АНАЛИЗ ПОЛЬЗОВАТЕЛЬСКИХ ПРЕДПОЧТЕНИЙ ДЛЯ ПРОФИЛИРОВАНИЯ ПОСЕТИТЕЛЕЙ ИНТЕРНЕТ-РЕСУРСА

© 2016 г. В.Ю. ДЕНИСОВ, И.С. СИНЕВА

Московский технический университет связи и информатики
e-mail: denisov@sports.ru

Бизнес-модель развития цифровых издательств основана на увеличении их посещаемости. Поэтому актуальна задача удержания старых пользователей и привлечения новых. Сейчас недостаточно только производить контент, очень важно его правильно предлагать и показывать потребителю ранее неизвестные ему варианты, которые его, возможно, смогут заинтересовать.

Профилирование (в информационной науке) включает в себя сбор данных об активности посетителей интернет-ресурсов, которые в дальнейшем будут анализироваться с целью построения модели персонализированных рекомендаций [1]. Законодательной основой для подобного сбора информации является принятие пользователем соглашения.

Главная задача профилирования пользователей заключается в автоматической фильтрации всего контента ресурса под конкретного пользователя. В случае ручной фильтрации читатель сам выбирает интересные ему темы, в результате чего получает определенный набор контента. Цель профилирования – предложение пользователю материалов, которые смогут его заинтересовать, тем самым, увеличив список интересных ему тем, и, скорее всего, активность на сайте [2]. Например, человек, интересующийся дрелью, штукатуркой и обоями вполне возможно интересуется темой ремонта. Таким образом, логично предлагать этому пользователю материалы про освещение, плитку, отвертки и так далее.

Основной площадкой для разработки, тестирования и внедрения системы рекомендаций будет раздел личных лент веб-сайта Sports.ru. Личные ленты включают себя набор из всех имеющихся на ресурсе типов контента: новости, статьи, блоги, теги, статусы, видео- и фотогалереи.

Профилирование пользователей будет проводиться в два этапа: создание инициализирующей модели рекомендаций, отражающей (частично) основные закономерности, и построение финальной модели после полной обработки. Такой подход обуславливается недостаточностью исходных данных: первая модель будет основана только на пользовательский подписках на теги, вторая же будет использовать данные о пользовательской активности за время работы первой модели.

Подписки на теги хранятся в базе данных в формате {*user_id*; *tag_id*}, где ключом является только комбинация из двух значений. Из этих данных создадим матрицу T размера $m \times n$, где m – число уникальных *user_id*, n – число уникальных *tag_id*. Элементы данной матрицы заполняются по следующему правилу:

$$t_{i,j} = \begin{cases} 1, \text{если } i-\text{й пользователь подписан на } j-\text{й тег} \\ 0, \text{иначе} \end{cases}$$

Заранее стоит исключить неактивных пользователей (дата последнего визита раньше 1 июня 2016 года и менее 3 тегов в подписках) и непопулярные теги (менее 10 подписок). Таким образом, матрица T имеет размер порядка 22000×4000.

Для выделения групп в массиве данных принято использовать обучение с учителем (классификация) или обучение без учителя (кластеризация). Внутри каждой

группы должны оказаться похожие объекты. Отличие классификации от кластеризации заключается в наличии обучающего вектора (или матрицы), благодаря значениям которого производится разбиение на кластеры.

В качестве метода классификации нужно отметить *Extreme Gradient Boosting* [3], который является достаточно популярным среди победителей и финалистов различных соревнований по машинному обучению [4]. Градиентный бустинг относится к жадным алгоритмам обучения с учителем. Как и другие алгоритмы бустинга, основной принцип работы данного метода заключается в создании сложной композиции из более простых алгоритмов обучения. Рассматривается проблема распознавания элементов многомерного пространства X с пространством меток Y. Искомый алгоритм запишем в следующем виде:

$$F_M(x) = \sum_{m=1}^{M} b_m h(x, a_m), \tag{1}$$

где $h(x, a_m)$ – оценочная функция, $a_m \in A$; $b_m \in \mathbb{R}$, A – матрица параметров. Жадность алгоритма (1) заключается в добавлении на каждом шаге слагаемого, являющегося наиболее оптимальным из возможных алгоритмом [5]. Задача алгоритма – найти оптимальный вектор $\{a, b\}$. Таким образом, зная классификатор F_{m-1} длины $m-1$, можем построить классификатор длины m:

$$F_m(x) = F_{m-1}(x) + b_m h(x, a_m) \tag{2}$$

Введем функцию потерь $L\big(y_i, F_m(x_i)\big), i = \overline{1, N}$ (это функция, которая характеризует потери при неправильном принятии решений) и проведем минимизацию функционала ошибки:

$$Q = \sum_{i=1}^{N} L\big(y_i, F_m(x_i)\big), \tag{3}$$

которая осуществляется при помощи метода градиентного спуска. Найдем градиент (3):

$$\nabla Q = \left[\frac{\partial Q}{\partial F_{m-1}}(x_i)\right]_{i=1}^{N} = \left[\frac{\partial \sum_{i=1}^{N} L\big(y_i, F_m(x_i)\big)}{\partial F_{m-1}}(x_i)\right]_{i=1}^{N} = \left[\frac{\partial L\big(y_i, F_m(x_i)\big)}{\partial F_{m-1}}(x_i)\right]_{i=1}^{N} \tag{4}$$

для последовательного подсчета векторов a и b:

$$a_m = \underset{a \in A}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(\nabla Q_i, h(x_i, a)\big), \tag{5}$$

$$b_m = \underset{b \in R}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, F_{m-1}(x_i) - b h(x_i, a_m)\big). \tag{6}$$

Особенно эффективен данный метод при построении дерева решений (*Tree-Boost*). На каждом шаге описанного ранее алгоритма строится дерево с J вершинами, соответствующими непересекающимся областям $\{R_j\}, j = \overline{1, J}$. Каждой вершине соответствует коэффициент b_j (4-6). Это описывается следующей формулой:

$$h(x, \{a_j, R_j\}) = \sum_{j=1}^{J} a_j I[x \in R_j], \tag{7}$$

где $I = \{0,1\}$ – индикатор события (только один ненулевой на всю сумму).
Тогда из (2) и (7) следует, что классификатор можно записать в виде

$$F_m(x) = F_{m-1}(x) + b_m \sum_{j=1}^{J} a_{jm} I[x \in R_j] = \sum_{j=1}^{J} c_{jm} I[x \in R_j], c_{jm} = a_{jm} b_m \tag{8}$$

Оптимальные значения c_{jm} для (8) из-за непересекающихся областей R_j находятся по следующей формуле:

$$c_{jm} = \underset{c}{\operatorname{argmin}} \sum_{x_j \in R_{jm}} L(y_i, F_{m-1}(x_i) + c)\,. \tag{9}$$

Для задачи регрессии ускорить бустинг можно, выбрав в качестве функции потерь модуль отклонения (данный метод называется *LAD-boosting*):

$$L(y,F) = |y - F|,$$
$$\nabla Q_i = sign\big(y_i - F_{m-1}(x_i)\big). \tag{10}$$

Тогда:

$$b_m = \underset{b\in R}{\operatorname{argmin}} \sum_{i=1}^{N} |y_i - F_{m-1}(x_i) - bh(x_i,a_m)| = \underset{b\in R}{\operatorname{argmin}} \sum_{i=1}^{N} |h(x_i,a_m)| \left| \frac{y_i - F_{m-1}(x_i)}{h(x_i,a_m)} - b \right|$$
$$= median_W \left\{ \frac{y_i - F_{m-1}(x_i)}{h(x_i,a_m)} \right\}, W = h(x_i,a_m), i = \overline{1,N} \tag{11}$$

Таким образом, задачу удалось свести к поиску порядковой статистики в массиве, что возможно выполнить за $O(N)$ операций.

Комбинация из *Tree-Boost* и *LAD-boosting* является достаточно быстрой и устойчивой к шуму, так как медианы устойчивы к выбросам, а $\nabla Q_i = \{-1, +1\}$.

Для решения задачи бинарной классификации используется функция потерь от одного аргумента $L(y,F) = L(yF)$, т.е. отступ заменяется произведением настоящего класса и предсказанного значения. Тогда градиент можно записать в следующем виде:

$$\nabla Q = \left[\frac{\partial L(y_i F_{m-1})}{\partial F_{m-1}}(x_i) \right]_{i=1}^{N} = \left[y_i \frac{\partial \sum_{i=1}^{N} L(y_i F_{m-1})}{\partial (y_i F_{m-1})}(x_i) \right]_{i=1}^{N}. \tag{12}$$

Алгоритм обучения можно описать следующим уравнением:

$$h(x,a_m) = \underset{a_m}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(\nabla Q_i h(x_i,a_m)\big) = \underset{a_m}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i w_i h(x_i,a_m)\big), \tag{13}$$

где $w_i = \frac{\partial L(y_i F_{m-1})}{\partial F_{m-1}}$ – веса.

Для мультиклассовой классификации используется другая функция потерь :

$$L(y,F) = -\sum_{i=1}^{k} y_i \ln p_i(x), \tag{14}$$

где $y_i \epsilon \{0;1\}$ и p_i – принадлежность объекта и вероятность принадлежноси к *i*-му классу, соответственно. Градиент будет иметь следующий вид:

$$\nabla Q_i = y_{ik} - p_{k,m-1}(x_i) \tag{15}$$

Итоговый классификатор ищет такой класс, чтобы принадлежность объекта к другим классам была минимальна:

$$k(x) = \underset{k\epsilon[1;K]}{\operatorname{argmin}} \sum_{k=1}^{K} c\big(k,\tilde{k}\big) p_{\tilde{k}M}(x), \tag{16}$$

где K – число кластеров, $c\big(k,\tilde{k}\big)$ – функция стоимости ошибки (если предсказан класс k, а на самом деле класс $\tilde{k}$), $p_{\tilde{k}M}$ – вероятность принадлежности классу $\tilde{k}$ на -ой итерации.

Характер исходной выборки вынуждает использовать обучение без учителя. Для матрицы, состоящей из элементов $\{0;1\}$, целесообразно использовать метод бинарной кластеризации *Proximus* [6].

На первом шаге данного алгоритма проводится факторизация исходной матрицы. Факторизация – разложение некой матрицы A на векторы x и y, такие, что $A = xy^T$. Это возможно сделать при помощи двух следующих алгоритмов, которые показывают хорошую сходимость на больших массивах бинарных данных. Первый – дискретный – заключается в минимизации ошибки:

$$\|A = xy^T\|_F^2 = \|A\|_F^2 - 2x^T Ay + \|x\|_2^2 \|y\|_2^2, \tag{17}$$

что эквивалентно максимизации функции

$$C_d(x,y) = 2x^T Ay - \|x\|_2^2 \|y\|_2^2. \tag{18}$$

Если зафиксировать y и обозначить $s = Ay$, то x, максимизирующий функцию, задаётся следующим уравнением:

$$x(i) = \begin{cases} 1, \text{если } 2s(i) \geq \|y\|_2^2 \\ 0, \text{иначе} \end{cases}. \quad (19)$$

Отсюда следует, что ненулевой элемент x вносит положительный вклад в $C_d(x,y)$ тогда и только тогда, когда как минимум половина элементов y совпадают с соответствующим столбцом A. Аналогично вычисляется y, где $x = x'$ (посчитанный на предыдущей итерации вектор). Данная процедура продолжается итеративно, пока сходимость не будет достигнута. Данный алгоритм сходится в ближайшему локальному максимуму, этого позволяет избежать второй метод – непрерывный. Определим функцию, которую необходимо максимизировать:

$$C_c(x,y) = \frac{(2x^T Ay)^2}{\|x\|_2^2 \|y\|_2^2}. \quad (20)$$

Глобальный максимум функций $C_d(x,y)$ (18) и $C_c(x,y)$ (20) не находится в одной точке, однако их поведение высоко коррелировано, .поэтому $C_c(x,y)$ можно использовать как непрерывную аппроксимацию $C_d(x,y)$. Зафиксировав y и определив $s = \frac{Ay}{\|y\|_2^2}$, необходимо максимизировать функцию

$$C_c(x,y) = \frac{(x^T s)^2}{\|x\|_2^2}. \quad (21)$$

Необходимо отметить, что скорость и качество этих алгоритмов зависит от начальных значений выходных векторов x и y.

Полученная факторизация используется для разбиения строк исходной матрицы на две подматрицы A_0 и A_1. Это осуществляется по следующему правилу:

$$A(i) = \begin{cases} A_1, \text{если } x(i) = 1 \\ A_0, \text{иначе} \end{cases}, 1 \leq i \leq m, \quad (22)$$

где i – номер строки матрицы A.

Видно, что факторизация (22) A не даёт никакой информации об A_0, и позже будет использована факторизация A_0 и ее рекурсивное разбиение. С другой стороны, проверяется адекватность строк A_1 при помощи вектора y и критерия сходимости. Если критерий достигнут, считается, что $A_1 \approx xy^T$, иначе производится рекурсивное разбиение как A_0, так и A_1. Критерий адекватности факторизации – нормированный радиус Хэмминга для расстояний Хэмминга. Расстояние Хэмминга определяется по формуле

$$\hat{h}(x,y) = \frac{\|x \; XOR \; y\|}{n} = \frac{x^T x + y^T y + 2x^T y}{n}, \quad (23)$$

где $\|x\| = \|x\|_2^2 = \|x\|_1$ – число ненулевых значений бинарного вектора x. Тогда, имея набор бинарных векторов $X = \{x_1, x_2, \dots, x_n\}$ и вектор y, можно определить радиус Хэмминга:

$$\hat{r}(X,y) = \max_{1 \leq i \leq n} \hat{h}(x,y). \quad (24)$$

Итеративный процесс останавливается, когда радиус Хэмминга $\hat{r}(X,y)$ меньше заданного числа ε. Данный метод может быть использован для различных задач, среди которых стоит отметить обнаружение шаблонов, совместную фильтрация, классификацию и кластеризацию.

После того, как временные рекомендации получены, можно приступать к сбору данных для последующего обучения с учителем. Данные о пользовательской активности сохраняются в лог-файл, который впоследствии обрабатывается. События разбиваются на две группы: просмотры единиц контента и иное взаимодействие с контентом (изменение рейтинга, открытие комментариев и написание нового, удаление рекомендации и т.д.). Для каждого события заранее вычисляется определенный коэффициент в интервале $[-1; 1]$ (зачастую положительный, кроме событий удаления рекомендации и отрицательного изменения рейтинга). В дальнейшем создаётся матрица X размера аналогичного матрице T, с единственным изменением – элементы матрицы теперь обозначаются следующим образом:

$$t_{i,j} = \sum_{m=1}^{M_{i,j}} k_m,$$

где $M_{i,j}$ – число просмотренного контента *i*-м пользователем по *j*-му тегу, k_m – коэффициент события.

Аналогично первому этапу здесь применяется обучение без учителя, однако используется метод k-средних. Его суть в минимизации расстояния от точек кластера до его центра [7]. Это означает минимизацию функции

$$V = \sum_{i=1}^{k} \sum_{x(j)\epsilon S_i} (x(j) - \mu(i))^2, \quad (25)$$

где k – число кластеров, S_i – полученные кластеры, $\mu(i)$ – центры масс векторов $x(j)\epsilon S_i$.

В качестве меры близости для расчетов в (25) было использовано Евклидово расстояние:

$$\rho(x, y) = \|x - y\| = \sqrt{\sum_{p=1}^{n} (x(p) - y(p))^2}, \quad (26)$$

где $\mathrm{x, y} \in \mathbb{R}^{\mathrm{n}}$.

Определить оптимальное число кластеров в (25-26) можно с помощью анализа силуэтов, алгоритм которого заключается в нахождении для каждой точки i кластера средней непохожести $\mathrm{a_i}$ на другие элементы кластера, аналогично $\mathrm{b_i}$для соседнего кластера (соседним считается кластер с наименьшей непохожестью). Далее высчитывается силуэт:

$$\mathrm{s(i)} = \begin{cases} 1 - \dfrac{\mathrm{a(i)}}{\mathrm{b(i)}}, & \text{если } \mathrm{a(i)} < b(i) \\ 0, & \text{если } \mathrm{a(i)} = \mathrm{b(i)} \\ \dfrac{\mathrm{a(i)}}{\mathrm{b(i)}}, & \text{если } \mathrm{a(i)} > b(i) \end{cases} \quad (27)$$

Таким образом, будет получено необходимое разбиение на классы, благодаря которому можно подобрать факторы для каждого пользователя. Данная работа описывает методы, которые могут применяться и на других массивах и типах данных.

СПИСОК ЛИТЕРАТУРЫ

1. *Синева И.С.* Будущий интернет: сетевой подход // Фундаментальные проблемы радиоэлектронного приборостроения. – 2010. – Т. 10. – № 1-3. – С. 237-240.
2. *Gorbunov J.A., Krotov L.N., Krotova E.L.* Legitimate User Profiling Based on the Third Order Spline Approximation of the Initial Data Sequence // World Applied Sciences Journal 29. – 2014. – №12. – pp. 1605-1610.
3. *Chen T., Guestrin C.* XGBoost: A Scalable Tree Boosting System // KDD'16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – New York.: ACM, 2016. – pp. 785-794.
4. Past winners of the John M. Chambers Statistical Software Award. ASA Statistics Computing and Graphics [Электронный ресурс]. – Режим доступа: http://stat-computing.org/awards/jmc/winners.html. (01.10.2016)
5. *Friedman J.* Greedy Function Approximation: A Gradient Boosting Machine // Annals of Statistics 29. – 2001. – № 2. – pp. 1189-1232.
6. *Koyutürk M., Grama A.* PROXIMUS: a framework for analyzing very high dimensional discrete-attributed datasets // KDD'03 Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – New York.: ACM, 2013. – pp. 147-156.
7. *Денисов В.Ю., Синева И.С.* Методы многомерного data mining для анализа больших данных // Труды Северо-Кавказского филиала Московского технического университета связи и информатики. – Ростов-на-Дону.: ПЦ "Университет" СКФ МТУСИ, 2016. – С. 98-102.