$$u_{is} = u_{is} - \alpha \sum_{t:\, f_{t1}=i} e_t v_{s,f_{t2}} - \lambda_1 u_{is},$$

$$v_{sj} = v_{sj} - \alpha \sum_{t:\, f_{t2}=j} e_t u_{f_{t1},s} - \lambda_2 v_{sj}.$$

Параметры α и λ, вообще говоря, могут зависеть от номера шага. Подробнее об описанном алгоритме и его модификациях см. в [3].

Аналогично, в общем случае, когда заданы n категориальных признаков, можно считать, что задана информация о m элементах многомерной матрицы размера $n_1 \times n_2 \times \ldots \times n_n$ – тензора n-го порядка [9]. Будем искать матрицы $U(r) = \| u_{ij}^r \|_{n_r \times k}$, $r \in \{1,2,\ldots,m\}$, чтобы минимизировать

$$J = \sum_{t=1}^{m} e_t^{\,2} + \sum_{r=1}^{n} \lambda_r \left(\sum_{i,j} (u_{ij}^r)^2 \right),$$

$$e_t = \sum_{s=1}^{k} \prod_{r=1}^{n} u_{f_{r1},s}^r - y_t.$$

Формулы для пересчёта параметров запишутся следующим образом

$$u_{ij}^r = u_{ij}^r - \alpha \sum_{t:\, f_{tr}=i} e_t \prod_{\substack{d=1 \\ d \neq r}}^{n} u_{f_{td},j}^d - \lambda_r u_{ij}^r.$$

Для простоты (качество не сильно отличалось) приведём результаты настройки алгоритма с формулой пересчёта

$$u_{ij}^r = u_{ij}^r - \alpha \left(e_t \prod_{\substack{d=1 \\ d \neq r}}^{n} u_{f_{td},j}^d - \lambda u_{ij}^r \right),$$

$i \in \{i' \mid f_{tr} = i'\}$, $j \in \{1,2,\ldots,k\}$ (при этом перебираем все $t \in \{1,2,\ldots,m\}$). Опишем результаты применения этого алгоритма в задаче [4]. От исход-

ных меток классов {0,1} лучше перейти к меткам {±1} (это незначительно повышает качество). Изначально матрицы $U(r) = \| u_{ij}^r \|_{n_r \times k}$ следует инициализировать случайными числами около единицы, $\alpha = 0.04$, $\lambda = 0.001$, см. рис. 6. В данной задаче изменение параметра k практически не влияло на результат (качество менялось не более чем на 0.001 AUC). Качество на обучении достигает 0.9851 AUC (и может ещё расти при увеличении числа итераций), на контроле – 0.8468 AUC.

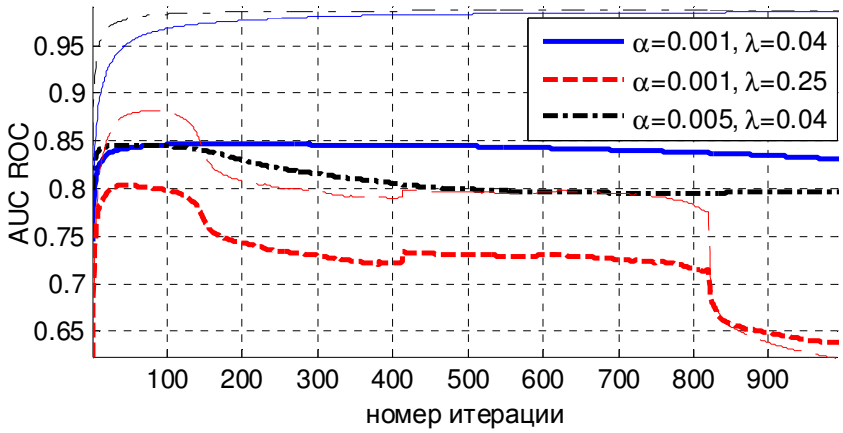

Рис. 6. Качество на обучении (тонкие графики) и контроле (толстые) методом, основанном на тензорном разложении.

9. Методы, основанные на кодировках признаков

One-hot-кодировки, описанные выше, часто подходят для решения задачи линейными методами, но не подходят для многих других, например для решающих деревьев [20] и бустинговых схем над деревьями, по причине слишком большого числа признаков после кодирования. Можно предложить другие способы кодировки, которые позволят применять описанные выше методы.

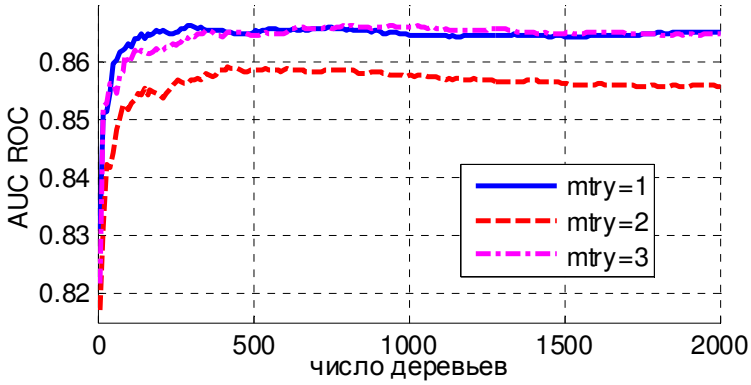

Рис. 7. Качество случайного леса от числа деревьев при случайных кодировках

Иногда используют случайные кодировки: для каждого признака на множество его значений $\{1,\ldots,n_i\}$ действуют случайной перестановкой. После этого строится дерево решений (или небольшой случайный лес). Затем процедура повторяется. Естественно, эти же перестановки действуют и на описания новых объектов (из контроля). Ответы деревьев (лесов) усредняют по всем кодировкам. На рис. 7–8 показано качество случайного леса (random forest) после случайных кодировок (признаки перекодировались после построения 10 деревьев) – оно достигает 0.8664 AUC. На рис. 7 – от числа деревьев, на рис. 8 – от значения ключевого параметра mtry метода «Случайный лес» (число выбираемых случайно вершин при построении ветвления в дереве) [20], [2]. Ниже покажем, что можно использовать более логичные кодировки, после применения которых значения признаков можно интерпретировать. При этом они существенно повышают качество стандартных методов.

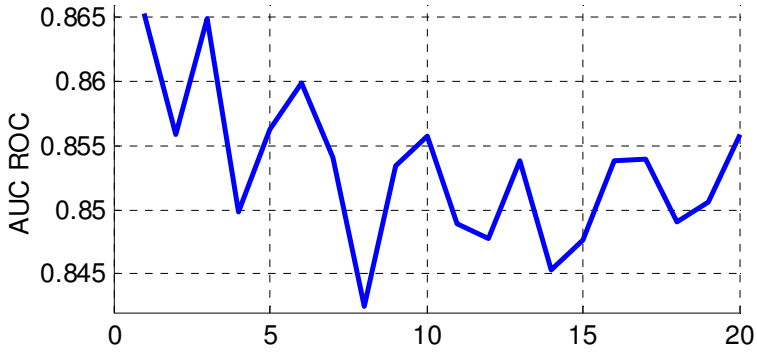

Рис. 8. Качество случайного леса от параметра mtry при случайных кодировках

В задачах, где встречаются и категориальные и вещественные признаки, хорошо работает следующая кодировка. Пусть Ф – некоторое множество вещественных функций, в котором для произвольного натурального числа k есть ровно одна функция k переменных. При этом все функции симметричные, т.е. для любой функции φ от k переменных из Ф

$$\varphi(x_1,\ldots,x_k)=\varphi(x_{\sigma(1)},\ldots,x_{\sigma(k)})$$

для любой перестановки σ. Примером таких множеств может быть множество сумм:

$$\varphi(x_1,\ldots,x_k)=x_1+\ldots+x_k,$$

средних арифметических, максимумов и т.д. Для кодирования значения f_j j-го категориального признака выбираем множество объектов из обучения с таким значением:

$$I=\{t\in\{1,2,\ldots,m\}\mid f_{tj}=f_j\},$$

выбираем вещественный признак, относительно которого будем кодировать, например s-й, и кодируем значение f_j значением подходящей функции из Φ (т.е. функции от $|I|$ переменных) от значений f_{is}, $i \in I$. Такой приём успешно применялся на практике для решения задачи анализа поведений клиентов Интернет-магазина [21][3]. Получаемые новые признаки (для каждого кодируемого признака мы можем менять кодирующий признак и множество функций Φ) имели также простую интерпретацию. Например, пусть кодируемый признак – категория товара (электроника, бытовая техника, книги и т.п.), кодирующий признак – цена, множество Φ описывает средние арифметические, тогда номер категории мы заменяем на среднюю цену товаров в данной категории. Если в качестве кодирующего признака выбрать константный (равен единице на всех объектах, его часто добавляют в задачу), а множество Φ состоит из сумм, то номер категории заменяется на число товаров в этой категории.

К сожалению, описанный подход не всегда может быть успешно применён в задачах, где все признаки факторные. Предложим новые способы кодирования в таких задачах. Выберем пару признаков: кодируемый и кодирующий. Пусть для определённости это первый и второй. Напомним, что, с учётом нашей договорённости, первый признак принимает значения из $\{1,\ldots,n_1\}$, а второй – из $\{1,\ldots,n_2\}$. Пусть $P = \| p_{ij} \|_{n_1 \times n_2}$, где

$$p_{ij} = |\{t \in \{1,2,\ldots,m\} | f_{t1} = i, f_{t2} = j\}|,$$

т.е. ij-й элемент матрицы равен числу объектов, у которых первый признак равен i, а второй – j. При построении матрицы P разумно использовать все имеющиеся объекты (не обязательно с известной классификацией).

В принципе, можно предложить следующий способ кодирования: значение i заменять на i-е значение фиксированного столбца матрицы P. Но при таком способе опять слишком возрастает число признаков (поскольку надо перебрать все столбцы матрицы P). Поэтому сделаем неполное (первые k слагаемых) сингулярное разложение матрицы $P \approx U\Lambda V$. Получаем k различных кодировок: в t-й кодировке заменяем значение i на it-й элемент матрицы U.

Предложенный метод имеет понятную интерпретацию. В матрице P хранится информация о связи между двумя признаками. Например, если это одинаковые признаки (отличаются только нумерацией категорий), то это матрица диагональная с точностью до перестановки столбцов. Сингулярное разложение позволяет сохранить максимум информации о

[3] С помощью описанных кодировок была выиграна Интернет-олимпиада Викимарта [21] (соревнования, в котором 47 участников разрабатывали алгоритмы прогнозирования действий клиентов по реальным данным компании WikiMart).

связи признаков, используя минимум кодирующих векторов. С помощью этого метода после кодирования сохраняются важные свойства исходной информации. Например, если в задаче [4] один из признаков – идентификаторы пользователей, то похожие пользователи при кодировке получат похожие значения признаков. Похожие пользователи – это пользователи, которые обращаются с запросами к одним и тем же ресурсам, работают в одном отделе, у них одинаковая должность и т.п.

Также отметим, что для кодирования второго признака относительно первого не надо проводить симметричную процедуру: k нужных кодировок соответствуют строкам матрицы V. Вместо кодирующего можно брать не обязательно какой-то реально существующий признак, а, например, конъюнкцию всех признаков, кроме кодируемого.

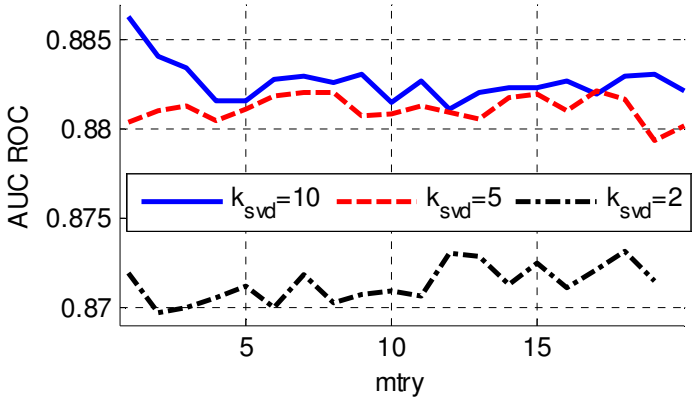

Рис. 9. Зависимость качества от параметра mtry случайного леса.

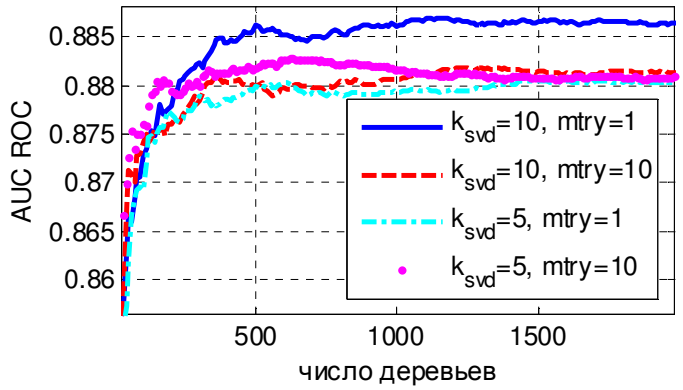

Рис. 10. Зависимость качества от числа деревьев в случайном лесе.

В задаче [4] перебирались все пары признаков, поочерёдно один считался кодирующим, другой кодируемым. Объединение всех матриц кодировок было новой признаковой матрицей. Однако выяснилось, что при наборе пар достаточно брать первый признак из признаков, описывающих пользователей, а второй – из признаков, описывающих ресурсы (их было всего 2). Наиболее эффективным на новой признаковой матрице

оказался метод случайного леса: качество достигает 0.8863 AUC. Как всегда, при повышении числа деревьев в лесе качество увеличивается (после построения 2000 деревьев качество меняется незначительно). От параметров метода «Случайный лес» качество зависит существенно слабее, чем от числа слагаемых k в неполном SVD, см. рис. 9. При увеличении k качество повышается, но при этом увеличивается число признаков (после кодирования), кроме того, увеличивается время сингулярного разложения и требования к оперативной памяти (больше 10 не удалось выполнить в 32битной версии Windows).

10. Выпуклые комбинации алгоритмов

На практике почти всегда строят разные алгоритмы и используют их ансамбли [1], [14]. Данная работа посвящена обзору различных моделей алгоритмов, тем не менее, рассмотрим также простейшие ансамбли – выпуклые линейные комбинации. Для двух алгоритмов с ответами A_1, A_2, ответ комбинации получают в виде $\alpha A_1+(1-\alpha)A_2$, $\alpha \in [0,1]$ – параметр для настройки. В табл. 6 показано качество таких комбинаций (при оптимальном выборе параметра), в табл. 7 – коэффициенты корреляции между векторами ответов на контрольной выборке алгоритмов из пар. Алгоритмы идентифицируются номерами разделов, в которых они описаны. Так, лучшей линейной комбинацией стал ансамбль байесовского алгоритма и случайного леса (после предложенной кодировки в вещественные признаки).

алгоритмы	4	5	7	8	9
4	0.8713	0.8914	0.8919	0.8731	0.8879
5		0.8878	0.8972	0.8884	**0.8974**
7			0.8919	0.8924	0.8919
8				0.8453	0.8872
9					0.8863

Табл. 6. Качество лучших линейных комбинаций

алгоритмы	4	5	7	8	9
4	1	**0.9523**	0.9543	0.9696	0.9568
5		1	0.9989	0.9858	0.9983
7			1	0.9866	0.9989
8				1	0.9880
9					1

Табл. 7. Корреляция ответов алгоритмов

Также представляет интерес несклонность алгоритмов к «переобучению» (показывать на практике меньшее качество, чем при настройке). Во всех таблицах качество было приведено на отложенном контроле (hold out). Дополнительно был проведён контроль на сайте [4] (на данных, на которых организаторы соревнования определяли победителя: 58921 объектов, на отложенном контроле было лишь 7769). Как видно из рис. 12 на «большом контроле» качество только увеличилось. Наилучшим оказался случайный лес после перекодировки признаков в вещественные. Из популярных современных методов в работе не упомянут лишь LibFM [22], который решал задачу [4] хуже рассмотренных методов, но показывает высокое качество в задачах коллаборативной фильтрации.

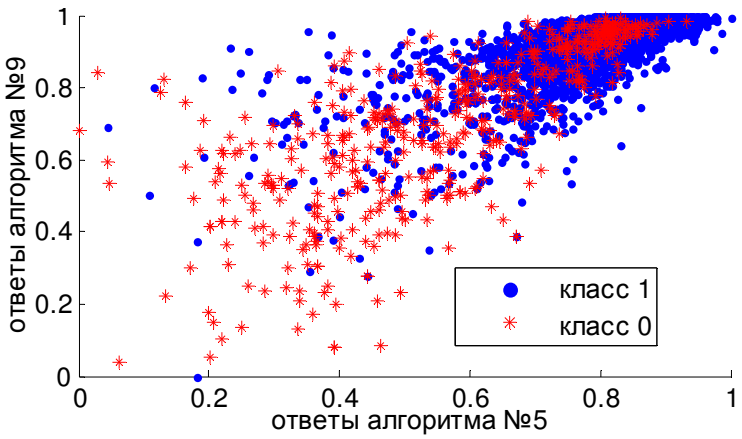

Рис. 11. Ответы двух алгоритмов.

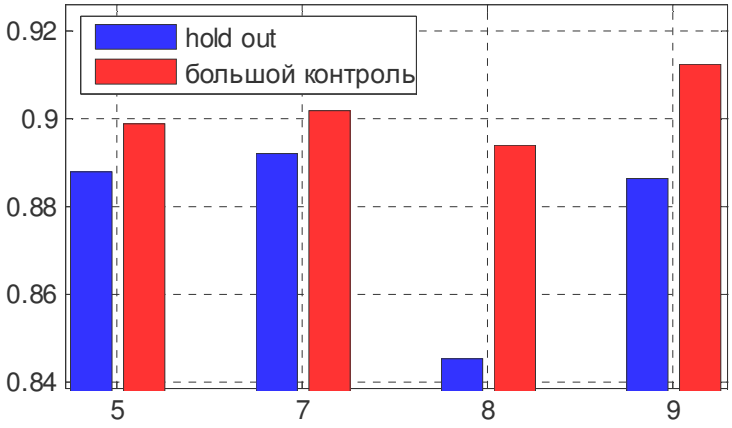

Рис. 12. Качество при проверке на переобучение.

Заключение

В работе рассмотрены различные модели алгоритмов, которые предназначены для решения задач классификации с категориальными признаками. Некоторые из них являются простейшими обобщениями классических алгоритмов (байесовских, SVD), другие – принципиально новыми: кодировки для использования случайных лесов и обобщение алгоритмов, основанных на близости. Как показала практика (не только при решении рассмотренной задачи), предложенные алгоритмы являются одними из лучших. Алгоритмы, основанные на близости, дольше настраивать (хотя они настраиваются непосредственно на нужный функционал), и при классификации они требуют хранения всей обучающей выборки. Случайные леса при использовании кодировок оказались более удобными для прикладного применения алгоритмами.

Мы подробно описали решение задачи с двумя непересекающимися классами, но решали её как задачу регрессии: получали ответ из отрезка $[0, 1]$ и оптимизировали функционал AUC ROC. Ясно, что рассмотренные алгоритмы легко обобщаются на случай многоклассовых задач и задачи восстановления регрессии.

Список литературы

1. Воронцов К.В. Машинное обучение, лекции // http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf
2. Система для статистического анализа данных R // http://cran.r-project.org
3. Y. Koren, R.M. Bell, C. Volinsky Matrix Factorization Techniques for Recommender Systems // IEEE Computer 42(8): 30-37 (2009).
4. Международное соревнование «Amazon.com – Employee Access Challenge» по анализу данных // http://www.kaggle.com/c/amazon-employee-access-challenge
5. Библиотека scikit-learn для языка Python // https://github.com/scikit-learn/scikit-learn
6. T. Fawcett An introduction to ROC analysis // Pattern Recognition Letters 27 (2006) 861–8.
7. Дьяконов А.Г. Теория систем эквивалентностей для описания алгебраических замыканий обобщенной модели вычисления

оценок // Журнал вычислительной математики и математической физики, 2010, Т. 50, №2. С.388-400.

8. G. Strang Linear Algebra and its Applications, fourth edition, Thomson Brooks/Cole, 2005.
9. Carla D. Martin, Mason A. Porter The Extraordinary SVD // American Mathematical Monthly, Vol. 119, No. 10: 838-851 (2012).
10. G.H. Golub, C.F. Van Loan, Matrix Computations, third edition, The Johns Hopkins University Press, Baltimore, MD, 1996.
11. T.G. Kolda, B.W. Bader Tensor Decompositions and Applications // SIAM Review 51(3):455-500, September 2009.
12. A Library for Large Linear Classification http://www.csie.ntu.edu.tw/~cjlin/liblinear/
13. C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan A Dual Coordinate Descent Method For Large-Scale Linear SVM // ICML 2008.
14. A. D'yakonov A Blending of Simple Algorithms for Topical Classification // Rough Sets and Current Trends in Computing, Lecture Notes in Computer Science, 2012, Volume 7413/2012, 432–438. http://www.springerlink.com/content/73g4kl50m6112420/
15. Маннинг К. Д., Рагхаван П., Шютце Х. Введение в информационный поиск. – М.: ООО «И.Д. Вильямс», 2011. – 528 с.
16. Журавлёв Ю.И. Об алгебраическом подходе к решению задач распознавания или классификации // Пробл. кибернетики. – М.: Наука, 1978. – Вып. 33. – С. 5–68.
17. Дьяконов А.Г. Прогноз поведения клиентов супермаркетов с помощью весовых схем оценок вероятностей и плотностей // Бизнес-информатика. 2014 (в печати).
18. D'yakonov A.G. Two Recommendation Algorithms Based on Deformed Linear Combinations // Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 21-28 (2011).
19. S. Funk Netflix Update: Try This at Home // http://sifter.org/~simon/journal/20061211.html.

20. L. Breiman Random Forests // Machine Learning 45 (1): 5-32 (2001).
21. Соревнование по анализу данных «Олимпиада ВикиМарта» // http://olymp.wikimart.ru/
22. S. Rendle Factorization Machines with libFM // ACM Trans. Intell. Syst. Technol. 3(3) 57:1-57:22 (2012).