СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
SIBERIAN FEDERAL UNIVERSITY

1945 2015 ПОБЕДА! 70 ЛЕТ

Краевой фонд науки
Основан в 1992 году

ПРОСПЕКТ СВОБОДНЫЙ-2015

МЕЖДУНАРОДНАЯ КОНФЕРЕНЦИЯ СТУДЕНТОВ,
АСПИРАНТОВ И МОЛОДЫХ УЧЕНЫХ

ЭЛЕКТРОННЫЙ СБОРНИК МАТЕРИАЛОВ МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ СТУДЕНТОВ, АСПИРАНТОВ И МОЛОДЫХ УЧЕНЫХ **«ПРОСПЕКТ СВОБОДНЫЙ-2015»**, ПОСВЯЩЕННОЙ 70-ЛЕТИЮ ВЕЛИКОЙ ПОБЕДЫ

КРАСНОЯРСК, СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
15-25 АПРЕЛЯ 2015 Г.

Министерство образования и науки Российской федерации
ФГАОУ ВПО «Сибирский федеральный университет»

Сборник материалов Международной конференции студентов, аспирантов и молодых ученых «Проспект Свободный-2015», посвященной 70-летию Великой Победы

Красноярск, Сибирский федеральный университет, 15-25 апреля 2015 г.

Красноярск, 2015.

«Математика, информатика:
Теория вероятностей, математическая
статистика и финансово-актуарная математика»

УДК 519.213

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РАСЧЕТА ХАРАКТЕРИСТИК СЛУЧАЙНОГО МНОЖЕСТВА СОБЫТИЙ

Атабеков М. А.
научный руководитель канд. физ.-мат. наук Семенова Д. В.
Сибирский федеральный университет

Пусть (Ω,F,**P**)- вероятностное пространство. Зафиксируем некоторый конечный набор событий $\boldsymbol{U} \subset F$.

Случайное множество событий K на конечном множестве событий $\boldsymbol{U} \subset F$ определяется как отображение K: $\Omega \to 2^U$, измеряемое относительно пары алгебр (F,2^{2^U}) в том смысле, что для всякого $\boldsymbol{U} \in 2^{2^U}$ существует прообраз $K^{-1}(\boldsymbol{U}) \in F$,

такой что $\mathbf{P}(\boldsymbol{U}) = \mathbf{P}(K^{-1}(\boldsymbol{U}))$.

Выражение $K^{-1}(\omega) \in \{u \in U : \omega \in u\}$ может быть истолковано, как "случайное множество наступивших событий", поскольку элементарному исходу эксперимента $\omega \in \Omega$ ставится в соответствие некоторое подмножество событий $X \subseteq \boldsymbol{U}$, которое содержит все те события, которые наступили в данном испытании.

Случайное множество событий K, заданное на конечном множестве событий $\boldsymbol{U}$, определяется своим дискретным вероятностным распределением. Если мощность рассматриваемого множества событий $|\boldsymbol{U}| = N < \infty$, то имеется 2^N видов вероятностных зависимостей между событиями этого множества, т.е. ровно столько, сколько у этого множества подмножеств. Дискретное вероятностное распределение (далее просто вероятностное распределение) случайного множества событий K, заданного на конечном множестве избранных событий $\boldsymbol{U} \subset F$ - это набор 2^N значений вероятностной меры **P** на событиях из $2^{\boldsymbol{U}}$. Как известно, такое распределение можно задать шестью эквивалентными способами по формулам Мёбиуса. Рассмотрим для примера два из них.

Вероятностное распределение I-го рода случайного множества событий K на $\boldsymbol{U}$ - это набор $\{p(X), X \subseteq \boldsymbol{U}\}$ из 2^N вероятностей вида $p(X) = \mathbf{P}(K = X) = \mathbf{P}\left(\left(\bigcap_{x \in X} x\right) \bigcap \left(\bigcap_{x \in X^c} x^c\right)\right)$, где $X^c = \boldsymbol{U} \setminus X$, $x^c = \Omega \setminus x$.

Вероятностное распределение I-го рода всегда легитимно, т.е. обладает свойствами $0 \le p(X) \le 1, X \subseteq \boldsymbol{U}$, $\sum_{X \subseteq U} p(X) = 1$.

Вероятностное распределение II-го рода случайного множества событий K на $\boldsymbol{U}$ - набор $\{pX, X \subseteq \boldsymbol{U}\}$ из 2^N вероятностей вида $pX = \mathbf{P}(X \subseteq K) = \mathbf{P}\left(\left(\bigcap_{x \in X} x\right)\right)$, $X \subseteq \boldsymbol{U}$

Вероятностное распределение II-го рода $\{pX, X \subseteq \boldsymbol{U}\}$ случайного множества событий K на $\boldsymbol{U}$ удовлетворяет системе из 2^N неравенства Фреше-Хёвинга: $0 \le p_x^- \le p_x \le p_x^+ \le 1$, где $p_x^- = \max\left\{0, 1 - \sum_{x \in X}(1 - p_x)\right\}$ нижняя граница Фреше-Хёвинга, $p_x^+ = \min_{x \in X} p_x$ - верхняя граница Фреше-Хёвинга.

Вероятность распределения I-го и II-го рода связаны взаимно-обратными формулами обращения Мёбиуса: $p_x = \sum_{Y \in 2^U; X \subseteq Y} p(Y)$, $\mathrm{p(X)} = \sum_{Y \in 2^U; X \subseteq Y} (-1)^{|Y|-|X|} p_Y$, для всех

$X \in 2^U$. Также существуют формулы обращения Мёбиуса, связывающие вероятностные распределения I-го и III-VI-рода.

В общем случае количество параметров, задающих вероятностные распределения случайного множества событий, зависит от мощности $\boldsymbol{U}$, поскольку каждое множество из N событий характеризуется набором из 2^N вероятностей.

Программный комплекс Random Set of Events (RSE) разработана в среде программирования Pascal на языке программирования Delphi версии XE3 и предлагает решение следующих задач:

- моделирование значений случайного множества событий булев;
- генерирование/ввод вероятностного распределения I-го рода и расчет остальных родов вероятностных распределений по формулам обращения Мёбиуса.

В качестве входных данных пользователем задаются:

- мощность избранных событий;
- вероятностное распределение I-го рода событий.

Основное практическое предназначение работы RSE - вспомогательное программное средство для специалистов в области эвентологии.

Может быть использована в научной и образовательной деятельности.

Список литературы

1. Воробьев О.Ю., Воробьев А.О. Суммирование сет-аддитивных функций и формула обращения Мёбиуса. С. 417-420.

2. Воробьев О.Ю. Эвентология. Красноярск: Сибирский федеральный университет, 2007. 435 с.

3. Семенова Д.В., Кочанова Ю.С. Методы оценки эвентологических распределений. Кемерово: Практика, 2012. С. 119-122.

4. Прохоров Ю.В. Вероятность и математическая статистика: Энциклопедия. М.: Больш. Рос. Энцикл., 1999. 910 с.

5. Воробьев О.Ю Фреше-граничные эвентологические распределения и их применение. Красноярск: Крас. Гос. Торг.-эконом. ин-т, Сиб.фед. ун-т, 2010. С. 57-69.

УДК 519.2.

РАСЧЕТ СПРАВЕДЛИВОЙ ЦЕНЫ ДЛЯ МОДЕЛИ (B,S)-РЫНКА С ДИВИДЕНДАМИ

Галицына Е.Н.
научный руководитель: доцентДанилова Н.В.
Институт математики, механики и компьютерных наук им.И.И.Воровича

В работе приводится алгоритм расчета справедливой цены для Европейского Call-опциона с финансовым обязательством:

$$f_N = (S_N - K)^+ = \begin{cases} S_N - K, если S_N > K \\ 0, \quad если S_N \le K \end{cases}.$$

В результате находится капитал портфеля с дивидендами. Расчёты задаются с помощью рекуррентных формул в непрерывном и дискретном времени.

Рассмотрим портфель с дивидендами в случае дискретного времени:

$$\Delta\left(\frac{X_n}{B_n}\right) = \gamma_n\left(\Delta\left(\frac{S_n}{B_n}\right) + \frac{\Delta D_n}{B_n}\right), \quad n = 1,2,\ldots,N.$$

$$\Delta D_n = cS_{n-1}, \quad c \equiv const$$

Модель (B,S) – рынка в случае дискретного времени выглядит следующим образом:

$$\begin{cases} S_n = S_{n-1}(1 + \mu_n + \delta_n\varepsilon_n) \\ B_n = B_{n-1}(1 + r_n) \end{cases}$$

Процесс $(S_n)_{n=0}^N$ описывает стоимость акции, процесс $(B_n)_{n=0}^N$ описывает величину банковского счёта. Значения S_0, B_0 известны. Источником случайности является последовательность независимых случайных величин $(\varepsilon_n)_{n=0}^N$ таких, что $\varepsilon_n \epsilon \{-1,1\}$.Рассматривается естественная фильтрация $F_0 = \{\emptyset, \Omega\}, F_n = \sigma(\varepsilon_1, \ldots, \varepsilon_n)$. Параметры модели: r_n- процентная ставка, $\delta_n > 0$ волатильность,μ_n – снос.

Мартингальная мера для портфеля с дивидендами имеет следующий вид:

$$p(\varepsilon_n = 1) = p = \frac{r_n - \mu_n - c(1+\mu_n)}{2\delta_n(1+c)} + \frac{1}{2}$$

$$p(\varepsilon_n = -1) = 1 - p = q = \frac{1}{2} - \frac{r_n - \mu_n - c(1+\mu_n)}{2\delta_n(1+c)}$$

Ограничения на параметры:

$$\max(-\delta_n + \frac{r_n - c}{1+c}, \delta_n - 1) < \mu_n < \delta_n + \frac{r_n - c}{1+c}$$

p –такая мера, относительно которой процесс $\left(\frac{X_n}{B_n}\right)_{n=1}^N$ является мартингалом.

Задача для расчёта справедливой цены X_0 имеет следующий вид:

$$\begin{cases} \min\limits_{\gamma} X_0 \\ \Delta\left(\frac{X_n}{B_n}\right) = \gamma_n\left(\Delta\left(\frac{X_n}{B_n}\right) + c\frac{S_n}{B_n}\right) \\ X_N \ge f_N \end{cases}$$

Формула для решения:

$$X_N = f_N = g_{n-1}(S_n)$$

$$X_{n-1} = g_{n-1}(S_{n-1}) = \frac{1}{1+r_n}\{g_n(S_{n-1}(1+\mu_n+\delta_n))p + g_n(S_{n-1}(1+\mu_n-\delta_n))q\}$$

Аналогично рассмотрим модель (B,S) – рынка в случае непрерывного времени:

$$\begin{cases} dS_t = S_t(\tilde{\mu}dt + \tilde{\delta}dW_t) \\ dB_t = B_t(\tilde{r}dt) \end{cases} t \in [0T]$$

Балансовое приращение имеет вид:

$$\begin{cases} \min\limits_{\gamma} X_0 \\ d\left(\frac{X_t}{B_t}\right) = \gamma_t\left(d\left(\frac{X_t}{B_t}\right) + \tilde{c}\frac{S_t}{B_t}dt\right) \\ X_T \geq f_T \end{cases}$$

Зависимость параметров дискретного и непрерывного времени выглядит следующим образом : $\mu = \tilde{\mu}h, r = \tilde{r}h, c = \tilde{c}h, \ \delta = \tilde{\delta}\sqrt{h}$

Где $h = \frac{T}{N}$.

Формула для решения задачи имеет следующий вид:

$$X_0 = S_0 exp(-\tilde{\delta}T)\Phi\left(\frac{\ln\left(\frac{S_0}{K}\right) + T\left(r - \tilde{c} + \frac{\tilde{\delta}^2}{2}\right)}{\tilde{\delta}\sqrt{T}}\right) - K\exp(-\tilde{r}T)\Phi\left(\frac{\ln\left(\frac{S_0}{K}\right) + T\left(r - \tilde{c} - \frac{\tilde{\delta}^2}{2}\right)}{\tilde{\delta}\sqrt{T}}\right)$$

$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int\limits_{-\infty}^{x} e^{-\frac{t^2}{2}}dt$ - это функция Лапласа.

Пусть $M_1 < M_2$. Введем два барьера M_1, M_2 такие, что: $\begin{matrix} \overline{M_1}(ih) = M_1 exp(dih) \\ \overline{M_2}(ih) = M_2 exp(dih) \end{matrix}$

Тогда параметры модели вычисляются следующим образом:

$$r_i = \hat{r}_1 I_{\{S_{i-1} < \overline{M_1}((i-1)h)\}} + \hat{r}_2 I_{\{\overline{M_1}((i-1)h) < S_{i-1} < \overline{M_2}((i-1)h)\}} + \hat{r}_3 I_{\{S_{i-1} \geq \overline{M_2}((i-1)h)\}}$$

$$\mu_i = \hat{\mu}_1 I_{\{S_{i-1} < \overline{M_1}((i-1)h)\}} + \hat{\mu}_2 I_{\{\overline{M_1}((i-1)h) < S_{i-1} < \overline{M_2}((i-1)h)\}} + \hat{\mu}_3 I_{\{S_{i-1} \geq \overline{M_2}((i-1)h)\}}$$

$$\delta_i = \hat{\delta}_1 I_{\{S_{i-1} < \overline{M_1}((i-1)h)\}} + \hat{\delta}_2 I_{\{\overline{M_1}((i-1)h) < S_{i-1} < \overline{M_2}((i-1)h)\}} + \hat{\delta}_3 I_{\{S_{i-1} \geq \overline{M_2}((i-1)h)\}}$$

Пусть задано следующее разбиение: $0=T_0<T_1<...<T_N=T$.

Формула для решения имеет следующий вид :

$$V(x,t) = exp(-\tilde{r}(T-t)) \times$$

$$\times Ef\left(xexp\left(\left(\tilde{r} - \frac{\tilde{\sigma}^2}{2}\right)(T-t) - \sum_{i=1}^{N}\frac{g(S_{T_{i-1}})}{S_{T_{i-1}}}(T_i - T_{i-1}) + \tilde{\sigma}sqrt(T-t)\xi\right)\right)$$

где $x = S_0, t = 0, g(S_{i-1}) = \Delta x^{\alpha}, \alpha \in [0,1], \xi \sim (0,1)$.

Рассмотрим реализацию метода вычисление справедливой цены европейского опциона-колл для модели с дивидендами.

Для исходных данных T := 1; K := 6; S_0 := 6; delta := 0.1e-1; r := 0.5; N := 10

c	X_0^d	X_0^c
0	0.391	0.391
0.005	0.376	0.374
0.01	0.362	0.361
0.015	0.347	0.345
0.02	0.332	0.331
0.025	0.318	0.319
0.03	0.304	0.302
0.035	0.291	0.29
0.04	0.276	0.275

Таблица 1. Зависимость справедливой цены от параметра c

Поведение справедливой цены убывает с ростом c - это вполне реалистично, так как от дивидендов мы получаем прибыль, поэтому можем позволить себе иметь меньшую справедливую цену. Если портфель самофинансируемый, то есть дивидендов нет, то мы от них прибыли не получаем, следовательно нам необходимо платить большую справедливую цену. В таблице 1. представлена зависимость справедливой цены от c, хорошо видно, что при $c = 0$ у нас самая большая справедливая цена, которая совпадает со справедливой ценой самофинансируемого портфеля. При увеличении c видно, что справедливая цена уменьшается. В таблице 1 получены данные при расчете справедливой цены в непрерывном(X_0^c) и дискретном времени(X_0^d).

Рассмотрим пример с участием двух барьеров. Пусть $r_1 = 0.1, \sigma_1 = 0.001, r_2 = 0.11, \Delta := 1, \sigma_2 = 0.002, r_3 = 0.13, \sigma_3 = 0.003 N = 5, S_0 = 6, T = 1, t = 0$

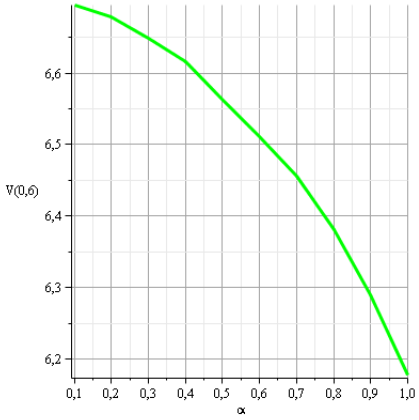


Рис.1. График зависимости справедливой цены от α.

При значении $\alpha = 0$ значение справедливой цены, совпадает со значением справедливой цены вычисленной в дискретном времени при одинаковых параметрах.

Список литературы

1.А.В.Мельников "Финансовые рынки: стохастический анализ ирасчёт производных ценных бумаг." М.:ТВП, 1997.

2.А.Н.Ширяев."Основы стохастической финансовой математики". Том1.Факты модели. М.:Фазис, 1998.

3.Г.И.Белявский,Н.В.Данилова "Диффузионные модели со случайным переключением параметров. Расчеты и финансовые приложения. "LambertAcademicPublishing" 2002

УДК 519.7

АНАЛИЗ ONLINE АЛГОРИТМОВ ДЛЯ ЗАДАЧИ ПОИСКА ЦЕЛИ НА ПРЯМОЙ

Гончарик К. В.,
научный руководитель д-р физ.-мат. наук Быкова В. В.
Сибирский федеральный университет

Под online алгоритмом принято понимать алгоритм, обрабатывающий вход по мере поступления [2]. Считается, что для offline алгоритма все исходные данные доступны с самого начала и требуется вывести ответ. Поскольку online алгоритм не знает весь вход, то он вынужден принимать решение, которое позже может оказаться не оптимальным. По существу online алгоритм стремится сформировать оптимальное решение в условиях некоторой неопределенности, не будучи в состоянии видеть будущее, то есть каким в итоге будет обрабатываемый объект. Конечно, online алгоритм может найти решение гораздо быстрее, чем соответствующий offline алгоритм (ведь длина входа меньше), но это решение может оказаться качественно хуже (поскольку online алгоритм располагает только частью потенциального входа). Качество работы (или эффективность) online алгоритмов принято оценивать с помощью конкурентного анализа [1].

В общем случае online алгоритм принимает в качестве входа последовательность исходных данных, разбитую на части (эти части часто называют запросами):

$$\sigma = \sigma(1),\ldots,\sigma(n). \qquad (1)$$

Части входа online алгоритм должен обработать в порядке поступления. При получении части $\sigma(t)$ online алгоритм не знает $\sigma(t')$ для $t' > t$. Размеры частей, как правило, также неизвестны. В конкурентном анализе всякий конкретный online алгоритм A сравнивается с соответствующим ему offline алгоритмом B, который сразу знает последовательность (1), может обработать ее с минимальными затратами и выдать решение требуемого качества.

Пусть задана некоторая последовательность σ вида (1). Обозначим через $\mathrm{cost}_A(\sigma)$ и $\mathrm{cost}_B(\sigma)$ расходы, понесенные алгоритмами A и B соответственно для обработки последовательности σ. Online алгоритм A называется c-конкурентным, если для любой последовательности σ, состоящей из некоторых частей исходных данных, и любого offline алгоритма B (в том числе и оптимального), выполняется отношение

$$\mathrm{cost}_A(\sigma) \le c \cdot \mathrm{cost}_B(\sigma) + c_1, \qquad (2)$$

где константы c, c_1 независимы от σ. Принято значения этих констант вычислять применительно к худшему случаю (для наиболее затратной последовательности σ). Считается, что чем меньше значение константы c, тем более конкурентным является online-алгоритм A по сравнению с B.

При выполнении конкурентного анализа online алгоритмов приходится сталкиваться с некоторыми проблемами. Прежде всего, как выбрать алгоритм B, и как ввести функцию $\mathrm{cost}(\sigma)$ стоимости алгоритмов. Конечно, желательно для рассматриваемой задачи располагать оптимальным алгоритмом (в смысле введенной функции стоимости $\mathrm{cost}(\sigma)$), и использовать его в (2) в качестве B. Однако это не для всякой задачи удается сделать. Тогда в роли B используют любой известный offline алгоритм. Задание функции $\mathrm{cost}(\sigma)$ также неоднозначно, ведь $\mathrm{cost}(\sigma)$ может отражать как качество получаемого решения, так и затраты на его получения. Общих правил выбора B и $\mathrm{cost}(\sigma)$ конкурентный анализ не дает. Для каждой конкретной задачи, допускающей offline и online формулировки, необходимо это осуществлять с учетом особенностей решаемой задачи.

Одной из задач, для которой возможны offline и online формулировки, является задача поиска цели на прямой. Особенностью данной задачи является то, что в явном виде в ней отсутствует последовательность (1). Тем не менее, online подход к ее решению предполагает выбор оптимальной стратегии действий в условиях неполноты исходных данных. Можно предложить следующую интерпретацию этой задачи. Рассеянный профессор, выйдя из университета, понял, что не помнит, где припарковал свой автомобиль. Парковка расположена вдоль здания университета по прямой, как в правую, так и в левую сторону от входа. Что же делать профессору? Необходим алгоритм, который укажет профессору последовательность шагов для поиска своего автомобиля. Для offline алгоритма расположение автомобиля известно, поэтому следует просто идти в нужном направлении до цели. Очевидно, что это оптимальная стратегия достижения цели в данных условиях. Для online алгоритма расположение автомобиля неизвестно.

Формально поиск цели на прямой можно выразить так. Пусть задано пространство поиска R в виде бесконечной в обе стороны прямой. Пусть стартовая точка поиска $s = 0$, то есть совпадает с началом координат. Обозначим через t координату расположения цели поиска. Предположим, что движение осуществимо только по целочисленным координатам прямой (ведь профессор движется дискретно – шагами). При этом координата цели t может быть задана вещественным числом. Тогда для offline и online алгоритмов имеем.

Вход offline алгоритма *B*: $s = 0$, значение t в R (может быть положительным, отрицательным и нулем).

Выход: t шагов в нужном направлении, а именно при $t < 0$ сделать t шагов влево от начала координат, при $t > 0$ сделать t шагов вправо от начала координат, а при $t = 0$ остаться на месте.

Вход online алгоритма *A*: R и $s = 0$; значение t неизвестно.

Выход: описание самого пути (последовательность целочисленных координат точек на R).

Для данной задачи всякий online алгоритм A – некоторая стратегия порождения последовательности шагов движения по прямой, начиная с $s = 0$, когда место расположения цели неизвестно (в условиях неполноты исходных данных). Стратегия должно быть такой, чтобы после каждого шага информация о месте расположения цели пополнялась. В качестве функции стоимости здесь целесообразно выбрать число ON шагов, которые потребовались для достижения цели алгоритмом A. Заметим, что эта величина зависит от t. Обозначим через OPT число шагов, необходимых алгоритму B для достижения цели. Здесь $OPT = t$.

Пусть online алгоритм A реализует следующую последовательность шагов поиска:

$(0, +1, -2, +4, ..., +2^k, -2^{k+1}, \ldots)$.

Тогда наихудшее расположение клада такое: $t = -(2^{k+1} + \varepsilon)$. Это отвечает следующей стоимости поиска:

$$ON = 2(1 + 2 + \ldots + 2^{k+2}) + 2^{k+1} + \varepsilon = 2 \cdot (2^{k+3} - 1) + 2^{k+1} + \varepsilon = 9 \cdot 2^{k+1} + \varepsilon - 2.$$

Поскольку $OPT = 2^{k+1} + \varepsilon$, то $ON / OPT < 9$. Следовательно, online алгоритм A является 9-конкурентным по отношению к оптимальному offline алгоритму B.

В работе доказано, что всякий детерминированный online алгоритм, реализующий стратегию движения по увеличивающейся амплитуде поочередно влево и вправо в виде последовательности шагов:

$(0, +x_0, -x_1, +x_2, -x_3, \ldots)$, (3)

$0 < x_0 < x_2 < x_4 < \ldots, 0 < x_1 < x_3 < x_5 < \ldots$ (4)

всегда является 9-конкурентным. Это означает, что online алгоритму для достижения цели в худшем случае (при самом неблагоприятном расположении цели) понадобиться в 9 раз больше усилий, чем offline алгоритму, который знает место расположения цели и действует оптимально.

Была написана программа имитирующая стратегию (3)–(4). Вычислительные эксперименты, проведенные с помощью этой программы, подтвердили 9-конкурентность стратегии (3)–(4). Результаты экспериментов представлены в таблице.

t	$OPT = t$	ON	c
1	1	1	1
2	2	6	3
5	5	25	5
8	8	28	3,5
11	11	31	2,818181818
14	14	98	7
18	18	102	5,666666667
22	22	106	4,818181818
27	27	111	4,111111111
31	31	115	3,709677419
36	36	120	3,333333333
41	41	125	3,048780488
46	46	386	8,391304348
172	172	1536	8,930232558
688	688	6148	8,936046512

Рассмотренная задача поиска цели на прямой в режиме online изучается в робототехнике, где исследуются различные обобщения этой задачи [3, 4]. К ним относятся: поиск цели на $m \geq 2$ прямых плоскости; поиск цели в неизвестном регионе плоскости (t известно, но нет информации о препятствиях); поиск цели на плоскости в многоугольнике видимости и др. В этих постановках обобщается понятие пространства поиска и модель видимости цели. В рассмотренной задаче поиска предполагается, что цель обнаруживается только при непосредственном ее достижении, то есть когда расстояние до цели менее одного шага. Это самая примитивная дискретная модель видимости цели. Для более сложных моделей пространства поиска в настоящее время разрабатываются детерминированные и вероятностные online алгоритмы поиска цели. Для оценки качества их работы привлекаются инструменты теории сложности вычислений, конкурентного анализа и имитационного моделирования.

Список литературы

1. Borodin A., El-Yaniv R. Online Computation and Competitive Analysis. Cambridge University Press, 1998.

2. Fiat A., Woeginger G. Online Algorithms: The State of the Art. Springer: LNCS 1442, 1998.

3. Ghosh S. K. Visibility Algorithms in the Plane. Cambridge University Press. United Kingdom, 2007.

4. Latombe J. C. Robot Motion Planning. Kluwer Academic Publishers Boston. MA, 1991.

УДК 004.9

РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ СОЗДАНИЯ ЭЛЕКТРОННЫХ ГЛОССАРИЕВ

Горохова Е. А.,
научный руководитель кандидат физ.-мат. наук Баранова И. В.
Сибирский федеральный университет
Институт математики и фундаментальной информатики

Введение

В современном обществе высоких технологий и всеобщего использования компьютера электронные ресурсы и справочники постепенно заменяют обычные книги, делая процесс обучения интереснее, легче и нагляднее.

Целью данной работы является разработка инструмента для создания электронных глоссариев. Также в работе проведена апробация работы инструмента на практике и с его помощью создан глоссарий по теории графов.

Создание электронного глоссария

Электронный глоссарий по теории графов предназначен для студентов, изучающих дисциплины "Дискретная математика", "Комбинаторные алгоритмы" и другие, родственные с ними дисциплины.

Согласно современным требованиям к электронным глоссариям, создаваемый электронный справочник должен содержать набор терминов, для каждого из которых должны быть приведены определения на русском и английском языках, а также указан источник, из которого взято данное определение.

К глоссарию выдвинуты следующие требования:

- глоссарий должен демонстрироваться в сети Интернет, то есть представлять собой электронный ресурс, реализованный на языке html,
- в нем должен быть предусмотрен выбор терминов на любую букву, указанную пользователем, или полный список терминов,
- глоссарий должен иметь возможность встраивания в электронный учебник,
- должна быть предусмотрена возможность импорта глоссария в глоссарии электронных ресурсов на базе платформы Moodle.

Поэтому для удобства представления данных на сайте и реализации возможностей импорта, указанных выше, для хранения информации было решено использовать текстовые данные, отформатированные с помощью язык разметки XML. [1], [2].

Язык XML решает задачу хранения и передачи данных. XML может использоваться в любых приложениях, которым нужна структурированная информация. При внимательном взгляде на окружающий нас информационный мир можно выделить множество задач, связанных с созданием и обработкой структурированной информации, для решения которых может использоваться XML:

- XML-документы выполняют роль универсального формата для обмена информацией между отдельными компонентами большой программы.
- Язык XML позволяет описывать данные произвольного типа и используется для представления специализированной информации, например химических, математических, физических формул, медицинских рецептов, нотных записей и других.
- Информация, содержащаяся в XML-документах, может изменяться, передаваться на машину клиента и обновляться по частям.

- XML может использоваться в обычных приложениях для хранения и обработки структурированных данных в едином формате.

Кроме того, этот язык очень удобен для хранения и изменения записей. Также XML-файл имеет небольшой вес, что очень важно при конструировании web-приложений.

Опишем структуру XML-документа, выполняющего роль простейшей текстовой базы данных, предназначенной для хранения данных глоссария.

Структура XML-документа

Корневой элемент XML-файла называется <dataroot>. Этот элемент представляет собой список всех терминов. Каждый термин(каждая запись нашей XML-базы данных) называется <items>. Каждая запись имеет поля: <term>, <englishterm> , <determination> , <english>, <literature> , <englishliterature>. Расшифруем назначение этих полей:

- <term> - сам термин (на русском языке),
- <englishterm> - термин на английском языке,
- <determination> - определение на русском языке
- <english> - определение на английском языке
- <literature> - источник литературы на русском языке,
- <englishliterature> - источник литературы на английском языке.

Реализация

Предложенный в данной работе инструмент создания электронных глоссариев реализован на языке PHP[3], поскольку данный язык представляет собой удобную среду для написания web-приложений, к которым относятся сайты и электронные ресурсы[4] (в том числе и рассматриваемые электронные глоссарии).

Инструмент представляет собой программный комплекс, включающий в себя два PHP-файла. Первый из них выполняет следующие функции:

- считывает информацию из XML-файла,
- размещает информацию на странице,
- реализует удобный переход по буквам русского алфавита и упрощающий поиск нужного термина.

Согласно выдвинутым требованиям программное приложение создает каждую страницу глоссария в виде web-страницы, написанной на языке html.

Второй PHP-файл выполняет считывание и отображение списка источников, из которых взяты термины. Также приведен список литературы, из которого взяты определения для глоссария.

Электронный глоссарий по теории графов

В работе проведена апробация инструмента на практике и с его помощью создан электронный глоссарий по теории графов. В базу данных терминов по теории графов было внесено около 900 терминов на русском языке с их определениями на русском и английском языке. Также был создан список использованных источников, в котором находится 31 источник.

Для набора математических формул в глоссарии был использован javascript jsMath. Для набора формул используется язык TeX, после чего jsMath сам преобразовывает его в привычные для нас математические формулы.

Внешний вид созданного электронного глоссария

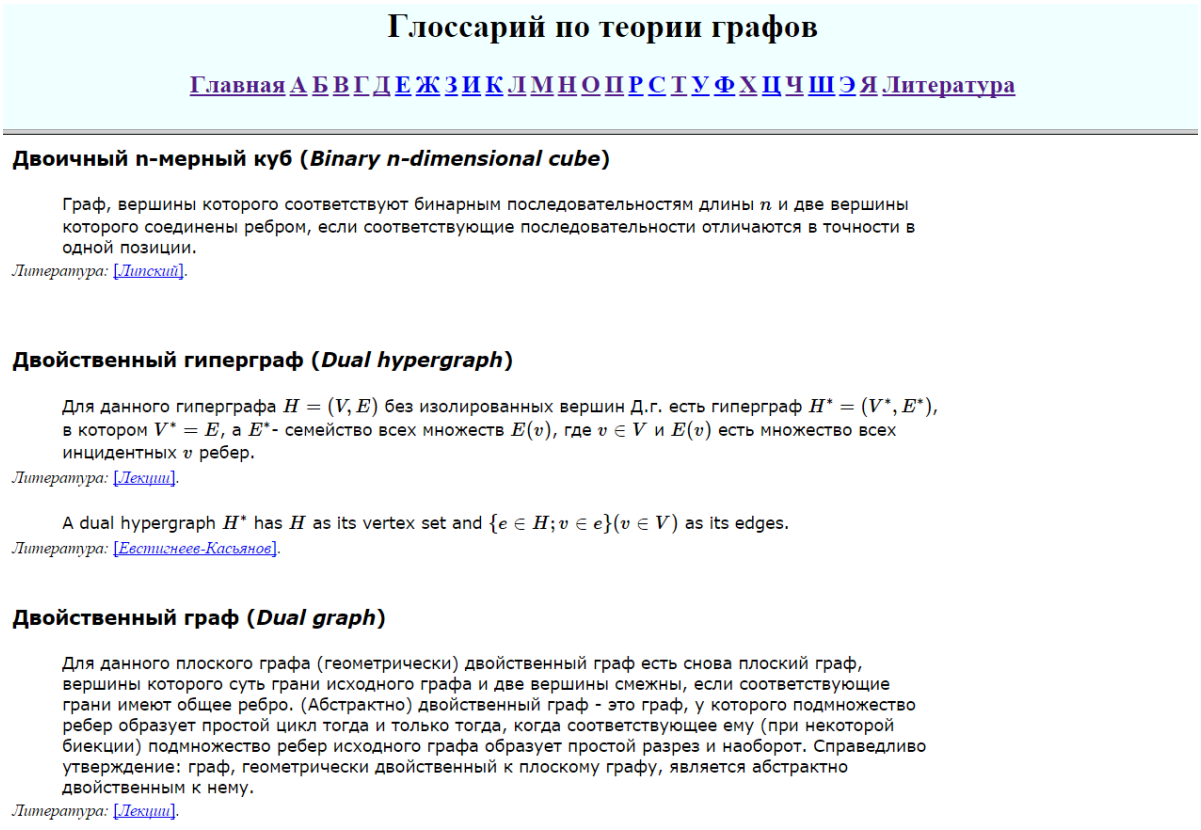
Как было сказано выше, электронный глоссарий представляет собой программно-формируемый электронный ресурс. Каждая страница глоссария является web-страницей, написанной на языке html.

В качестве макета страниц был выбран одноколоночный макет с плавающей шириной и с шапкой. В шапке находится название глоссария, под которой располагается навигационное меню по страницам глоссария. Ниже находится основная часть глоссария, в которой отображаются все термины с соответствующими определениями. Все страницы сайта оформлены в едином стиле и цветовой гамме.

Всего в составе глоссария может присутствовать 35 страниц: главная страница, 33 страницы, озаглавленные буквами русского алфавита и литература.

На главной странице отображаются абсолютно все термины, находящиеся в базе. Это может быть полезно при первичном ознакомлении с содержимым глоссария. На рис. 1 показан внешний вид главной страницы глоссария.

В глоссарии предусмотрена такая важная функция, как возможность перехода по первым буквам русского алфавита, что значительно упрощает и ускоряет поиск нужного термина.



Глоссарий по теории графов

Главная А Б В Г Д Е Ж З И К Л М Н О П Р С Т У Ф Х Ц Ч Ш Э Я Литература

Двоичный n-мерный куб (*Binary n-dimensional cube*)

Граф, вершины которого соответствуют бинарным последовательностям длины n и две вершины которого соединены ребром, если соответствующие последовательности отличаются в точности в одной позиции.

Литература: [Литский].

Двойственный гиперграф (*Dual hypergraph*)

Для данного гиперграфа $H = (V, E)$ без изолированных вершин Д.г. есть гиперграф $H^* = (V^*, E^*)$, в котором $V^* = E$, а E^*- семейство всех множеств $E(v)$, где $v \in V$ и $E(v)$ есть множество всех инцидентных v ребер.

Литература: [Лекции].

A dual hypergraph H^* has H as its vertex set and $\{e \in H; v \in e\}(v \in V)$ as its edges.

Литература: [Евстигнеев-Касьянов].

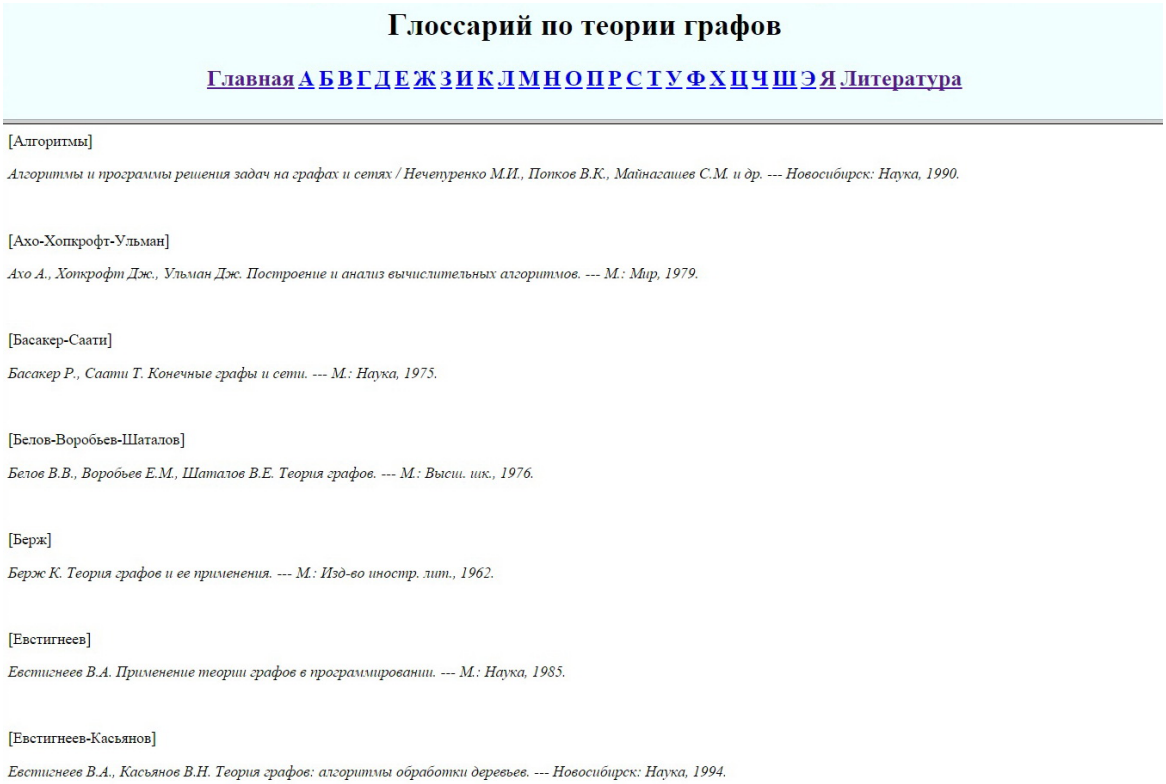
Двойственный граф (*Dual graph*)

Для данного плоского графа (геометрически) двойственный граф есть снова плоский граф, вершины которого суть грани исходного графа и две вершины смежны, если соответствующие грани имеют общее ребро. (Абстрактно) двойственный граф - это граф, у которого подмножество ребер образует простой цикл тогда и только тогда, когда соответствующее ему (при некоторой биекции) подмножество ребер исходного графа образует простой разрез и наоборот. Справедливо утверждение: граф, геометрически двойственный к плоскому графу, является абстрактно двойственным к нему.

Литература: [Лекции].

Рисунок 1. Вывод терминов, начинающихся на букву "Д"

Также в электронном глоссарии есть страница «Литература», отображающая список использованных источников. На эту страницу пользователь может попасть, непосредственно нажав на ссылку «Литература», либо нажав на одну из ссылок, находящихся под каждым термином и указывающих на источник взятой информации. На рис. 2 демонстрируется внешний вид страницы «Литература».



Глоссарий по теории графов

Главная А Б В Г Д Е Ж З И К Л М Н О П Р С Т У Ф Х Ц Ч Ш Э Я Литература

[Алгоритмы]

Алгоритмы и программы решения задач на графах и сетях / Нечепуренко М.И., Попков В.К., Майнагашев С.М. и др. --- Новосибирск: Наука, 1990.

[Ахо-Хопкрофт-Ульман]

Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. --- М.: Мир, 1979.

[Басакер-Саати]

Басакер Р., Саати Т. Конечные графы и сети. --- М.: Наука, 1975.

[Белов-Воробьев-Шаталов]

Белов В.В., Воробьев Е.М., Шаталов В.Е. Теория графов. --- М.: Высш. шк., 1976.

[Берж]

Берж К. Теория графов и ее применения. --- М.: Изд-во иностр. лит., 1962.

[Евстигнеев]

Евстигнеев В.А. Применение теории графов в программировании. --- М.: Наука, 1985.

[Евстигнеев-Касьянов]

Евстигнеев В.А., Касьянов В.Н. Теория графов: алгоритмы обработки деревьев. --- Новосибирск: Наука, 1994.

Рисунок 2. Страница "Литература"

Заключение

В работе была реализована разработка инструмента для создания электронных глоссариев. Также в работе проведена апробация работы инструмента на практике и с его помощью создан глоссарий по теории графов.

В дальнейшем планируется добавить в программный комплекс еще один файл, предоставляющий возможность пользователю самому добавлять термины в глоссарий.

Список литературы

1. Баранова, И.В. Технологии создания Internet-баз данных и программирования web-приложений: учеб. пособие / И. В. Баранова. – Красноярск: ИПК СФУ, 2009. – 116 с.

2. Штайнер Г. HTML / XML / CSS / Г.Штайнер. – М.: Бином, 2005. – 510 с.

3. Т.Р. Нието, П.Дж. Дейтел, Х.М. Дейтел. Как программировать для Internet и WWW. – Москва: Бином, 2007. – 944 с.

4. Дэн Уитворт, Томас Пауэл. HTML: Справочник программиста. Москва: Бином, 2001. – 384 с.

УДК 519.23

МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ ИДЕНТИФИКАЦИИ АВТОРСКОГО ТЕКСТА

Додонова М.М.,
научный руководитель: канд. физ.-мат. наук, доцент Баранова И.В.
Сибирский Федеральный Университет,
Институт математики и фундаментальной информатики

Введение

Кластерный анализ [1], [2] – задача разбиения заданной выборки объектов (ситуаций) на подмножества так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Задача кластеризации относится к статистической обработке, а также к широкому классу задач обучения без учителя [3].

Авторский инвариант — это количественная характеристика литературных текстов или некий параметр, который однозначно характеризует своим поведением произведения одного автора или небольшого числа «близких авторов», и принимает существенно разные значения для произведений разных групп авторов. Авторский инвариант применяется в задаче идентификации авторства текста.

Задача идентификации авторства текста − это задача установления авторства неизвестного текста с помощью выделения особенностей авторского стиля и сравнения этих особенностей с другими произведениями, авторство которых известно.

Задача идентификации авторства текста

Формулировка задачи идентификации автора текста при ограниченном наборе альтернатив выглядит следующим образом: имеется множество текстов $T = \{t_1,...,t_k\}$ и множество авторов $A = \{a_1,...,a_l\}$. Для некоторого подмножества текстов $T'\{t_1,...,t_m\} \subseteq T$ авторы известны $D = \{(t_i, a_j)\}_{i=1}^{m}$. Необходимо установить, кто из множества A является истинным автором остальных текстов (анонимных или спорных) $T'' = \{t_{m+1},...,t_k\} \subseteq T$.

Существует довольно много методов анализа стиля. В целом можно разделить их на две большие группы – экспертные и формальные. Экспертные методы предполагают исследование текста профессиональным лингвистом-экспертом. К формальным относятся приемы из теории вероятностей и математической статистики (например, байесовский классификатор), алгоритмы кластерного анализа и нейронных сетей, деревья решений и генетические алгоритмы.

Наиболее часто в задачах классификации или кластеризации текстовых данных используется байесовский классификатор.

Решение задачи с помощью байесовского классификатора

Данная модель классификации базируется на понятии условной вероятности принадлежности документа *d* классу *c*.

В основе наивного байесовского классификатора лежит теорема Байеса:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

которая позволяет вычислить условную вероятность $P(A|B)$ того, что имело место событие A, если в результате эксперимента наблюдалось событие B, при известных вероятностях наступления событий A и B – $P(A|B)$ и $P(B)$, соответственно, и условной вероятности наступления события A при существующем B – $P(B|A)$.

Для данной модели документ – это вектор: $d = \{w_1,...,w_n\}$, где w_i - вес i-ого термина, а n – размер словаря выборки. Согласно теореме Байеса, вероятность принадлежности документа d к классу c вычисляется следующем образом:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)},$$

где $P(c|d)$ – вероятность, что документ d принадлежит классу c, именно ее нам надо рассчитать; $P(d|c)$ – вероятность встретить документ d среди всех документов класса c; $P(c)$ – безусловная вероятность встретить документ класса c в корпусе документов; $P(d)$ –безусловная вероятность встретить документ d в корпусе документов.

Цель классификации состоит в том, чтобы понять к какому классу принадлежит документ. Поэтому нам нужно не само значение вероятности, а наиболее вероятный класс.

Наиболее вероятным классом c^*, к которому принадлежит документ d является тот класс, для которого условная вероятность принадлежности документа d классу с максимальна:

$$c^* = argmax_c P(c|d).$$

Нам надо рассчитать вероятность для всех классов и выбрать тот класс, который обладает максимальной вероятностью. По теореме Байеса:

$$c^* = argmax_c P(c|d)*P(c)$$

Так как $d = \{w_1,...,w_n\}$, то $c^* = argmax_c P(w_1,...,w_n|c)*P(c)$.

Далее делается существенное допущение, которое и объясняет, почему этот алгоритм называют наивным. Оно звучит следующим образом: знаменатель может быть опущен, так как для одного и того же документа d вероятность $P(d)$ будет одинаковой, а это значит, что ее можно не учитывать.

Также в модели наивного байесовского классификатора предполагается, что все признаки $x_1, x_2,..., x_n$ документа d независимы друг от друга. Из-за этого допущения модель и получила название «наивная». Это очень серьезное упрощающее допущение и, в общем случае, оно неверно, но наивная байесовская модель демонстрирует неплохие результаты, несмотря на это. Также вносится уточнение, что позиция термина в предложении не важна.

Таким образом, условную вероятность $P(w_1,...,w_n|c)$ для признаков $x_1, x_2,..., x_n$ можно представить как

$$P(w_1|c)*(w_2|c)*...*(w_n|c) = \prod_i P(w_i|c_j).$$

Таким образом, для нахождения наиболее вероятного класса для документа $d = \{w_1,...,w_n\}$ с помощью наивного Байесовского классификатора, необходимо посчитать условные вероятности принадлежности документа d для каждого из представленных классов отдельно и выбрать класс, имеющий максимальную вероятность:

$$C_{NB} = argmax_c [P(c_j)*\prod_i P(w_i|c_j)$$

Теперь необходимо оценить $P(c_j)$ и $P(w_i | c_j)$. Оценить вероятность класса несложно: $P(c_j)$ является отношением количества документов класса j в обучающей выборке к общему количеству документов в выборке.

$$P(c) = \frac{D_c}{D},$$

где D_c - количество документов класса c, а D – общее количество документов в выборке.

Для оценки условных вероятностей для признаков, используется формула:

$$\hat{P}(w_i|c_j) = \frac{count(w_i|c_j)}{\sum_{w \in V} count(w|c_j)}.$$

Здесь $\hat{P}(w_i|c_j)$ определяется как отношение количества терминов w_i в классе c_j общему количеству терминов в этом классе. V – словарь обучающей выборки.

Решение задачи с помощью выделения вектора характеристик

Предполагается текстовый документ представлять в виде модели «мешка слов». Пусть $D = (d_1,...,d_{|D|})$ – множество всех текстовых документов, k – заданное число кластеров, на которое требуется разбить множество D.

Требуется задать функцию расстояния на множестве документов:

$$\rho(d_i, d_j) : D \times D \to \Re_+,$$

и провести кластеризацию текстовой коллекции. В качестве меры близости текстов, представленных векторами значений признаков, обычно используется скалярное произведение векторов.

Перед проведением кластеризации проводится предварительная обработка текстов – из них удаляются *стоп-слова* и слова, встречающиеся не более одного раза в тексте, как шумовые составляющие.

Стоп-слова формально определим как слова из некоторого ранее заданного списка S (слова, знаки или символы, которые самостоятельно не несут никакой смысловой нагрузки и просто игнорируются поисковыми системами при осуществлении ранжирования или индексации сайтов). К ним относят союзы и союзные слова, местоимения, предлоги, частицы, междометия, указательные слова, цифры, знаки препинания, вводные слова, ряд некоторых существительных, глаголов, наречий (например, сайт, давать, всегда, однако и др.)

Далее представим каждым преобразованный документ в виде вектора:

$$\mathbf{d_i} = \left(n(d_i,\omega_1)...n(d_i,\omega_j)...n(d_i,\omega_W) \right),$$

где $n(d_i,\omega_j)$ – число вхождений слова $\omega_j \in W$ в текст d_i.

Затем вводится расстояние между документами d_i и d_j как расстояние между векторами. Чаще всего в качестве функции расстояния $\rho(d_i, d_j)$ используется евклидово расстояние:

$$\rho(d_i, d_j) = \sqrt{\sum_{k=1}^{n} \left(d_i^k - d_j^k \right)^2} .$$

Таким образом, если два текста близки друг другу, то расстояние между соответствующими им векторами будет наименьшим.

Практическая задача идентификации авторских текстов

В работе было создано программное обеспечение, реализующее работу обоих рассмотренных методов классификации текстов. Программный модуль, выполняющий работу наивного байесовского классификатора был выполнен в среде Borland C++ Builder 6. Решение задачи с помощью выделения вектора характеристик – на языке C++. Для тестирования алгоритмов были выбраны небольшие тексты трех известных авторов.

В данном примере рассматривались тексты русских поэтов: А.С. Пушкина, А.А. Блока и С.А. Есенина. В обучающей выборке были заданы известные тексты (с отнесением к заданному автору). Затем на вход программе подавался текст неизвестного автора, и задачей алгоритма было установить, кто из предложенного множества автором является истинным автором этого текста.

Для обоих методов, совпадение оказалось всего лишь по 1-2 словам, так как размер выборки для обучения был небольшим. В дальнейшем предполагается увеличить обучающую выборку, для создания большего количества авторских элементов.

Заключение

В данной работе была рассмотрена одна из задач кластеризации слабоструктурированных данных – задача идентификации текстов. Были рассмотрены два метода решения данной задачи. Для обоих методов было реализовано программное обеспечение. Также в работе был решен практический пример задачи идентификации авторства текста.

Список литературы

1.Воронцов К. В. Лекции по алгоритмам кластеризации и многомерного шкалирования / К. В. Воронцов. − М.: МГУ, 2007. – 18 с.

2. Миркин Б.Г. Методы кластер – анализа для поддержки принятия решений: обзор: препринт WP7/2011/03/ Б. Г. Миркин. − М.: Изд. Дом Национального исследовательского университета «Высшая школа экономики», 2011. – 88 с.

3. Загоруйко Н.Г. Прикладные методы анализа данных и знаний. - Новосибирск: ИМ СО РАН, 1999. – 270 с.

4. Дюран Б. Кластерный анализ/ Пер. с англ. Е. З. Демиденко под ред. А.Я. Боярского / Б.Дюран, П. Одел. М.: «Статистика», 1977. – 128 с.

УДК 519.213

СУПЕРПОЗИЦИЯ СЕТ-РЕГРЕССИЙ

А. И. Иванова
научный руководитель канд.физ.-мат. наук, доцент Д. В. Семенова
Сибирский федеральный университет,
Институт математики и фундаментальной информатики

Рассмотрим вероятностное пространство (Ω, *F*, **P**). Пусть $U \subset F$ конечное множество событий выбранных из алгебры *F* этого пространства. Обозначим *N*= | *U* |.

Определение 1. Случайное множество событий *K* на конечном множестве событий $U \subset F$ определяется как отображение $K:\Omega \to 2^U$, измеримое относительно пары алгебр $\left(F, 2^{2^U}\right)$ в том смысле, что для всякого $X \in 2^{2^U}$ существует прообраз $K^{-1}(X) \in F$, такой что $\mathbf{P}(X) = \mathbf{P}\left(K^{-1}(X)\right)$.

Случайное множество событий *K*, заданное на конечном множестве событий *U*, определяется своим дискретным вероятностным распределением. Если мощность рассматриваемого множества событий $|U| = N < \infty$, то имеется 2^N видов вероятностных зависимостей между событиями этого множества, т.е. ровно столько, сколько у этого множества подмножеств.

Определение 2. Дискретное вероятностное распределение (вероятностное распределение) случайного множества событий *K*, заданного на конечном множестве избранных событий $U \subset F$ есть это набор 2^N значений вероятностной меры **P** на событиях из 2^U. Как известно, такое распределение можно задать шестью эквивалентными способами [1]. В настоящей работе рассматриваются только два из них:

P I. Вероятностное распределение I-го рода случайного множества событий *K* на *U* – это набор $\{p(X), X \subseteq U\}$ из 2^N вероятностей вида

$$p(X) = \mathbf{P}(K = X) = \mathbf{P}\left(\left(\bigcap_{x \in X} x\right) \cap \left(\bigcap_{x \in X^C} x^c\right)\right),\ X^c = U \setminus X,\ x^c = \Omega \setminus x.$$

Вероятностное распределение I-го рода всегда легитимно и удовлетворяет следующим условиям: $0 \le p(X) \le 1,\ X \subseteq U,$ и $\sum_{X \subseteq U} p(X) = 1.$

P II. Вероятностное распределение II-го рода случайного множества событий *K* на *U* – этонабор из 2^N вероятностей вида $\{p_X, X \subseteq U\}$, где $p_X = \mathbf{P}(K \supseteq X) = \mathbf{P}\left(\bigcap_{x \in X} x\right)$, которые удовлетворяют системе из 2^N неравенств Фреше-Хёффдинга

$$0 \le \max\left\{0, 1 - \sum_{x \in X}\left(1 - \mathbf{P}(x)\right)\right\} \le p_X \le \min_{x \in X} \mathbf{P}(x) \le 1.$$

Вероятностные распределения I-го и II-го рода связаны взаимно-обратными формулами обращения Мёбиуса [1]:

$$p_X = \sum_{Y \in 2^U : X \subseteq Y} p(Y), \quad p(X) = \sum_{Y \in 2^U : X \subseteq Y} (-1)^{|Y|-|X|} p_Y, \quad \text{для всех } X \in 2^U.$$

Если дано совместное распределение двух случайных множеств событий K и L, значения которых содержаться в конечных множествах U и W соответственно, то регрессией K на L называется любой оператор φ, приближенно представляющий статистическую зависимость K от L [1 – 3]. Для определения зависимости между двумя случайными множествами событий в [1 – 3] было предложено использовать сет-регрессию, которая устанавливает вид средней сет-функциональной зависимости между этими двумя случайными множествами событий.

Постановка задачи. Пусть заданы случайное множество событий K на конечном множестве $U \subset F$, случайное множество событий L на конечном множестве $W \subset F$ и случайное множество событий M на конечном множестве $S \subset F$, $U \cap W \cap S = \varnothing$. Известны совместное распределение $\{p(X,Y), X \subseteq U, Y \subseteq W\}$ случайных множеств событий K и L, где $p(X,Y) = \mathbf{P}(K = X, L = Y)$, и совместное распределение $\{p(Y,Z), Y \subseteq W, Z \subseteq S\}$ случайных множеств событий L и M, где $p(Y,Z) = \mathbf{P}(L = Y, M = Z)$. Необходимо найти сет-функцию $\Psi : 2^U \to 2^S$, которая устанавливает зависимость между случайными множествами Kи M.

Алгоритм решения задачи

1. На первом этапе необходимо найти решение задачи сет-регрессии для множеств K и Lв виде некоторого сет-среднего: $\theta(L = Y | K = X)$, $X \subseteq U, Y \subseteq W$.
2. На втором этапе необходимо найти решение задачи сет-регрессии для множеств L и M в виде некоторого сет-среднего $\delta(M = Z | L = Y)$, $Y \subseteq W, M \subseteq S$.
3. Решение задачи определяется как суперпозиция $\Psi(X) = \delta(M = Z | \theta(L = Y | K = X))$.

Комплекс программ SupPositionразработан в программном обеспечение Mathcad.

Комплекс программ SupPositionреализует следующие функции:

– Функция SetKvan определяет сет-квантиль порядка α: $\mathrm{Q}_\alpha(L = Y | K = X) = \{y \in W : \mathbf{P}(Y | \mathrm{ter}(X)) \geq \alpha\}$,

– SetMedопределяет сет-медиану: $\mathrm{Med}(L = Y | K = X) = \{y \in W : \mathbf{P}(Y | \mathrm{ter}(X)) \geq 1/2\}$,

– Функция SetMod определяет сет-моду $\mathrm{Mod}(L = Y | K = X) = \left\{Y \subseteq W : \mathbf{P}(Y | X) = \max_{X' \subseteq U} \mathbf{P}(Y | X')\right\}$,

где $\mathrm{ter}(X) = \left(\bigcap_{x \in X} x\right) \cap \left(\bigcap_{x \in X^C} x^c\right)$.

– Функция Compositionвыводит результат суперпозицию сет-регрессии.

Основными входными данными являются совместное распределение случайного множества событий.

На рис. 1 дано графическое представление решения рассматриваемой задачи для $U = \{x, y, z, w\}$, $W = \{a, b, c, d\}$, $S = \{\alpha, \beta, \gamma, \eta\}$, известных совместных распределений

$\{p(X,Y), X \subseteq U, Y \subseteq W\}$ и $\{p(Y,Z), Y \subseteq W, Z \subseteq S\}$. Решение задач сет-регрессий для первого и второго этапа искали в виде условного сет-медиана.

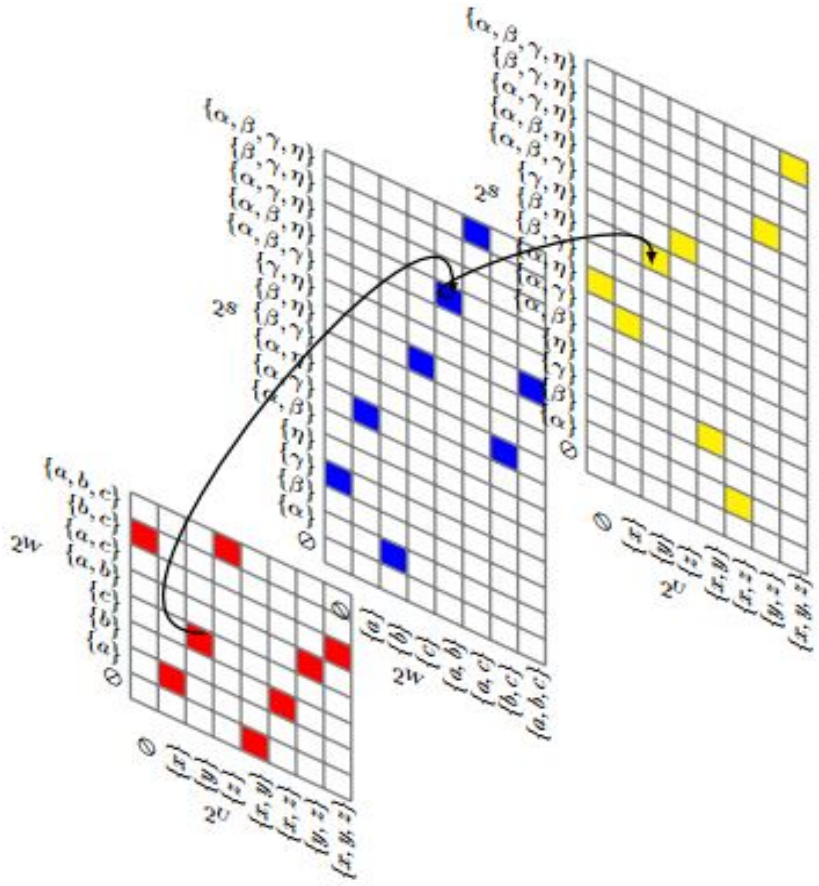


Рис.1. Суперпозиция сет-регрессий $\Psi(X) = \delta\left(M = Z \middle| \theta\left(L = Y \middle| K = X\right)\right)$.

Разработанный комплекс программ может быть использован в решении задач моделирования, прогнозирования и контроля развития статистических систем природы и общества, события в которых, как правила, представлены неколичественной информацией, имеется практическое приложение [4].

Список литературы

1. Воробьёв, О.Ю. Эвентология. Красноярск: Сибирский федеральный университет, 2007. 435 с.
2. Воробьёв, О.Ю. Сет-регрессионный анализ зависимостей событий в статистических системах: учеб.пособие О.Ю. Воробьев, А.Ю. Фомин-Красноярск: ИВМ СО РАНбКрасГУ, 2004. 116 с.
3. Тарасова, О. Ю. Сеточные и регрессионные алгоритмы аппроксимации сложных систем событий: автореф. дис. на соиск. учен. степ. канд. физ.-мат. н.(151301) / Ольга Юрьевна Тарасова;-Красноярск, 2007. – С. 14-16
4. Семенова Д.В., Иванова А.И. Нечеткая и сет-регрессионная модели распределения потребительских предпочтений между фирмами. Труды XIII ФАМЭМС'2014 конференции (под ред. О.Ю. Воробьёва). --- Красноярск: СФУ, 2014 – С. 222-227.

УДК 519.23

ЗАДАЧА КЛАСТЕРИЗАЦИИ ДАННЫХИ ЕЁ РЕШЕНИЕ МЕТОДАМИ K-СРЕДНИХ И ЛАНСА-УИЛЬЯМСА

Кулистов А.В.,
научный руководительканд. физ.-мат. наук, доцентБаранова И. В.
Сибирский Федеральный Университет,
Институт математики и фундаментальной информатики

Введение

Кластерный анализ является одним из самых востребованных направлений современной теоретической информатики и статистического анализа данных.Он имеет широкий спектр практических применений во многих областях человеческой деятельности: в извлечении и поиске информации, в задачах принятия решений, задачах обработки и проектирования многомерных данных, абстракции данных, классификации данных и других.

Кластерный анализ представляет собой раздел статистического анализа данных, объединяющий методы разбиения (группировки) множества объектов на сравнительно однородные группы.

Кластерпредставляет собой часть данных (в типичном случае – подмножество объектов или подмножество переменных, или подмножество объектов, характеризуемых подмножеством переменных), которая выделяется из остальныхданных наличием некоторой однородности ее элементов. В простейшем случае речь идет о похожести элементов, в идеальном случае – о совпадающих значениях основных переменных или иного рода близости, выражаемой геометрической близостью соответствующих объектов.

Целью данной работы является изучение и реализациянаиболее популярных алгоритмов кластеризации данных:метода k-среднихи метода Ланса-Уильямса. Также в работе рассматривается решение практической задачи кластеризации данных с помощью перечисленных методов.

Постановка задачи кластеризации

Пусть $X = \{x_1, x_2, \ldots, x_m\}$ – множество объектов, заданных значениями в пространстве признаков $P = \{P_1, P_2, \ldots, P_n\}$ (т.е. каждый объект $x_i = (x_i^1, x_i^2, \ldots, x_i^n)$) и задана функция расстояния (метрика) между объектами $\rho(x_i, x_j)$, $x_i, x_j \in X$.

Функция кластеризации представляет собой функцию $f : X \to Y$, которая любому объекту $x \in X$ ставит в однозначное соответствие номер кластера $y \in Y = \{1, \ldots, k\}, k \le m.$

Тогда постановку задачу кластеризации данных можно сформулировать следующим образом:

Требуется найти такую функцию кластеризации f^*, чтобы

$$Q(f^*, C, \rho) = \min_f Q(f, C, \rho),$$

где $Q(f, C, \rho)$ – выбранный критерий качества кластеризации.

Следует отметить, что выбор того или иного функционала качества, как правило, опирается на эмпирические соображения.

Рассмотрим наиболее распространенный функционал качества разбиения. Пусть исследователем выбрана метрика ρ в пространстве X и пусть $S = (S_1, S_2,..., S_p)$ — некоторое фиксированное разбиение наблюдений $x_1, ..., x_n$ на заданное число p классов $S_1, S_2, ..., S_p$.

За функционал качества берут сумму («взвешенную») внутриклассовых дисперсий

$$Q(S) = \sum_{l=1}^{p} \sum_{x_i \in S_l} \rho^2(x_i, x_l),$$

где x_l — вектор средних для l-го кластера.

Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин:

- не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд эвристических критериев, а также ряд алгоритмов, не имеющих четко выраженного критерия, но осуществляющих достаточно разумную кластеризацию, по построению все они могут давать разные результаты;
- число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием;
- результаты кластеризации существенно зависят от метрики, выбор которой, как правило, также субъективен и определяется экспертом.

Основные методы кластеризации данных

Существует около 100 разных алгоритмов кластеризации, однако общепринятой классификации методов кластеризации не существует, но их можно разделить на две группы: иерархические и неиерархические.

Выбор метода кластеризации зависит от количества данных и от того, есть ли необходимость работать одновременно с несколькими типами данных.

Суть иерархической кластеризации состоит в последовательном объединении меньших кластеров в большие или разделение больших кластеров на меньшие. Преимуществом иерархических методов кластеризации является их наглядность.

Иерархические алгоритмы связаны с построением дендрограмм, которые являются результатом иерархического кластерного анализа. Дендрограмма описывает близость отдельных точек и кластеров друг к другу, представляет в графическом виде последовательность объединения (разделения) кластеров.

При большом количестве наблюдений иерархические методы кластерного анализа не пригодны. В таких случаях используют неиерархические методы, основанные на разделении, которые представляю собой итеративные методы дробления исходной совокупности. В процессе деления новые кластеры формируются до тех пор, пока не будет выполнено правило остановки.

В нашей работе подробно рассматриваются два наиболее популярных метода кластеризации: метод k-средних и алгоритм Ланса–Уильямса.

Метод k-средних

Алгоритм *k*–средних является одним из самых популярных методовкластеризации данных. Также этот метод называют быстрым кластерным

анализом. Для возможности использования этого метода необходимо иметь гипотезу о наиболее вероятном количестве кластеров.

Центроид– точка, представляющая собой центр масс точек кластера, т.е. покоординатные средние точек из кластеров:

$$\mu_j = (\mu_{j1,}, \mu_{j2}, ..., \mu_{jn}),$$

$$\mu_j = \sum_{x_j \in S} x_{jl}$$

где $l = 1,..., n$, $j = 1,..., k$. Каждому кластеру соответствует один центр.

Приведем подробное описание работы алгоритма k-средних.

Пусть имеется множество точек данных $X = \{x_1, ..., x_m\}$, где $x_i = (x_i^1, x_i^2, ..., x_i^n) \in \Re^n$.

До начала работы алгоритма необходимо задать количество кластеровk. Затем начинается работа метода.

Алгоритм k-средних имеет следующий вид:

1.Задаются начальныецентроиды. Выбор начальных значений может осуществляться следующим образом:

- случайный выбор *k*объектов (точек) x_i,
- выбор первых*k*точек,
- выбор *k*точекс максимальными значениями расстояний между ними.

Все кластеры пока задаются пустыми: $S_j = \{\varnothing\}$, $j = 1,\ldots,k$.

2. Производится распределение объектов по кластерам. Точка $x_i, i = 1,..., n$ относится к ближайшему кластеру, т.е. $x_i \in S_{j*}$, где

$$p(x_i, \mu_{j*}) = \min_{j=1,...,k} p(x_i, \mu_j)$$

В качестве метрики чаще всегоиспользуется евклидова метрика.В результате каждый объект относится к определенному кластеру.

3. Вычисляются новые центры кластеров μ_j, $j = 1,..., k$ как центры масс новых кластеров S_j, $j = 1,..., k$ полученных на предыдущем этапе:

$$\mu_j = \sum_{x_j \in S} x_{jl}.$$

4. Итерационный процесс вычисления центров и перераспределения объектов продолжается до тех пор, пока не выполнится одно из условий:

- кластерные центры μ_j стабилизировались (перестали изменяться);
- число итераций равно максимальному числу итераций (ограничение на число итераций).

Выбор числа кластеров является сложным вопросом. Если кластеры служат для дальнейшего машинного использования, то можно выбирать большие значения *k*, сообразуясь только с имеющимся объемом памяти для их хранения. Если же кластеризацией будет пользоваться человек «вручную», то чаще всего используется от пяти до девяти кластеров.

Алгоритм Ланса-Уильямса

Имеется множество точек данных $X = \{x_1,...,x_m\}$. В алгоритме обозначение C_t используется для обозначения множества кластеров, полученных на t–ой итерации алгоритма, под R_t понимается расстояние между кластерами на –ой итерации.

Описание алгоритма:

1. Задаем одноэлементные кластеры: $t = 1$

$R(\{x_i\},\{x_j\}) = p(x_i, x_j)$

Для всех $t = 2,...,1$ (t – номер итерации);

2. Найти в C_{t-1} два ближайших кластера $U,V \in C_{t-1}$; $R_t := R(U,V)$

$(U,V) := \min_{U \neq V} R(U,V)$

3. Слить их в один кластер: $W := U \cup V; C_{t-1} \cup \{W\}\{U,V\}$;

4. Для всех $S \in C_t$ вычислить $R(W,S)$ по формуле Ланса-Уильямса.

Формула Ланса-Уильямса в общем виде выглядит следующим образом:

$$R(U \cup V, S) = \alpha_U \cdot R(U,V) + \alpha_V \cdot R(V,S) + \beta \cdot R(U,V) + \gamma \cdot |R(U,S) - R(V,S)|,$$

где $\alpha_U, \alpha_V, \beta, \gamma$ – числовые параметры.

Перечислим наиболее употребляемые частные случаи формулы Ланса–Уильямса:

а) Расстояние ближнего соседа:

$$R^б(W,S) = \min_{R_t := R(U,V)} p(w,s),\ \alpha_U = \alpha_V = \frac{1}{2}, \beta = 0, \gamma = \frac{1}{2}.$$

б) Расстояние дальнего соседа:

$$R^д(W,S) = \max_{w \in W, s \in S} p(w,s),\ \alpha_U = \alpha_V = \frac{1}{2}, \beta = 0, \gamma = \frac{1}{2}.$$

в) Групповое среднее расстояние:

$$R^с(W,S) = \frac{1}{|W||S|} \sum_{w \in W} \sum_{s \in S} p(w,s),\ \alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = \gamma = 0$$

Практическая задача кластеризации

В работе решается практическая задача кластеризации задача кластеризации марокрастворимого кофе на основе предпочтений потребителей. Для решения задачи использовалась статистика, полученная в результате опроса потребителейрастворимого кофе из 8 регионов РФ. По заданиюодной из торговых фирм были изучены предпочтения покупателей имелкооптовых продавцов растворимого кофе. Опрос проводился в форме интервью с 1400 покупателями кофе. Анкетасостояла из 10 основных вопросов и 3 дополнительных, посвященныхсоциально-демографической информации о покупателе. Вопросы позволяли уточнить частоту потребления и покупок кофе покупателем,список марокрастворимого кофе, с которыми знаком потребитель, а также основныехарактеристики продукта, которые являются наиболее ценными дляпотребителя. Были предложены следующие характеристики каждой маркикофе: вкус, аромат, консистенция и привлекательность марки.Именно эти показатели использовались для решения задачикластеризации. Вкус, аромат и консистенция каждой марки кофеоценивались каждым потребителем по шкале от 1 до 10 (наихудшее

инаилучшее значения, соответственно). Были вычислены средние значения по каждому показателю и выполнена кластеризация марок.

Кластеризация данных проводилась с помощью метода k-среднихиалгоритма Ланса-Уильямса.

Также в работе было проведено сравнение кластеров, полученных в результате работы каждого метода.

Список литературы

1. Факторный, дискриминантный и кластерный анализ: Пер с англ./Дж. – О. Ким, Ч. У. Мьюллер, У. Р. Клекка и др.; Под ред. И. С. Енюкова. – Москва: Финансы и статистика, 1989. – 215 с.

2.Олдендерфер М.С. и Блэшфилд Р.К. Кластерный анализ / Факторный, дискриминантный и кластерный анализ. – Москва: Финансы и статистика, 1989. – С. 139–214.

3. http://www.bibliotekar.ru/economicheskaya-statistika-2/13.htm

УДК 519.178

АЛГОРИТМЫ И ПРОГРАММЫ ВЫЧИСЛЕНИЯ МЕРЫ СБАЛАНСИРОВАННОСТИ ЗНАКОВОГО ГРАФА

Махонин И. В.,
научный руководитель д-р физ.-мат. наук Быкова В. В.
Сибирский федеральный университет

Впервые знаковые графы были введены Ф. Харари для решения задач, возникающих в социальной психологии. В дальнейшем они нашли применение в теории групп, в неферромагнитной модели Изинга и в кластеризации данных. На сегодняшний день класс знаковых графов математически малоизучен. Все понятия, связанные со знаковыми графами, были введены преимущественно психологами или экономистами. На данный момент известны некоторые меры сбалансированности знаковых графов и методики их вычисления для графов малой размерности. Однако недостаточно исследованными остаются вопросы, касающиеся свойств знаковых графов, отсутствуют эффективные алгоритмы распознавания сбалансированности, методов вычисления мер сбалансированности, математических формулировок задач, связанных с построением экстремальных множеств элементов знакового графа с заданным требованием сбалансированности.

В данной работе представлены алгоритмы и программы для вычисления меры сбалансированности знакового графа. Предложены два алгоритма вычисления точного меры, выполнено их сравнение по результатам вычислительных экспериментов.

Знаковым графом называется пара $\sum = (G, \sigma)$, где $G = (V, E)$ – неориентированный граф, на ребрах которого задана функция знака σ: $E \to \{+, -\}$, $|V| \geq 2$, $|E| \geq 1$. Знак цикла (как соответствующее множество ребер графа $\sum$) равен произведению знаков его ребер. Другими словами, цикл графа является положительным, если число отрицательных ребер в нем четное, и отрицательным, если нечетное. Для простоты изложения знаковый граф будем обозначать далее через $G = (V, E)$.

Знаковый граф $G = (V, E)$ называется сбалансированным, если любой его простой цикл положительный. Свойства сбалансированного знакового графа отражает теорема 1.

Теорема 1 (Картрайт – Харари) [2]. Следующие утверждения эквивалентны:

- $G = (V, E)$ − сбалансированный знаковый граф;
- множество вершин графа $G = (V, E)$ можно разбить на два подмножества A и B таким образом, что $A \cup B = V$, $A \cap B = \varnothing$ и любое ребро, соединяющее вершины из одного подмножества, имеет знак «+», а ребра, соединяющие вершины из разных подмножеств, имеют знак «–»;
- любые две простые (u, v)-цепи графа $G = (V, E)$ имеют одинаковый знак.

Вероятность того, что случайно выбранный знаковый граф сбалансирован, крайне мала. Поэтому интересен вопрос, насколько сбалансирован несбалансированный граф. Ответ на этот вопрос дает мера сбалансированности знакового графа. Возможны различные определения меры сбалансированности, однако в социологии и психологии ее определяют следующим образом. Мера сбалансированности знакового графа G − отношение числа C^+ простых положительных циклов к общему числу C циклов этого графа:

$$b(G) = C^+ / C. \quad (1)$$

Таким образом, для вычисления меры сбалансированности по формуле (1) необходимо выявить все простые циклы графа и выявить их знаки. Следует заметить, что это может потребовать много времени, а именно $O(n^3 \cdot 2^n)$ операций, где $n = |V|$. Так, для

полного графа с n вершинами общее число простых циклов определяется формулой (эта ситуация отвечает худшему случаю):

$$C = \sum_{k=3}^{n} \frac{n!}{(n-k)! \cdot 2k} = O(2^n),$$

а число различных вариантов расстановки знаков ребер равно $2^{n(n-1)/2}$. Такие задачи называют трудно разрешимыми (за реальное время при больших значениях n). Точное значение рассмотренной выше меры сбалансированности согласно (1) может быть найдено только полным перебором всех простых циклов исходного графа G. Даже если ограничить область поиска (анализировать лишь часть простых циклов), то формула (1) не гарантирует получение некоторого приближения к точному значению меры.

Для решения задачи вычисления меры сбалансированности, предлагается два переборных алгоритма: алгоритм А1 (генерация всех циклов графа на основе множества фундаментальных циклов); алгоритм А2 (генерация всех циклов графа с помощью специального обхода вершин графа).

Классический алгоритм А1 вначале строит остовное дерево входного графа. А затем находит множество его фундаментальных циклов [1]. Все другие циклы графа формируются на основе теоремы 2.

Теорема 2 [1]. Произвольный цикл графа G можно однозначно представить, как симметрическую разность некоторого числа фундаментальных циклов.

Заметим, что симметрическая разность подмножества множества фундаментальных циклов не всегда задает цикл в графе G.

В процессе работы алгоритма А1 выполняются следующие основные операции: формирование результата симметрической разности некоторого подмножества множества фундаментальных циклов; проверка, определяет ли результат симметрической разности простой цикл в графе; проверка знака цикла; подсчет общего числа простых циклов и числа положительных циклов.

Трудоемкость подготовительных действий (построение основного дерева и нахождение множества фундаментальных циклов) составляет $O(n^2)$. Для проверки условия, определяет ли результат симметрической разности простой цикл, и знак этого цикла требуется в худшем случае просмотреть $m = O(n^2)$ ребер графа. Формирование результата симметрической разности можно выполнить за $O(n^3)$ элементарных операций. Число фундаментальных циклов связного графа $G = (V, E)$ определяется формулой [1]: $\lambda = m - n + 1$, где $m = |E|$ – число ребер, $n = |V|$ – число вершин. Для ациклического графа $\lambda = 0$, а для полного графа

$$\lambda = \frac{(n-1)(n-2)}{2} = O(n^2).$$

Алгоритм А1 выполняет просмотр всех 2^{λ} различных подмножеств множества фундаментальных циклов. Таким образом, можно говорить, что сложность алгоритма А1 для произвольного графа составляет

$$O(n^3 \cdot 2^{\lambda}). \quad (2)$$

Для плотного графа G, когда $\lambda = O(n^2)$, согласно (2) время работы алгоритма составляет

$$O(n^3 \cdot 2^{n^2}), \quad (3)$$

а для разреженного, когда $\lambda = O(n)$, время работы будет

$$O(n^3 \cdot 2^{n}). \quad (4)$$

Предлагаемый алгоритм А2 формирует множество всех простых циклов графа с помощью обхода всех вершин графа и поиска всех циклов, проходящих через очередную просматриваемую вершину. Теперь оценим сложность алгоритма А2. Для этого достаточно оценить число простых циклов, проходящих через фиксированную вершину графа. Не трудно показать, что это число составляет $O(2^n)$. Поскольку алгоритм А2 выполняет просмотр всех вершин поочередно, и каждый найденный простой цикл проверяет на знак, то сложность алгоритма А2 составляет

$$O(n^3 \cdot 2^n). \quad (5)$$

Сравнение формул (3)–(5) позволяет сделать следующий вывод, что алгоритм А2 имеет сложность (по времени выполнения) ниже, чем алгоритм А1, и их сложности сопоставимы на разреженных графах. Этот вывод подтверждают результаты вычислительных экспериментов, выполненных с помощью программ, реализующих алгоритмы А1 и А2.

Программы разработаны в интегрированной среде RAD Studio XE7 на языке программирования Delphi. Для эксплуатации программ необходим персональный компьютер типа IBM PC Pentium IV с операционной системой Windows XP/Vista/7/8 и оперативной памятью от 512 Мб.

Некоторые результаты вычислительных экспериментов представлены в следующей таблице.

Граф	n	m	k	λ	Вычисление меры							
					Количество циклов				$b(G_i)$		Время работы	
					Все		Положительные					
					А2	А1	А2	А1	А2	А1	А2	А1
*G*1	6	3	3	0	0	0	0	0	0	0	0,00054	0,00041
*G*2	6	4	2	0	0	0	0	0	0	0	0,00048	0,00048
*G*3	6	5	1	0	0	0	0	0	0	0	0,00056	0,00049
*G*4	6	6	2	2	3	3	1	1	0,33	0,33	0,00015	0,0005
*G*5	6	7	2	3	6	6	6	6	1	1	0,00057	0,00044
*G*6	6	8	1	3	4	4	2	2	0,5	0,5	0,00058	0,00028
*G*7	6	9	1	4	10	10	10	10	1	1	0,00058	0,00046
*G*8	6	10	1	5	20	20	9	9	0,45	0,45	0,00038	0,00053
*G*9	6	11	1	6	32	32	14	14	0,43	0,43	0,0005	0,00131
*G*10	6	12	1	7	59	59	28	28	0,47	0,47	0,00035	0,00523
*G*11	6	13	1	8	85	85	41	41	0,48	0,48	0,00053	0,03963
*G*12	6	14	1	9	133	133	63	63	0,47	0,47	0,00051	0,24694

Разработанные программы могут быть использованы для вычисления меры сбалансированности знаковых графов, число вершин которых не более 20.

Список литературы

1. Лекции по теории графов / В. А. Емеличев [и др.]. – М.: Книжный дом «ЛИБРОКОМ», 2012.

2. Шварц, Д. А. О мере сбалансированности полных знаковых графов. / Д. А. Шварц. – М.: Изд. дом. Высшей школы экономики, 2013.

УДК 519.2

СРЕДНЕ-ФЕНОМЕННЫЙ ПОРТФЕЛЬНЫЙ АНАЛИЗ

Нифонтов А.С.
научный руководитель д-р физ.-мат. наук, профессор Воробьев О.Ю.
Сибирский федеральный университет

Введение

Во второй половине XX века Гарри Марковиц предложил методы для формирования оптимальных портфелей активов. Так же в его работах были сформулированы понятия “риск” и “доходность” и их математическое обоснование. Работа основывается на эвентологической интерпретации работ Марковица.

Средне-феноменная задача Марковица

Рассмотрим прямую средне-феноменную эвентологическую задачу Марковица, в которой необходимо найти неизвестные доли событий в средне-феноменном портфеле при известном вероятностном распределении множества этих портфельных событий.

Каждый феномен событий это только один из 2^N все возможных феноменов. А каждый феномен можно интерпретировать, как одна из всевозможных точек зрения на рассматриваемый портфель. Используя средне-феноменную постановку, мы сможем оценивать портфель со всех точек зрения одновременно.

Пусть $\{\mathfrak{X}(\boldsymbol{p}), \boldsymbol{a}\}$ — портфель событий, где X — множество портфельных событий с известным вероятностным распределением

$$\boldsymbol{p} = \{p(\mathrm{X}) : \mathrm{X} \subseteq \mathfrak{X}\},$$

а

$$\boldsymbol{a} = (a_x : a_x \geq 0, x \in \mathfrak{X}, \sum_{x \in \mathfrak{X}} a_x = 1)$$

— множество неизвестных долей этих событий в портфеле.

Предполагается, что доходности портфельных событий $x \in \mathfrak{X}$ определяются их индикаторами:

$$\Upsilon_x = 1_x.$$

Тогда средняя доходность каждого портфельного события $x \in \mathfrak{X}$ является вероятностью этого события:

$$\mathbf{E}\Upsilon_x = \mathbf{E}1_x = \mathbf{P}(x),$$

дисперсия доходности портфельного события равна

$$\mathbf{D}\Upsilon_x = \mathbf{D}1_x = \mathbf{P}(x)\big(1 - \mathbf{P}(x)\big) = \sigma_x^2,$$

где $\sigma_x^2 = D1_x$ - дисперсия индикатора портфельного события $x \in \mathfrak{X}$.

Доходность всего портфеля событий полагается равной линейной комбинации доходностей его событий с долями из $\boldsymbol{a}$:

$$\Upsilon_{\mathfrak{X}} = \sum_{x \in \mathfrak{X}} a_x 1_x.$$

X-феномен стандартного портфельного множества событий определяется по формуле

$$\mathfrak{X}^{(c|\mathrm{X})} = \mathrm{X} + (\mathfrak{X} - \mathrm{X})^{(c)}.$$

Феномен, значения вероятностей, которого первоначально известны, будем называть базовым феноменом.

Теперь для каждого $\mathfrak{X}^{(c|\mathrm{X})}$ рассчитываем значения математического ожидания и дисперсии по формулам прямой эвентологической задачи Марковица.

$$\mathbf{E}(\Upsilon_{\mathfrak{X}^{(c|\mathrm{X})}}) = \sum_{x \in \mathfrak{X}^{(c|\mathrm{X})}} a_x \mathbf{P}(x),$$

а дисперсия — это квадратичная форма от $\boldsymbol{a}$:

$$\mathbf{D}(\Upsilon_{\mathfrak{X}^{(c|\mathrm{X})}}) = \sum_{x \in \mathfrak{X}^{(c|\mathrm{X})}} \sum_{y \in \mathfrak{X}^{(c|\mathrm{X})}} a_x a_y \mathrm{Kov}_{xy},$$

с коэффициентами, равными $\mathrm{Kov}_{xy} = \mathbf{P}(x \cap y) - \mathbf{P}(x)\mathbf{P}(y)$ — парной ковариации событий x и y.

Переменной $p_{\mathfrak{X}^{(c|\mathrm{X})}}$ будем обозначать вес X-феномена или вероятность выполнения феномена $\mathfrak{X}^{(c|\mathrm{X})}$относительно других феноменов. Сумма весов всех феноменов равна единице.

$$\sum_{\mathrm{X} \subseteq \mathfrak{X}} p_{\mathfrak{X}^{(c|\mathrm{X})}} = 1$$

Теперь просуммируем по всем феноменам математические ожидания, помножив на вес феномена. Так же просуммируем и дисперсии.

$$< \mathbf{E}(\Upsilon_{\mathfrak{X}^{(c|\mathrm{X})}} >_{\mathrm{X} \subseteq \mathfrak{X}} = \sum_{\mathrm{X} \subseteq \mathfrak{X}} p_{\mathfrak{X}^{(c|\mathrm{X})}} * \mathbf{E}(\Upsilon_{\mathfrak{X}^{(c|\mathrm{X})}})$$

$$< \mathbf{D}(\Upsilon_{\mathfrak{X}^{(c|\mathrm{X})}} >_{\mathrm{X} \subseteq \mathfrak{X}} = \sum_{\mathrm{X} \subseteq \mathfrak{X}} p_{\mathfrak{X}^{(c|\mathrm{X})}} * \mathbf{D}(\Upsilon_{\mathfrak{X}^{(c|\mathrm{X})}})$$

Постановка прямой средне-феноменной эвентологической портфельной задачи выглядит стандартным образом: *найти такое множество долей* $\boldsymbol{a}$ *событий в данном портфеле с известным вероятностным распределением* $\boldsymbol{p}$*, которое обеспечивает данное фиксированное значение средней доходности портфеля при его минимальном риске (дисперсии портфеля):*

$$< \mathbf{E}_{\mathfrak{X}^{(c|\mathrm{X})}} >_{\mathrm{X} \subseteq \mathfrak{X}} = < \Upsilon_{\mathfrak{X}} >$$

$$< \mathbf{D}_{\mathfrak{X}^{(c|\mathrm{X})}} >_{\mathrm{X} \subseteq \mathfrak{X}} \rightarrow \min_{a}$$

Итогом решения прямой средне-феноменной эвентологической портфельной задачи будет распределение портфеля, которое в среднем эффективно на всех феноменах.

Примеры решения средне-феноменной задачи

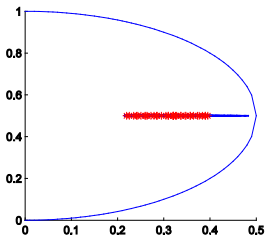
Для решения средне-феноменнной задачи опишем значения феноменов с помощью таблицы.

1	x	y	z	γ_1
2	x	y	z^c	γ_2
3	x	y^c	z	γ_3
4	x	y^c	z^c	γ_4
5	x^c	y	z	γ_5
6	x^c	y	z^c	γ_6
7	x^c	y^c	z	γ_7
8	x^c	y^c	z^c	γ_8

где первый столбец - номер феномена, x, y, z - события и их дополнение - x^c, y^c, z^c, а γ_i - вес -го феномена. Для удобства множество весов все феноменов будем обозначать Г.

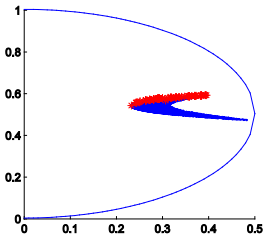
Смоделируем средне-феноменный портфель событий, веса феноменов в котором равны.

Портфель построим из трех событий с вероятностями (0.9 0.4 0.2) соответственно. Вес каждого феномена будет равен $\gamma_i = \frac{1}{8}$. Средне-феноменная пуля, построенная при решении, получается вырожденной из-за того, что противоположные(например (x, y, z) и (x^c, y^c, z^c)) феномены "уничтожают" друг друга. Решение, полученное на данной задаче, не имеет смысла, так как полученная эффективна граница отображает всевозможные значения этой средне-феноменной пули. На практике подобные задачи почти не встречаются, в связи с тем, что веса нескольких феноменов сильно преобладают над весами остальных, а значения некоторых весов стремятся к нулю.

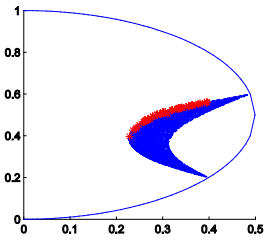


На графиках синие точки обозначают возможные значения портфеля, а красные звездочки – отображают эффективную средне-феноменную границу, а на графиках отдельных феноменов указывают распределения, которые легли на эффективную границу.

Смоделируем средне-феноменный портфель событий с вероятностями из прошлого пункта, но теперь веса феноменов распределены не равно мерно, но нулевых весов нет $\Gamma = (0.05, 0.05, 0.15, 0.4, 0.2, 0.05, 0.05, 0.05)$.

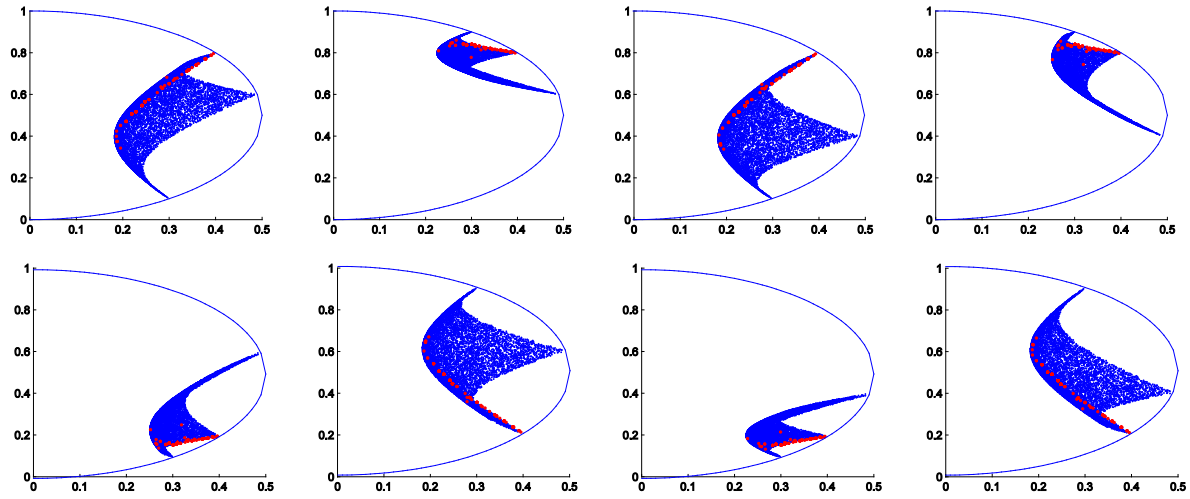


И второй случай с весами феноменов, где преобладают два феномена, а у остальных феноменов веса равны нулю $\Gamma = (0, 0, 0, 0, 0.6, 0.4, 0, 0)$.

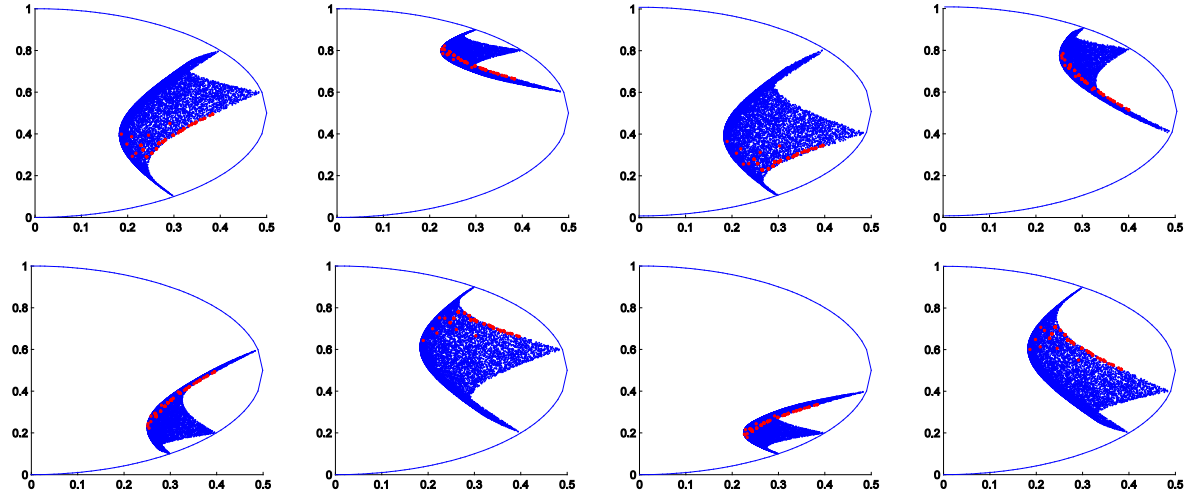


Теперь для рассмотренных выше случаев вычислим и стандартную пулю Марковица. Значения распределения активов, при которых доходность портфеля лежит на эффективной границе, найденные при решен средне-фенноменной постановки отобразим на пуле построенной для каждого феномена в отдельности.Первые 8 графиков отображают поведение пули Марковица для случая $\Gamma = (0.05, 0.05, 0.15, 0.4, 0.2, 0.05, 0.05, 0.05)$, а вторые восемь графиков отображают случай $\Gamma = (0, 0, 0, 0, 0.6, 0.4, 0, 0)$.

Первый случай.



Второй случай.



На графиках видно, как средне-феноменная эффективная граница смещается в зависимости от распределения весов феноменов. И при преобладании нескольких феноменов средне-феноменная граница сдвигается к одной из возможных границ, не всегда являющихся эффективными.

УДК 519.25

МОДИФИКАЦИЯ АЛГОРИТМА КЛАСТЕРИЗАЦИИ ДАННЫХ FOREL

Ооржак О.Е.,
научный руководитель: канд. физ.-мат. наук Баранова И. В.
Сибирский федеральный университет

Одной из основных задач интеллектуального анализа данных на настоящий момент является кластеризация – процесс получения «кластеров» или групп похожих объектов. Кластерный анализ представляет собой раздел статистического анализа, объединяющий методы разбиения совокупности объектов на однородные группы (кластеры). Вычислительные процедуры кластерного анализа данных применяются в самых разных предметных областях и позволяют обнаружить в данных неизвестные ранее закономерности.

Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин:

- Не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд эвристических критериев, а также ряд алгоритмов, не имеющих четко выраженного критерия, но осуществляющих достаточно разумную кластеризацию, по построению все они могут давать разные результаты;
- Число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием;
- Результаты кластеризации существенно зависят от метрики, выбор которой, как правило, также субъективен.

Постановка задачи кластеризации – пусть $X = \{x_1, x_2, \ldots, x_m\}$ – множество объектов, заданных значениями в пространстве признаков $P = \{P_1, P_2, \ldots, P_n\}$ (т.е. каждый объект $x_i = \left(x_i^1, x_i^2, \ldots, x_i^n\right)$) и задана функция расстояния (метрика) между объектами $\rho\left(x_i, x_j\right)$, $x_i, x_j \in X$.

Определение 1.*Функцией кластеризации*называется функция $f : X \to Y$, которая любому объекту $x \in X$ ставит в однозначное соответствие номер кластера $y \in Y = \{1, \ldots, k\}$, $k \le m$.

Определение 2. *Множество кластеров* $C = \{C_1, C_2, \ldots, C_k\}$, $k \le m$, представляет собой разбиение множества объектовX такое, что *кластер* $C_i = \{x \in X,\ f(x) = i\}$, $C_i \cap C_j = \varnothing$. Причем для$C$ справедливо следующее: если $x_i, x_j \in C_i$, то $\rho\left(x_i, x_j\right) \to \min$. Если $x_i \in C_i, x_j \in C_j$, то $\rho\left(x_i, x_j\right) \to \max$.

Тогда постановку задачу кластеризации данных можно сформулировать следующим образом: требуется найти такую функцию кластеризации f^*, чтобы

$$Q\left(f^*, C, \rho\right) = \min_f Q\left(f, C, \rho\right),$$

где $Q\left(f, C, \rho\right)$ – выбранный критерий качества кластеризации.

Итак, пусть есть несколько разбиений на кластеры. Как их оценить? Существует много разновидностей функционалов качества кластеризации, но нет «самого правильного» функционала. По сути дела, каждый метод кластеризации можно

рассматривать как точный или приближённый алгоритм поиска оптимума некоторого функционала. Какие же общие требования нужно предъявить к результату разбиения? Их немного.

1. Среднее внутрикластерное расстояние должно быть как можно меньше

$$\frac{\sum_{i<j}[c(x_i)=c(x_j)]\rho(x_i,x_j)}{\sum_{i<j}[c(x_i)=c(x_j)]}\to\min.$$

2. Среднее межкластерное расстояние должно быть как можно больше

$$\frac{\sum_{i<j}[c(x_i)\neq c(x_j)]\rho(x_i,x_j)}{\sum_{i<j}[c(x_i)\neq c(x_j)]}\to\max.$$

Если алгоритм вычисляет центры кластеров μ_c,то можно определить функционалы, вычислительно более эффективные.

Сумма средних внутрикластерных расстояний должна быть как можно меньше:

$$\sum_{c\in C}\frac{1}{|c|}\sum_{X\in c}\rho^2(\mathrm{x},\mu_c)\to\min.$$

Сумма межкластерных расстояний должна быть как можно больше:

$$\sum_{c\in C}\rho^2(\mu_c,\mu)\to\max,$$

где μ – центр масс всей выборки[1].

Для вычисления расстояния между объектами используются различные меры сходства (меры подобия), называемые также метриками или функциями расстояний. Наиболее популярными являются: евклидово расстояние, расстояние Хемминга, расстояние Чебышева, расстояние Махалонобиса, расстояние Журавлева.

Существует множество различных алгоритмов кластеризации, однако общепринятой классификации методов кластеризации не существует, но их можно разделить на две группы: иерархические и неиерархическиеПри иерархической кластеризации выполняется не просто разбиение на классы, а иерархия разбиений, т.е. последовательное объединение меньших кластеров в большие или разделение больших кластеров на меньшие.Неиерархические алгоритмы, напротив, в результате работы выдают некоторое конкретное разбиение и имеют ряд параметров, позволяющих настраивать алгоритм для имеющихся данных.

Одним из наиболее эффективных методов кластеризации числовых данных является алгоритм ФОРЭЛ (формальный элемент, FOREL) – иерархический алгоритм кластеризации, основанный на идее объединения в один кластер объектов в областях их сгущения. Метод был предложен Загоруйко Н.Г. и Елкиной В.Н. в 1967 году [2].

Опишем поэтапно данный алгоритм:

1. Предположим, что мы находимся в линейном пространстве (к примеру, R^n).
2. Пусть задана точка x_0 и параметр R.
3. Выделим все точки из набора данных, попадающие в сферу радиуса Rс центром в x_0.
4. Перенесем центр кластера из x_0 в точку центра масс точек, которые были выбраны на предыдущем шаге.
5. Повторять, пока центр не останется на месте.
6. Центр – это и есть «формальный элемент», т.к. он не обязательно содержится в исходной выборке.

Формализуем вышеизложенные пункты.

- Инициализировать множество некластеризованных точек U:=Xи множество кластеров $C := \varnothing$.
- Пока $U \neq \varnothing$:
  - Выбрать случайную точку x_0.
  - Пока процесс не стабилизируется:
    - Образовать кластер $c = \{x \in X \mid \rho(x, x_0) < \mathrm{R}\}$.
    - $x_0 := \frac{1}{|c|} \sum_{X \in c} x$.
    - $U := U \setminus c$, $\mathrm{C} := \mathrm{C} \cup \{c\}$.
- Выдать множество полученных кластеров C.

У алгоритма ФОРЭЛ имеется ряд преимуществ:

1. Точность минимизации функционала качества (при удачном подборе параметра R)
2. Наглядность визуализации кластеризации
3. Сходимость алгоритма
4. Возможность операций над центрами кластеров – они известны в процессе работы алгоритма
5. Возможность подсчета промежуточных функционалов качества, например, длины цепочки локальных сгущений
6. Возможность проверки гипотез схожести и компактности в процессе работы алгоритма

Однако имеется и ряд недостатков, которые являются «обратной стороной медали» самой идеи алгоритма – вычислений центров (формальных элементов) на каждой итерации, а также следствием его неиерархичности.

1. Относительно низкая производительность (введение функции пересчета поиска центра при добавлении одного объекта внутрь сферы).
2. Плохая применимость алгоритма при плохой разделимости выборки на кластеры.
3. Неустойчивость алгоритма (зависимость от выбора начального объекта).
4. Произвольное по количеству разбиение на кластеры.
5. Необходимость априорных знаний о ширине (диаметре) кластеров.

С опорой на идеюпостроения кластеров в областях сгущения элементов, предлагается модифицировать алгоритм ФОРЭЛ с целью устранения вышеуказанных недостатков, следующим образом:

1. Упорядочиваем выборку по возрастанию

$X = \{x_0, x_1, ..., x_n\}$, где x_0 – минимальный объект по абсолютной величине. Тем самым мы обходим неустойчивость алгоритма путем фиксирования первого элемента для начала цикла.

2. Область сгущения элементов будем определять на основе евклидовой метрики.

$$\rho(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2}$$

3. Считаем исходное состояние в качестве набора одноэлементных кластеров

$C = X$

4. Зададим параметр Rследующим образом:сперва вычисляем расстояние от каждого элемента до последующего и вычисленные результаты запишем в массив D =

$\{d_1, \ldots, d_{n-1}\}$, где $d_i = \rho(x_i, x_{i+1})$. Обозначим d_{min} – минимальный элемент в *D*. На каждой итерации цикла *R*будет вычисляться по формуле

$$R = d_i + d_{\min}$$

5. Пока $X \neq \varnothing$:

- Выбираем начальную точку x_0.

i. Образуем кластер $c = \left\{ x \in X \middle| \rho(x_0, x) < \mathrm{R} \right\}$.

ii. Переопределяем x_0 -минимальныйэлемент множества $X \setminus c$.

iii. $X := X \setminus c$, $\mathrm{C} := \mathrm{C} \cup \{c\}$.

6. Выдать множество полученных кластеров *C*.

Относительно низкая производительность алгоритма ФОРЭЛ(из-за пересчета промежуточного центра) повышается за счет отказа использования центров кластеров как таковых.Тем самым мы заменяем «формальный элемент» в выборке на каждой итерации цикла на фиксированный параметр x_0. Так как мы основываемся на гипотезе компактности (сгущения) элементов в кластерах, то данный вариант алгоритма родственен концепции алгоритмов семейства KRAB [3].

После работы алгоритма над готовой кластеризацией можно производить следующие действия:

1. Выбор наиболее репрезентативных (представительных) объектов из каждого кластера. Можно выбрать несколько объектов из каждого кластера, учитывая априорные знания о необходимой репрезентативности выборки. Таким образом, по готовой кластеризации мы имеем возможность строить наиболее репрезентативную выборку.

2. Пересчет кластеризации (многоуровневость, проход «вглубь»). Это может быть даже необходимым действием для больших массивов данных.

Практическая задача кластеризации

В работе решается практическая задача кластеризации 670 российских банков и кредитных организаций по показателю «Вклады физических лиц» (по состоянию на март 2015 года)с помощьюалгоритма ФОРЭЛ и его модификации.Также рассматривается вопрос кластеризации по нечисловому (качественному) признаку и приводится сравнительный анализ полученных кластеров.

Список использованной литературы

1. Воронцов К.В. Лекции по алгоритмам кластеризации и многомерного шкалирования. Курс лекций. МГУ, 2007 – 18 с.

2. ЗагоруйкоН.Г., Ёлкина В.Н., Лбов Г.С. Алгоритмы обнаружения эмпирических закономерностей. — Новосибирск: Наука, 1985. — 999 с.

3. Дюран Б., Оделл П. Кластерный анализ. Пер. с анг. Е.З. Демиденко / Под ред. А.Я. Боярского. Предисловие А.Я. Боярского. - М.: Статистика, 1997.

УДК 519.7

АНАЛИЗ ОПТИМАЛЬНОСТИ ONLINE АЛГОРИТМОВ ДЛЯ ЗАДАЧИ КЭШИРОВАНИЯ

Петраков И. Е.,
научный руководитель д-р физ.-мат. наук Быкова В. В.
Сибирский федеральный университет

На практике существуют задачи, которые нужно решать, не зная заранее все входные данные. Алгоритмы, обрабатывающие вход по мере поступления, называются online алгоритмами. Эти алгоритмы составляют теоретическую основу изучения проблем интерактивных вычислений. Online алгоритмы моделируют ситуацию, когда в интерактивную систему вход поступает не целиком, а в виде последовательности частей, и система должна реагировать в ответ на каждую входящую часть. Заметим, что в любой момент времени неизвестно, какой будет следующая часть входа. Качество работы (или эффективность) online алгоритмов невозможно оценить традиционными способами через ресурсную сложность, поскольку изначально неизвестны все входные данные и длина входа алгоритма. Эффективность online алгоритмов принято оценивать с помощью конкурентного анализа [Sleator D. D, Tarjan R.E. Amortized efficiency of list update and paging rules. Communications of the ACM. 1985. 28. P. 202–208].

Формально online алгоритм принимает на вход последовательность запросов

$\sigma = \sigma(1),\ldots,\sigma(n)$.

Эти запросы должны быть обработаны в порядке появления. При получении запроса $\sigma(t)$ online алгоритм не знает запросы $\sigma(t')$ для $t' > t$. Обслуживание запросов сопровождается некоторыми затратами. Цель состоит в минимизации общей стоимости обслуживания всей последовательности запросов. В конкурентном анализе всякий конкретный online алгоритма A сравнивается с соответствующим ему offline алгоритмом B, который знает всю последовательность запросов σ заранее и может обработать ее с минимальными затратами.

Для заданной последовательности запросов σ обозначим через $\mathrm{cost}_A(\sigma)$ и $\mathrm{cost}_B(\sigma)$ расходы, понесенные A и B соответственно. Online алгоритм A называется c-конкурентным (c-competitive), если для любой последовательности σ, состоящей из некоторых частей входных данных, и любого алгоритма B, которому вся последовательность запросов дается сразу, выполняется неравенство

$$\mathrm{cost}_A(\sigma) \le c \cdot \mathrm{cost}_B(\sigma) + c_1,$$

где константы c, c_1 должны быть независимыми от последовательности запросов σ. Обычно константа c вычисляется применительно к худшему случаю (для наиболее затратной последовательности σ). Считается, что чем меньше значение константы c, тем более конкурентным является online-алгоритм A по сравнению с B.

За последние 15 лет online алгоритмы изучались во многих прикладных областях, включая управление ресурсами в операционных системах, структурирование данных, планирование, сети и финансы.

Одной из задач, для которой естественен online подход, является задача кэширования. Суть задачи кэширования заключается в следующем. Память компьютера разбита на страницы равной длины. Страницы пронумерованы, причем в кэш-память (далее просто кэш) может быть записано k страниц, а на дисковую внешнюю память (далее просто диск) t страниц. На очередном шаге поступает запрос на какую-то страницу. Если эта страница в кэше уже имеется, то достаточно сообщить

место расположения ее в кэше. В противном случае требуется решить, в какое место кэша можно ее записать и сделать это, считав страницу с диска. Стоимость расходов для алгоритмов кэширования определяется количеством обращений к диску.

Наиболее известными детерминированными online алгоритмами кэширования являются алгоритмы LRU и FIFO. Эти алгоритмы широко используются в системах ввода-вывода реальных операционных систем. Данные online алгоритмы реализуют следующие стратегии выбора места для записи запрашиваемой страницы:

– *Least Recently Used*, LRU: удаление из кэша страницы, последний запрос на которую был раньше всего; освободившее место предоставляется запрашиваемой странице;

– *First-in, First-out*, FIFO: удаление страницы, находящейся в кэше дольше всего; освободившее место предоставляется запрашиваемой странице.

Известны и другие детерминированные online алгоритмы решения задачи кэширования (MRU, LFU, ARC и др). Все эти алгоритмы принято называть политиками вытеснения (или выталкивания). Выполним конкурентный анализ алгоритмов LRU и FIFO. В качестве offline алгоритма *B* рассмотрим алгоритм, реализующий следующую стратегию: удаление из кэша страницы, которая не будет использована дольше всего.

В работе доказаны следующие утверждения.

Утверждение 1. Online алгоритм LRU является *k*-конкурентным.

Утверждение 2. Все детерминированные online алгоритмы типа LRU, FIFO решения задачи кэширования *k*-конкурентные.

Алгоритмы LRU, FIFO и *B* были реализованы в виде программ, с помощью которых были выполнены вычислительные эксперименты. В таблицах 1 и 2 представлены вычисленные значения константы *c* для различных последовательностей запросов σ длины *n* и размера кэша *k*.

Таблица 1. Значения константы *c* для LRU

n k	100	500	1000	2500	5000
3	2.77778	2.95858	2.97619	2.99043	2.99581
5	4.16667	4.80769	4.90196	4.96032	4.98008
10	5.26316	8.47458	9.17431	9.65251	9.82318
16	4.54545	10. 6383	12.8205	14.5349	15.2439
20	4.16662	11.3636	14.4928	17.3611	18.5874
30	3.0303	10.8696	15.873	22.1239	25.5102

Таблица 2. Значения константы *c* для FIFO

n k	100	500	1000	2500	5000
3	2.77778	2.95858	2.97619	2.99043	2.99581
5	4.16667	4.80769	4.90196	4.96032	4.98008
10	5.26316	8.47458	9.17431	9.65251	9.82318
16	4.54545	10. 6383	12.8205	14.5349	15.2439
20	4.16667	11.3636	14.4928	17.3611	18.5874
30	3.0303	10.8696	15.873	22.1239	25.5102

Результаты вычислительных экспериментов подтверждают, что заложенные в алгоритмах стратегии вытеснения страниц LRU, FIFO эквивалентны с точки зрения конкурентного анализа.

УДК 004.9

СТОХАСТИЧЕСКИЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ О РЮКЗАКЕ

Семёнов М.Г.,
научный руководитель канд. физ.-мат. наук, доцент Баранова И.В.
Сибирский Федеральный Университет,
Институт математики и фундаментальной информатики

Введение

Задача о рюкзаке — одна из NP-сложных задач комбинаторной оптимизации. Своё название задача получила от оптимизационной задачи укладки как можно большего числа ценных вещей в рюкзак при условии, что общий объём (или вес) всех предметов, способных поместиться в рюкзак, ограничен. Задача о рюкзаке является актуальной и достаточно востребованной с точки зрения ее приложения в реальной жизни. Задача о загрузке и её модификации часто возникают в экономике, криптографии, генетике, лингвистике и логистике для нахождения оптимальной загрузки транспорта (самолёта, поезда, трюма корабля) или склада.

Данная задача может быть решена точными, приближенными или стохастическими методами. К точным методам относятся полный перебор и метод ветвей и границ. Группа приближенных методов представлена разновидностями метода динамического программирования. Стохастическими методами являются жадный и генетический алгоритм.

Целью данной работы является изучение постановки классической задачи о рюкзаке, реализация стохастических методов решения данной задачи и сравнение этих методов. Также в данной работе решается практический пример о рюкзаке – задача нахождения оптимальной загрузки транспорта, решение которой сводится к решению классической задачи о рюкзаке.

Классическая постановка задачи о рюкзаке:

Пусть имеется набор предметов, каждый из которых имеет два параметра – вес и ценность. Имеется рюкзак с некоторым заданным значением вместимости. Задача заключается в том, чтобы собрать рюкзак с максимальной ценностью предметов внутри, соблюдая при этом весовое ограничение рюкзака.

Математически постановка задачи формулируется следующим образом: пусть имеется n предметов. Для каждого i-го предмета задан его вес $p_i > 0$ и стоимость (ценность) $c_i > 0,\ i = 1,2,\ldots,n.$ Задано ограничение на максимальный вес рюкзака – P. Каждый x_i может принимать только одно из двух значений: $x_i = 1$, если i-й предмет упаковывают в рюкзак, или $x_i = 0$, в противном случае.

Требуется выбрать из заданного множества предметов набор с максимальной суммарной стоимостью $\sum_{i=1}^{n} c_i x_i$ при одновременном соблюдении ограничения на суммарный вес найденного набора $\sum_{i=1}^{n} p_i x_i \le P.$

Методы решения задачи

В работе рассматриваются два стохастических алгоритма решения задачи о рюкзаке: жадный и генетический алгоритмы. Также проводится сравнение скорости работы данных алгоритмов с точным методом решения задачи – методом полного перебора.

Жадный алгоритм

Жадный алгоритм (greedy algorithm) всегда делает выбор, который кажется самым лучшим в данный момент — т.е. производится локально оптимальный выбор в надежде, что он приведет к оптимальному решению глобальной задачи.

Для задачи о рюкзаке жадный алгоритм имеет следующий вид:

1. Каждому предмету находится относительная стоимость единицы предмета $h_i = \frac{c_i}{p_i}$. Получается вектор $H = \{h_1, \ldots, h_n\}$

2. При помощи быстрой сортировки (англ. *quicksort*) упорядочиваются предметы по возрастанию удельной стоимости: $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_N}{w_N}$.

3. До тех пор пока набираются $\sum_{i=1}^{n} p_i x_i \leq P$ объекты с самой большой удельной стоимостью, процесс продолжается.

Генетический алгоритм

Генетические алгоритмы – это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора. Впервые понятие генетического алгоритма было предложено в 1975 г. в работах Джона Холланда. Генетические алгоритмы основаны на принципах естественного отбора Ч. Дарвина. Они относятся к стохастическим методам. Эти алгоритмы успешно применяются в различных областях деятельности (экономика, физика, технические науки и т.п.).

Перечислим основные операторы генетических алгоритмов:

Оператор селекции – отбирает популяцию для дальнейшего размножения. Заключается в том, что родителями могут стать только те особи, значение приспособленности которых не меньше пороговой величины, например, среднего значения приспособленности по популяции. Такой подход обеспечивает более быструю сходимость алгоритма.

Оператор скрещивания – служит для распространения хороших генов по популяции.

Оператор мутации – изменяет значение гена в хромосоме на любое другое возможное значение. Случайное изменение генов должно осуществляться с низкой вероятностью, обычно в пределах [0.001;0.01]. После процесса воспроизводства происходят мутации *(mutation*). Данный оператор необходим для «выбивания» популяции из локального экстремума и препятствует преждевременной сходимости. Это достигается за счет того, что изменяется случайно выбранный ген в хромосоме.

Основные принципы работы генетических алгоритмов заключены в следующей схеме:

1. генерируется начальная популяция из n хромосом (решений задачи) – обычно в качестве начальной выбирается случайная популяция множества решений,
2. вычисляется оценка качества пригодности для каждого решения (хромосомы) – обычно она пропорциональна $1/e^2$),
3. выбирается пара хромосом-родителей с помощью одного из способов *отбора*,
4. проводится *скрещивание* двух родителей с вероятностью p_c, производя двух потомков, т.е. создается новое решение на основе рекомбинации из существующих;
5. проводится *мутация* потомков с вероятностью p_m – создается новое решение на основе случайного незначительного изменения одного из существующих. Мутация

обычно происходит с вероятностью p_m для каждого гена. Хорошим эмпирическим правилом считается выбор вероятности мутации равным $p_m = 1/n$, где n - число генов в хромосоме (в среднем хотя бы один ген будет подвержен мутации),

6. повторяются шаги 3-5, пока не будет сгенерировано новое поколение популяции, содержащее n хромосом,

7. повторяются шаги 2-6, пока не будет достигнут критерий окончания процесса.

Разработанный генетический алгоритм решения задачи о рюкзаке

1. Генерируется начальная популяция из m хромосом (решений задачи); Каждая хромосома – это вектор $x^j = \{x_1^j, ..., x_n^j\}$, заполненный случайно числами $x_i^j = rand()\%2$, где j – это номер хромосомы в популяции, i – номер гена в хромосоме, m – число хромосом в начальной популяции, n – количество исходных точек. Число x_i^j, которое хранится в каждом гене хромосомы, и означает, что предмет с индексом i присутствует в наборе j.

2. Вычисляется коэффициент пригодности $Q(x^j)$ для каждой хромосомы. Оценка приспособленности особей в популяции заключается в вычислении значения функции пригодности для каждого члена популяции. Чем выше это значение, тем больше особь отвечает требованиям решаемой задачи. В качестве коэффициента пригодности используется полная цена набора. Кроме того также проверяется условие, чтобы масса набора не превышала максимальную вместимость рюкзака:

$$Q(x^j) = \sum_{i=1}^{n} c_i x_i \to \max,$$

$$\sum_{i=1}^{n} p_i x_i \le P.$$

3. Отбирается пара хромосом-родителей, у которых коэффициент пригодности выше среднего по популяции; Этап селекции предусматривает выбор тех особей, генетический материал которых будет участвовать в формировании следующей популяции решений, т.е. в создании очередного поколения. Описываемый выбор производится согласно принципу естественного отбора, благодаря которому максимальные шансы имеют особи с наибольшими значениями функции пригодности.

4. Проводится *скрещивание* двух родителей, производя потомка, т.е. создается новое решение на основе рекомбинации из существующих; Если индексы родителей p и k то результатом рекомбинации будет вектор $x = \{x_1^p, ..., x_{\frac{n}{2}}^p, x_{\frac{n}{2}+1}^k, ..., x_n^k\}$.

5. Проводится *мутация* потомка – создается новое решение на основе случайного незначительного изменения одного гена потомка;

6. Отбираются хромосомы с самым высоким коэффициентом пригодности $Q(x^j)$. Они образуют новую начальную популяцию.

7. Повторяются шаги 3-6, пока при переходе к новому поколению происходят изменения.

Практические задачи классической загрузки рюкзака и нахождения оптимальной загрузки транспорта

В работе рассматриваются два практических примера задачи о рюкзаке. Первая задача представляет собой пример классической задачи о рюкзаке. Для данной задачи задана максимальной вместимость рюкзака P = 30 и количеством предметов n = 20. Для каждого предмета были заданы его вес и стоимость.

Вторая рассматриваемая задача является практической логистической задачей – нахождение оптимальной загрузки транспорта. Для решения задачи использовалась реальная статистика стоимости и объёма товаров одной из фирм г. Красноярска, занимающейся розничной торговлей цифровой и бытовой техники.

С помощью перечисленных методов были найдены решения данных задач о рюкзаке, а также выполнено сравнение методов по скорости и точности нахождения решения. В рамках работы было разработано программное приложение, реализующее работу жадного алгоритма, метода полного перебора и разработанного генетического алгоритма. Программирование осуществлялось на языке C++.

Заключение

В работе была изучена постановки классической задачи о рюкзаке, реализованы метод полного перебора и два стохастических метода решения данной задачи: жадный и генетический алгоритм. Проведено сравнение рассмотренных методов. Также в данной работе были решены пример классической задачи о рюкзаке и практическая задача нахождения оптимальной загрузки транспорта.

Если требуется найти точное решение задачи, то необходимо использовать точный метод. Поскольку метод полного перебора очень трудоемкий, то размер рюкзака и количество предметов при его использовании должны быть ограничены. Если перебирать всевозможные подмножества данного набора из n предметов, то получится решение сложности не менее чем $O(2^n)$.

Стохастические методы работают значительно быстрее точных методов, но они могут не находить точное решение. Генетический алгоритм является одним из самых быстрых алгоритмов. Недостатком этого алгоритма является то, что в некоторых ситуациях он находит локальный экстремум вместо глобального. Происходит это из-за того, что алгоритм может заканчиваться не только при достижении оптимального решения, но и следующих условиях:

- пройдено максимальное заданное число итераций;
- прошло максимальное время, заданное для выполнения алгоритма;
- при переходе к новому поколению не происходит существенных изменений.

Список литературы

1. Быкова, В. В. Дискретная математика с использованием ЭВМ: учебное пособие / В. В. Быкова. – Красноярск: РИО КрасГУ, 2006. – 200 с.

2. Кормен, Т. Х. Алгоритмы: построение и анализ / Пер. с англ. под ред. А. Шеня. / Т. Х. кормен, Ч. И. Лейзерсон, Р.Л. Ривест. − М.: Вильямс, 2005. − 1296 с.

3. Панченко, Т. В. Генетические алгоритмы / Т. В. Панченко, под ред. Ю. Ю. Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. − 87 с.

4. Пападимитриу, Х. Х. Комбинаторная оптимизация. Алгоритмы и сложность / Х.Х. Пападимитриу, К. Стайглиц, пер. с англ. В. Б. Алексеева. – М.: Мир, 1985. – 512 с.

УДК 519.7

ПРОГРАММНЫЕ СРЕДСТВА ПРОВЕРКИ НЕКОТОРЫХ КРИПТОГРАФИЧЕСКИХ СВОЙСТВ БУЛЕВЫХ ФУНКЦИЙ

Семенова Е. В.
научный руководитель д-р физ.-мат. наук Быкова В. В.
Сибирский федеральный университет

Булевы функции широко используются в криптографии, в частности, при построении поточных шифров в качестве комбинирующих и фильтрующих функций в генераторах ключевого потока, а также для блочных шифров в качестве функций блоков замены [3]. Для обеспечения стойкости этих шифров булевы функции должны обладать рядом свойств. Многие из этих свойств противоречат друг к другу. Поэтому разработчику шифров постоянно приходится искать компромисс между ними.

В настоящей работе представлены программные средства, позволяющие вычислять основные числовые характеристики булевых функций, находить алгебраическую нормальную форму, выполнять проверку основных свойств булевых функций: аффинность, сбалансированность, устойчивость, нелинейность.

Введем следующие обозначения:

n – некоторое натуральное число;

$\mathbb{Z}_2$ – множество, состоящее из 0 и 1;

$x = (x_1, \ldots, x_n)$ – двоичный вектор с координатами из $\mathbb{Z}_2$;

$\mathbb{Z}_2^n$ – множество всех двоичных векторов длины n;

$f(x) = f(x_1, \ldots, x_n)$ – булева функция от n переменных, то есть отображение из множества $\mathbb{Z}_2^n$ во множество $\mathbb{Z}_2$.

Один из способов представления булевой функции – это описание булевой функции с помощью операций умножения и сложения по модулю 2, а также констант 0 и 1, в следующем виде:

$$f(x_1,\ldots,x_n) = \bigoplus_{k=1}^{n} \bigoplus_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k} x_{i_1} \ldots x_{i_k} \oplus a_0, \qquad (1)$$

$$\{i_1,\ldots,i_k\} \subseteq \{1,\ldots,n\}.$$

Считается, что все коэффициенты в (1) принадлежат $\mathbb{Z}_2$. Представление (1) называется алгебраической нормальной формой (АНФ) или полиномом Жегалкина булевой функции $f(x)$. Для всякой булевой функции оно единственное [1, 5].

Булева функция от n переменных вида

$$a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \ldots \oplus a_n x_n$$

называется аффинной, а при $a_0 = 0$ линейной. Ясно, что всякая линейная булева функция всегда является аффинной.

Алгебраической степенью булевой функции $f(x)$ называется число переменных в самом длинном слагаемом ее АНФ и обозначается через $deg(f)$. Булевы функции с высокой алгебраической степенью используются для построения блочных шифров: чем выше алгебраическая степень применяемой булевой функции, тем более надежен шифр. Для того чтобы найти алгебраическую степень булевой функции $f(x)$, а также проверить, является ли функция аффинной или линейной необходимо построить АНФ для $f(x)$.

Известно [2], что каждую булеву функцию $f(x)$ от n переменных можно однозначно определить вектором ее значений длины 2^n. При этом важен порядок, в котором перечисляются эти значения. Обычно значения булевой функции в векторе значений

перечисляются согласно лексикографическому порядку векторов длины n, соответствующих значениям ее аргументов.

Весом Хэмминга $wt(f)$ булевой функции $f(x)$ называется число ее единичных значений. Под расстоянием Хэмминга $dist(f, g)$ между двумя булевыми функциями $f(x)$ и $g(x)$ от n переменных понимается величина:

$dist(f, g) = wt(f \oplus g)$,

которая равна числу координат, в которых различаются векторы значений функций $f(x)$ и $g(x)$.

Расстояние Хэмминга между функцией $f(x)$ от n переменных и произвольным множеством M_n булевых функций от n переменных определяется следующим образом:

$$dist(f, M_n) = \min\{dist(f, g) | g \in M_n\}.$$

Нелинейность N_f булевой функции $f(x)$ от n переменных – это расстояние Хэмминга от $f(x)$ до множества A_n всех аффинных функций от n переменных

$$N_f = dist(f, A_n).$$

Чем выше нелинейность функции, используемой в качестве компоненты шифра, тем выше стойкость шифра к линейному криптоанализу. Суть линейного криптоанализа состоит в замене сложной булевой функции, описывающей некоторые нелинейные преобразования шифра, простой линейной функцией. При этом получается не исходный шифр, а его приближение, которое легче поддается анализу. Таким образом, чем выше нелинейность булевой функции, тем сильнее функция отличается от любой линейной функции и тем «неудачней» будет подобная замена, предложенная криптоаналитиком. Высокая нелинейность функции является одним из ее основных криптографических свойств [3]. Нелинейность произвольной булевой функции $f(x)$ от n переменных вычисляется по формуле [4]:

$$N_f = 2^{n-1} - \frac{1}{2}\max_y |W_f(y)|, \qquad (2)$$

где $W_f(y)$ – это коэффициенты Уолша – Адамара.

Преобразованием Уолша – Адамара булевой функции $f(x)$ от n переменных называется целочисленная функция $W_f(y)$, определяемая на множестве $\mathbb{Z}_2^n$ следующим образом:

$$W_f(y) = \sum (-1)^{\langle x, y\rangle \oplus f(x)}, \qquad (3)$$

где $x \in \mathbb{Z}_2^n$. Для каждого конкретного $y \in \mathbb{Z}_2^n$ величина $W_f(y)$, вычисленная по формуле (3), называется коэффициентом Уолша – Адамара булевой функции $f(x)$. Набор чисел $W_f(y)$ по всем $y \in \mathbb{Z}_2^n$ называется спектром Уолша – Адамара функции $f(x)$. При этом считается, что векторы y перебираются в лексикографическом порядке. Очевидно, что спектр булевой функции $f(x)$ от n переменных содержит 2^n целых (необязательно положительных) чисел. Известно, что булева функция однозначно восстанавливается по своему спектру [3, 4]. Формула (2) указывает способ вычисления показателя нелинейности N_f булевой функции $f(x)$ через ее спектр. Другими словами, для нахождения N_f достаточно вычислить все коэффициенты Уолша – Адамара и найти среди них максимальный по модулю.

Булева функция называется сбалансированной, если она принимает значение 0 и 1 одинаково часто. Для сбалансированной функции $f(x)$ от n переменных всегда $wt(f) = 2^{n-1}$. Таким образом, сбалансированность легко вычисляется, исходя из вектора значений булевой функции: если $wt(f) = 2^{n-1}$, то $f(x)$ сбалансированная, иначе несбалансированная. Свойство сбалансированности применяется в статистических методах криптоанализа.

Более сильным (по сравнения со сбалансированностью) свойством булевой функции является r-устойчивость. Пусть r –целое число и $0 \le r \le n-1$. Булева функция $f(x)$ от n переменных называется r-устойчивой, если любая ее подфункция, полученная фиксацией не более r переменных, является сбалансированной. Булевы функции, являющиеся r-устойчивыми, используются в корреляционном криптоанализе. Проверка данного свойства для функции $f(x)$ от n переменных сводится к генерации различных вариантов фиксации $0 \le k \le r$ переменных и проверки свойства сбалансированности для полученной подфункции.

Между свойствами нелинейности, сбалансированности и устойчивости имеются определенные противоречия. Так, бент-функция (максимально нелинейная булева функция от четного числа переменных) не может быть сбалансированной и устойчивой [3, 4]. Для исследования рассмотренных криптографических свойств булевых функций создан комплекс программ BUL_FUNC.

Комплекс программ BUL_FUNC разработан в среде Microsoft Visual Studio 2010 Express на языке программирования C++. Комплекс программ BUL_FUNC реализует следующие функции:

– нахождение коэффициентов АНФ;

– вычисление коэффициентов Уолша – Адамара и чисел $deg(f)$, $wt(f)$, N_f;

– проверка свойств аффинности, линейности, сбалансированности и r-устойчивости булевой функции.

Основными входными данными являются число переменных и вектор значений исходной булевой функции $f(x)$. Вычисление АНФ выполняется методом неопределенных коэффициентов [2, 4].

Комплекс программ BUL_FUNC также позволяет

– по заданным коэффициентам АНФ находить вектор значений булевой функции;

– по заданному вектору значений формировать СДНФ и СКНФ;

– проверять классические свойства булевой функции, такие как монотонность, самодвойственность, сохранение 0 и 1;

– вычислять расстояние Хэмминга между двумя заданными булевыми функциями.

Разработанный комплекс программ BUL_FUNC может быть использован в криптоанализе для исследования свойств булевых функций, а также в учебном процессе в средних и высших учебных заведениях при изучении булевых функций.

Список литературы

1 Гаврилов Г. П., Сапоженко А. А. Задачи и упражнения по дискретной математике. – М.: ФИЗМАТЛИТ, 2005.

2 Новиков Ф. А. Дискретная математика для программистов. – СПб: Питер, 2000.

3 Панкратова И. А. Булевы функции в криптографии: учебное пособие. – Томск: Томский гос. ун-т, 2014.

4 Токарева Н. Н. Симметричная криптография. Краткий курс: учебное пособие. – Новосибирск: Новосиб. гос. ун-т, 2012.

5 Яблонский С. В. Введение в дискретную математику. – М.: Наука, 1986.

УДК 004.9

АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ ФИЛЬТРАЦИИ СПАМА

Сизиков Г.В.,
научный руководитель канд. физ.-мат. наук, доцент Баранова И.В.
Сибирский Федеральный Университет,
Институт математики и фундаментальной информатики

1. Введение

Одним из самых востребованных направлений в области защиты информации является разработка методов фильтрации потока электронной почты. В последнее время электронная почта стала одним из наиболее распространенных средств связи, управления и бизнеса. Она является достаточно совершенной в техническом отношении и недорогой альтернативой привычным средствам связи.

Вместе с развитием электронной почты увеличивается и количество угроз ее нормальному функционированию. Наиболее серьезной и важной проблемой стал так называемый спам, то есть нежелательные массовые рассылки сообщений, в основном рекламного характера.В 2014 году доля спама в мировом почтовом трафике, по данным «Лаборатории Касперского», снизилась и составила 66,8%, что почти на три процентных пункта меньше, чем в 2013 году. Среди стран-источников спама по-прежнему лидируют США (16,7%), за которыми с существенным отрывом следует Россия (6%).

Среди вредоносных вложений чаще всего встречалась фишинговаяhtml-страница с формой для ввода конфиденциальных данных, которые затем направлялись злоумышленникам. На втором месте – червь Bagle, собирающий почтовые контакты жертвы и далее рассылающий себя по ним, а за ним – троянец Redirector, перенаправляющий пользователя на мошеннический сайт. При этом наибольшая доля срабатываний почтового антивируса зарегистрирована в США (9,8%), за которыми следуют Великобритания (9,6%) и Германия (9,2%).

Целью данной работы является изучение различных методов фильтрации спама, реализация основных методов решения данной задачи, классификация и сравнение этих методов.

2. Основные подходы к решению задачи фильтрации спама

На сегодняшний день разработан ряд технологий построения фильтров спама – специальных сервисов для отсеивания нежелательной корреспонденции. Все технологии можно разделить на настраиваемые вручную и интеллектуальные (автоматические). Настраиваемые вручную фильтры основываются на списках доступа и настраиваются непосредственно пользователем, который задаёт либо нежелательные адреса, при политике пропуска по «черному списку», либо разрешенные адреса, при политике пропуска по «белому списку». Однако ручные способы фильтрации нежелательных сообщений малоэффективны и требуют постоянного обновления списков доступа, создавая дополнительную нагрузку на пользователя.

Решения, которые в той или иной мере могут помочь снять проблему спама, можно условно разделить на следующие группы:

- Простейшие способы ручной или автоматической фильтрации почты по заголовкам.

Любой пользователь может перейти с протокола POP3 на IMAP4 или на Web-интерфейс и оценивать электронные письма только по их заголовкам, не получая текста. Во многих почтовых программах можно настроить автоматическую фильтрацию спама по заголовкам писем. Однако в последнем случае требуется очень

тонкая и трудоёмкая подгонка условий оценки, а также внимательность к оформлению сообщений от ваших постоянных корреспондентов.

- Специальные службы фильтрации спама, которые могут находиться у почтового провайдера или на отдельном сервере (последние, как правило, платные).

Этот способ является более надежным, чем предыдущий. В некоторых случаях вся почта отправляется на определенный адрес, где фильтруется, и к пользователю приходит уже очищенной. Данный метод с точки зрения трудозатрат пользователя является самый простым, но, как правило, и наименее контролируемым (велика вероятность, что может потеряться часть полезной корреспонденции, о чем никто никогда не узнает).

- Входные антиспам-фильтры, основанные на анализе IP-адреса хоста, передающего спам (который можно узнать, например, по отзывам пострадавших), и использование общих баз данных с адресами таких спамеров (DNSBL — DNS BlackLists).

Однако в данный момент этот способ борьбы с современными методами спама является малоэффективным, вследствие частого обновления адресов спамеров.

- Фильтрация на основе автоматического пополнения access-листа адресами спамеров.
- Программы или встраиваемые модули для анализа содержимого письма.

Программы для такой проверки (их может быть несколько) принимают информацию от почтовой программы, а возвращают, как правило, свою оценку и рекомендацию к дальнейшему действию.

Целью данной работы является изучение и реализация алгоритмов фильтрации спама, относящихся к последней группе. В работе рассматриваются следующие методы фильтрации спама: Байесовский метод и классификация на основе сравнения с эталоном.

3. Математическая постановка задачи

Фильтрация спама является разновидностью задачи классификации текстов, поэтому будем использовать следующую модель задачи классификации.

Ω – множество объектов распознавания (пространство образов).

$\omega : \Omega \in W$ – объект распознавания (образ).

$g(\omega) : \Omega \to M$, $M = \{1,2,...,m\}$ – индикаторная функция, разбивающая пространство образов на W на m непересекающихся классов$\Omega^1, \Omega^2 \dots \Omega^m$. Индикаторная функция неизвестна наблюдателю.

X – пространство наблюдений, воспринимаемых наблюдателем (пространство признаков).

$x(\omega) : \Omega \to X$ – функция, ставящая в соответствие каждому объекту ω точку x(ω) в пространстве признаков.

Вектор x(ω) - это образ объекта, воспринимаемый наблюдателем.

В пространстве признаков определены непересекающиеся множества точек $K_i \subset X$, i = 1,2...,m , соответствующих образам одного класса.

$\hat{g}(x) : X \to M$ – решающее правило – оценка для g(ω) на основании x(ω) , т.е. $\hat{g}(x) = \hat{g}(x(\omega))$.

Пусть$x_j = x(\omega_j)$, j = 1,2...,N – доступная наблюдателю информация о функциях g(ω) и x(ω) , но сами эти функции наблюдателю неизвестны. Тогда (g_j, x_j), j=1,2...,N – есть множество прецедентов.

Задача заключается в построении такого решающего правила $\hat{g}(x)$, чтобы распознавание проводилось с минимальным числом ошибок. Обычный случай – считать пространство признаков евклидовым, т.е.$X = R^1$. Качество решающего правила

измеряют частотой появления правильных решений. Обычно его оценивают, наделяя множество объектов W некоторой вероятностной мерой. Тогда задача записывается в виде $\min P\{g(x(w)) \neq g(w)\}$) .

4. Алгоритмы фильтрации спама

4.1. Метод Байеса. Характеристика и описание

Среди программ, предназначенных для борьбы со спамом, особенно интересны те, что работают по принципам Байеса и самообучаются в процессе анализа корреспонденции. Данная технология отличается использованием байесовских принципов для распознавания спама по образу, моделирование которого происходит благодаря анализу самого спама.

Однако простота применения байесовских принципов обманчива, так как отнесение письма к спаму производится по сложным алгоритмам выявления общих элементов в реальных посланиях. Таким образом, чем большее количество спама подверглось анализу, тем лучше работает фильтр. Кроме того, метод Байеса обладает автокоррекцией, поскольку в случае изменения структуры писем фильтр изменяется автоматически.

При обучении антиспам-фильтра по методу Байеса для каждого встреченного в письмах слова высчитывается и сохраняется его «вес» — вероятность того, что письмо с этим словом является спамом.

Отнесение письма к «спаму» или к обычной корреспонденции производится по тому, превышает ли его «вес» некую планку, заданную пользователем (обычно берут 60-80%). После принятия решения по письму в базе данных обновляются «веса» для вошедших в него слов.

Алгоритмы данного метода фильтрации спама элементарны, он удобен, достаточно эффективен (при условии обучения на достаточно большом количестве писем блокирует до 95-97% спама) и обучаем. На основе данного метода функционирует большинство современных спам-фильтров, установленных как на почтовых серверах, так и встроенных в почтовое программное обеспечение пользователя.

Однако у метода есть и принципиальные недостатки: во-первых, он базируется на предположении, что одни слова чаще встречаются в спаме, а другие— в обычных письмах, и неэффективен, если данное предположение неверно; во-вторых, данный метод фильтрации спама работает только с текстом, что позволяет спамерам обходить его, включая рекламную информацию не в тело письма, а в графическое вложение, сопровождая само письмо либо бессмысленным, либо нейтральным текстом.

4.2. Классификация на основе сравнения с эталоном

Пусть задано множество образов(эталонов). Задача состоит в том, чтобы для тестируемого объекта выяснить, какой эталон ближе на основе меры сходства(расстояниямежду объектами). Данная задача и получила название“сравнение с эталонами”.

В качестве эталонов могут рассматриваться следующие объекты:

1) Буквы в словах рукописного текста(применительно к распознаванию рукописного текста);

2) Силуэты объектов в сцене(применительно к машинному зрению);

3) Слова(команды), произносимые человеком(применительно к распознаванию речи).

В этих примерах признаки не выделены, но можно измерить сходство. Например, сравнение слов: *кошка ~ мошка ~ кора ~ норка*и.т.д. Или силуэт объекта в сцене, чье положение и ориентация заранее не известны(применительно к машинному зрению, робототехнике).

Рассмотрим строчный образ(слово). В данном случае можно выделить два критерия, на основе которых можно строить меру близости:

- Совпадение букв
- Монотонность (порядок совпадения букв)

Пусть $r_1, r_2, \dots r_I$-эталон,$t_1, t_2, \dots t_J$-пробный образ, причем $I \neq J$.

Построим соответствие между эталоном и пробным образом последующему правилу: каждому символу в первом слове должен соответствовать хотя бы один символ во втором слове и каждому символу во втором слове должен соответствовать хотя бы один символ в первом слове,(но соответствие между символами не взаимно-однозначное, в частности, поскольку $I \neq J$).

Введем меру следующим образом:

$$\rho(r_i, t_i) = \begin{cases} 1, r_i \neq t_i \\ 0, r_i = t_i \end{cases}$$

В качестве сходства меры двух слов принимаем соответствие, при котором суммарный вес всех дуг минимален:

$$v(\overline{r}, \overline{t}) = \min_S \mu(S), \text{где } \mu(S) = \sum_{(i,j)\in S} \rho(r_i, t_j)$$

Через $v(\overline{r}, \overline{t})$ будем обозначать меру близости двух слов $\overline{r}$ и $\overline{t}$. Соответствие Sдолжно быть двудольным графом без изолированных вершин с непересикающимися ребрами.

Список литературы

1.Местецкий Леонид Моисеевич Математические методы распознавания образов.–Москва: МГУ, 2002–2004

2.А.С. Потапов Распознавание образов и машинное восприятие, М.: "Политехника", 2007 - 552 с.

3.Загоруйко, Н.Г. Прикладные методы анализа данных и знаний. – Новосибирск: ИМ СО РАН, 1999. – 270 с.

УДК 519.178

АДАПТИВНОЕ РАЗМЕЩЕНИЕ ОРИЕНТИРОВ В ЗАДАЧЕ О КРАТЧАЙШЕМ ПУТИ В ГРАФЕ БОЛЬШОЙ РАЗМЕРНОСТИ

Солдатенко А. А.,
научный руководитель д-р физ.-мат. наук, Быкова В. В.
Сибирский федеральный университет

Задача поиска кратчайшего пути в графе (Shortest-Paths, SP) хорошо известная задача теории графов, имеющая многие реальные приложения, такие как планирование маршрута в веб структурах, навигационные системы, моделирование трафика, логистическая оптимизация. Задача SP состоит в поиске кратчайшего пути в заданном графе между двумя его вершинами s и t (стартовой и целевой соответственно), в которой минимизируется сумма длин рёбер, составляющих (s, t)-путь [1].

Существует много классических алгоритмов решения задачи SP, самый известный из них – алгоритм Дейкстры, который работает посредством приписывания меток вершинам и равномерного расширения пространства поиска решения, начиная от стартовой вершины и далее последовательно до целевой вершины графа. Алгоритм A* добавляет в алгоритм Дейкстры потенциальную функцию, которая оценивает длину пути от текущей вершины до целевой вершины [2]. Все эти алгоритмы выполнимы за полиномиальное время.

Во многих случаях требуется вычислить за несколько секунд кратчайший (s, t)-путь в графе большой размерности. Типичным примером является поиск кратчайшего маршрута в дорожной сети, насчитывающей несколько десятков тысяч узлов. В этих условиях классические алгоритмы становятся неприемлемыми. Чтобы справиться с этой проблемой был использован двухфазный подход решения задачи SP, который содержит фазу предобработки и фазу выполнения (s, t)-запроса – запроса на вычисление (s, t)-пути в исходном графе. На фазе предобработки выполняется просмотр графа с целью извлечения информации, позволяющей ускорить классические алгоритмы решения задачи. На второй фазе выполняется (s, t)-запрос с применением извлечённой на первой фазе информации.

Известно много различных вариантов реализации двухфазного подхода на основе алгоритма Дейкстры: ALT, Arc-Flags, SHARC и другие. Алгоритм ALT (A* with Landmarks and Triangle) представляет собой версию алгоритма A*, главная идея которого заключается в применении ориентиров и неравенства треугольника для определения допустимой и преемственной потенциальной функции [3]. Предполагается, что входной граф $G = (V, E)$ является взвешенным и связным, а неотрицательная вещественнозначная функция весов ребер удовлетворяет неравенству треугольника.

На фазе предобработки вначале выбирается множество $L \subseteq V$ вершин графа, в которых будут установлены ориентиры, а затем для каждой вершины $v \in V$ и всякого ориентира $l \in L$ значение потенциальной функции вычисляется по формулам:

$$\pi_{l+}(v) = d(v, l) - d(t, l),$$
$$\pi_{l-}(v) = d(l, t) - d(l, v),$$
$$\pi_L(v) = \max \{\pi_{l+}(v), \pi_{l-}(v) : l \in L\}.$$

Потенциальная функция $\pi_L(v)$ должна быть задана или вычислена для каждой вершины v графа G. Значение $\pi_L(v)$ определяет потенциал вершины v, как нижнюю оценку кратчайшего пути от вершины v до целевой вершины t

$$d(v, t) \geq \pi_L(v).$$

Такая потенциальная функция называется *допустимой*. Допустимая потенциальная функция не переоценивает значение кратчайшего (v, t)-пути. На потенциальную функцию накладывается еще одно ограничение: она должна быть *преемственной* (или монотонной), что подразумевает справедливость неравенства

$d(u, v) + \pi_L(v) \geq \pi_L(u)$

для любых вершин $u, v \in V$. Расстановка ориентиров в вершинах графа – существенный фактор в алгоритме ALT.

Сформулируем задачу оптимального размещения ориентиров (далее коротко MinALT). Пусть задан связный граф $G = (V, E)$ с неотрицательной вещественнозначной функцией весов $w: E \to R^+$, удовлетворяющей неравенству треугольника, и целое положительно число k. Пусть $V_L(s, t)$ – пространство поиска алгоритма Дейкстры для (s, t)-запроса и некоторого множества ориентиров $L \subset V$. Требуется найти множество $L \subset V$ такое, что

$$\Sigma_{s,t \in V} |V_L(s, t)| \to \min, \quad (1)$$

$$|L| \leq k, \quad (2)$$

где суммирование в (1) берется для фиксированной последовательности (s, t)-запросов σ. Под пространством поиска (s, t)-запроса понимается множество вершин, просмотренное алгоритмом Дейкстры во время выполнения (s, t)-запроса.

Укажем характерные особенности задачи MinALT (1)–(2):

– порядок перечисления запросов в σ неважен;

– последовательность σ в худшем случае может включать в себя все различные $n(n-1)/2$ пар вершин графа G, где $n = |V|$;

– допустимым решением задачи является всякое множество $L \subset V$ такое, что $|L| \leq k$;

– целевая функция (1) определена неявно;

– значение целевой функции вычисляется для всякого варианта выбора L и последовательности σ.

Таким образом, задача MinALT носит комбинаторный характер, и ее решение всегда может быть найдено переборным алгоритмом. Однако такой способ решения не целесообразен для практического применения, так как из-за неявного задания функции (1) требуется многократное выполнение алгоритма Дейкстры с целью определения $V_L(s, t)$ для каждого из возможных вариантов выбора множества $L \subset V$ ориентиров. Кроме того для SP в режиме online заведомо неизвестна последовательность возможных запросов σ. В этом случае разумно использовать эвристические алгоритмы поиска приближённого решения задачи MinALT. Именно такой подход наиболее популярен в настоящее время.

Основные эвристики выбора ориентиров на фазе предобработки алгоритма ALT для решения задачи SP [3]:

– H_1 случайный выбор ориентиров;

– H_2 каждый следующий ориентир выбирается как можно дальше от предыдущих;

– H_3 выбор ориентиров за пределами исследуемых областей поиска.

Все эти эвристики выполняются за полиномиальное время, и в этом их достоинство. Основной недостаток указанных стратегий: в них не используется информация, которая накапливается в процессе выполнения (s, t)-запросов. В данной работе предлагается эвристика H_4, которая в определенной степени устраняет этот недостаток. Эвристика H_4 является некоторой комбинацией и модификацией эвристик H_1 и H_2.

Отличие эвристики H_4 от известных эвристик H_1 и H_2 состоит в следующем:

– расстановка ориентиров проводится не только перед первым (s, t)-запросом, но и перед каждым n-м (s, t)-запросом;

– по ходу выполнения запросов формируется архив, в котором накапливается статистика о текущих ориентирах и вершинах, имеющих временные метки алгоритма Дейкстры.

Перед первым запросом ориентиры выбираются случайным образом. После каждого (s, t)-запроса архив обновляется путем добавления вершин, получивших временные метки, если эти вершины ещё не были просмотрены предыдущими (s, t)-запросами, и удалением вершин, которые изменили свои временные метки на постоянные. Также при выполнении (s, t)-запроса собирается статистика эффективности ориентиров, то есть всякий раз, когда ориентир предлагает наилучшую оценку пути среди всего набора ориентиров он получает очко. Как только выполнится (n – 1)-й запрос, начинается обновление ориентиров. Ориентир с наименьшим количеством очков подлежит замене на вершину из архива, которая дальше всего отстоит от текущего набора ориентиров (за исключением заменяемого ориентира). После обновления ориентиров статистика начинается собираться заново.

Представленный алгоритм ALT с эвристикой H_4 позволяет находить кратчайший путь между заданной парой вершин в графах большой размерности (состоящих из нескольких тысяч вершин) за реальное время. Эффективность алгоритма достигается за счёт хранения информации, получаемой при каждом запросе, и ее использования при расстановке ориентиров. Результаты вычислительных экспериментов показали, что для графов большой размерности достаточно устанавливать 9 – 16 ориентиров, а обновлять ориентиры каждые 15 – 30 запросов. Наибольший эффект эвристики H_4 достигается для евклидовой постановки задачи SP, то есть когда вершины исходного графа заданы координатами точек на плоскости. В этом случае значения потенциальной функции вычисляются через евклидово расстояние.

Список литературы

1. Лекции по теории графов / В. А. Емеличев, О. И. Мельников, В. И. Сарванов, Р. И. Тышкевич. – Москва, 1990. – 384с.

2. Рассел С. Искусственный интеллект: современный подход / С. Рассел, П. Норвинг – Изд. 2-е – Издательский дом «Вильямс», 2006. – 1408 с.

3. Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: *A** search meets graph theory // In SODA-2005, SIAM, 2005.– P. 156 – 165.

УДК 519.178

О НЕКОТОРЫХ «НОВЫХ» ОПЕРАЦИЯХ НАД МНОЖЕСТВОМ СОБЫТИЙ ИЛИ МНОЖЕСТВОМ ЛОГИЧЕСКИХ ВЫСКАЗЫВАНИЙ

Шведова Е.С.
научный руководитель д-р физ.-мат. наук, профессор Воробьев О.Ю.
Сибирский федеральный университет

Аннотация. *Обсуждаются некоторые необычные операции над множеством событий или множеством логических высказываний. На основе аналогий между булевыми алгебрами множеств, событий и высказываний предлагаются некоторые «новые» в эвентологическойтеории [1]бинарные и тернарные операции над множеством событий, названные «общая импликация событий» и «новые» в математической логике бинарные и тернарные операции нал множеством логических высказываний, названные «сет-чердак», «сет-подвал» и «сет-слой». Предложено «обще-импликационное» представление террасных распределений 1-го и 4-го рода дуплета и триплета событий.*

Ключевые слова. *Эвентология, булева алгебра, вероятность, событие, высказывание, теоретико-множественная операция, логическая операция, сет-чердак, сет-подвал, сет-слой, общая импликация, обще-импликационное представление.*

Булевы алгебры множеств, событий и высказываний. *Булева алгебра (БА)* – это непустое множество $\mathfrak{B}$ с заданными на нем двуместными операциями $\wedge$ (аналог пересечения и конъюнкции), $\vee$ (аналог объединения и дизъюнкции), одноместной операцией $\neg$ (аналог дополнения и отрицания) и двумя выделенными элементами 0 и 1 (аналоги пустого и «всего» множества; невозможного и достоверного события; лжи и истины), такими, что для всех элементов $x, y, z \in \mathfrak{B}$ выполняются следующие пять аксиом:

I	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$ $x \vee (y \vee z) = (x \vee y) \vee z$	ассоциативность
II	$x \wedge y = y \wedge x$ $x \vee y = y \vee x$	коммутативность
II I	$x \wedge (x \vee y) = x$ $x \vee (x \vee y) = x$	поглощение
I V	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	дистрибутивность
V	$x \wedge \neg x = 0$ $x \vee \neg x = 1$	дополнительность

Таблица 1.Аксиомы, определяющие булеву алгебру.

Аналоги между булевыми алгебрами. Рассматриваются аналоги между *БА множеств, БА событий* и *БА высказываний,* основанные на теореме Стоуна [2] в ее конечном варианте. *Множество всех подмножеств некоторого множества D*является булевой алгеброй относительно операций пересечения,объединения,дополнения. Роль нуляв такой алгебре играет пустое множество, роль единицы – все множество *D*.Если переменные логики высказываний интерпретироватьэлементамикакой-либобулевойалгебры, а связки конъюнкцию, дизъюнкцию, отрицание, ложь и истину интерпретировать соответствующими операциями на этой булевой алгебре, то все тавтологии будут интерпретироваться как 1. Более того, если некоторая формула во всех булевых алгебрах принимает значение 1, то она является тавтологией. Таким образом,

логика высказываний полна относительно булевых алгебр. *Всякая конечная булева алгебра изоморфна булевой алгебре всех подмножеств некоторого конечного множества* [2, 1936], [3, 1935].

Террасное представление произвольной операции над множеством событий.Рассматривается террасное представление [4] произвольных (теоретико-множественных, событийных или логических) $\mathcal{O}$-операций

$$\begin{matrix}\mathcal{O}\\ x\in\mathfrak{X}\end{matrix}\quad x=\sum_{X\in\mathcal{O}} ter(X//\mathfrak{X})$$

над множеством событий (м.с.) $\mathfrak{X}$,которые характеризуются генератором $\mathcal{O}\subseteq 2^{\mathfrak{X}}$. В частности, рассматриваются необычные «мощностные» операции, определяемые мощностным индикатором множества событий: *сет-слой, сет-чердак* и *сет-подвал* м.с. [1], которые являются $\mathcal{O}$-операциями при различных значениях генератора $\mathcal{O}\subseteq 2^{\mathfrak{X}}$.Для n-слоя:$\mathcal{O}=\{X\subseteq\mathfrak{X}\colon |X|=n\}$,для$n$-чердака:$\mathcal{O}=\{X\subseteq\mathfrak{X}\colon |X|\geq n\}$,для n-подвала:$\mathcal{O}=\{X\subseteq\mathfrak{X}\colon |X|\geq N+1-n\}$, где $n=0,\dots,|\mathfrak{X}|$.

Сравнение различных интерпретаций БА. Сравниваются различные интерпретации БА: теоретико-множественная, событийная и логико-высказывательная на дуплетах и триплетах событий (см. Рис. 1 и 2).Рисунок 1: Диаграммы Венна (1 столбец), 12 (из 2^3) «мощностных» тернарных теоретико-множественных операций (2 столбец) и соответствующие им «мощностные» тернарные логические функции (3 столбец) на триплете со́-бытий$\mathfrak{X}=\{x_1,x_2,x_3\}$, которые определяются мощностными индикатором $\Upsilon_{\mathfrak{X}}$ триплета $\mathfrak{X}$ на пространстве всеобщих элементарных исходов Ω.Рисунок 2: Диаграммы Венна (1 столбец), генераторы $\mathcal{O}$-операций (2 столбец), 16 бинарных теоретико-множественных операций (3 столбец) и соответствующие им бинарные логические функции (4 столбец.

«Новая» бинарная операция: *общая импликация двух событий.*Предлагается «новая» бинарная *импликациисобытий (общая импликации)*, которая определяется для событий x иy как событие$x\to y=x^c\cup y$,и с помощью которой террасное распределение 4-го рода дуплета событий $\{x,y\}\{Ter(X//\{x,y\}),X\subseteq\{x,y\}\}$может быть представлено в следующем виде:

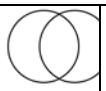
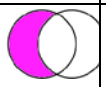
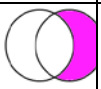
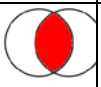
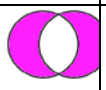
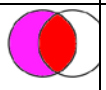
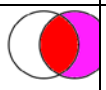
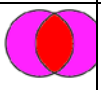
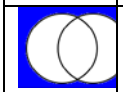
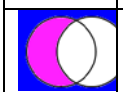
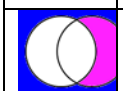
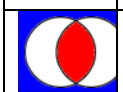
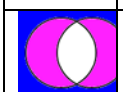
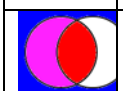
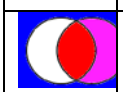
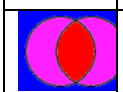
$x\to y^c=y\to x^c=x^c\cup y^c=Ter(\emptyset)$,
$x\to y=y^c\to x^c=x^c\cup y=Ter(\{y\})$,
$y\to x=x^c\to y^c=x\cup y^c=Ter(\{x\})$,
$x^c\to y=y^c\to x=x\cup y=Ter(\{x,y\})$,
где использована аббревиатура $Ter(X)=Ter(X//\{x,y\})$,когда $X\subseteq\{x,y\}$.

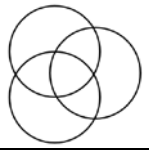
«Новая» тернарная операция: *общая импликация трех событий.* Предлагаются «новые» тернарные операции: *импликация (общая импликация) событий*, которая определяется для событий x,y и z как событие

$$(x\cap y)\to z=x^c\cup y^c\cup z,\quad x\to(y\cup z)=x^c\cup y\cup z.$$

и с помощью которой террасное распределение 4-го рода $\{Ter(X//\{x,y,z\}),X\subseteq\{x,y,z\}\}$ триплета событий $\{x,y,z\}$может быть представлено в следующем виде (см. рис. 3):

$x\cap y\cap z\to\emptyset$	$=\Omega\to x^c\cup y^c\cup z^c$	$=x^c\cup y^c\cup z^c$	$=Ter(\emptyset)$,
$y\cap z\to x$	$=x^c\to y^c\cup z^c$	$=x\cup y^c\cup z^c$	$=Ter(x)$,
$x\cap z\to y$	$=y^c\to x^c\cup z^c$	$=x^c\cup y\cup z^c$	$=Ter(y)$,
$x\cap y\to z$	$=z^c\to x^c\cup y^c$	$=x^c\cup y^c\cup z$	$=Ter(z)$,
$z\to x\cup y$	$=x^c\cup y^c\to z^c$	$=x\cup y\cup z^c$	$=Ter(xy)$,
$y\to x\cup z$	$=x^c\cup z^c\to y^c$	$=x\cup y^c\cup z$	$=Ter(xz)$,
$x\to y\cup z$	$=y^c\cup z^c\to x^c$	$=x^c\cup y\cup z$	$=Ter(yz)$,
$\Omega\to x\cup y\cup z$	$=x^c\cup y^c\cup z^c\to\emptyset$	$=x\cup y\cup z$	$=Ter(xyz)$,

	$\emptyset$	$\emptyset$ Невозможноесо-бытие	Ложь, Противоречие
	$\{\{x_1\}\}$	$x_1 \cap x_2^c, x_1 \backslash x_2$ Прямая разность со-бытий	$x_1 \nrightarrow x_2$ Прямая антиимпликация
	$\{\{x_2\}\}$	$x_1^c \cap x_2, x_1 / x_2$ Обратная разность со-бытий	$x_1 \nleftarrow x_2$ Обратная антиимпликация
	$\{\{x_1, x_2\}\}$	$x_1 \cap x_2$ Пересечение	$x_1 \& x_2, x_1 \wedge x_2,$ x_1 AND x_2 Конъюнкция
	$\{\{x_1\},\{x_2\}\}$	$x_1 \Delta x_2$ Симметрическая разность со-бытий	$x_1 \nleftrightarrow x_2, x_1 \not\equiv x_2,$ x_1 XORx_2 Исключительная дизъюнкция
	$\{\{x_1\},\{x_1, x_2\}\}$	x_1 Со-бытие x_1	x_1 Утверждение x_1
	$\{\{x_2\},\{x_1, x_2\}\}$	x_2 Со-бытие x_2	x_2 Утверждение x_2
	$\{\{x_1\},\{x_2\},\{x_1, x_2\}\}$	$x_1 \cup x_2$ Объединение со-бытий	$x_1 \vee x_2, x_1 \cup x_2,$ x_1 OR x_2 Дизъюнкция
	$\{\emptyset\}$	$x_1^c \cap x_2^c$ Антиобъединение со-бытий	$x_1 \downarrow x_2, x_1$ NOR x_2 Стрелка Пирса Антидизъюнкция
	$\{\emptyset,\{x_1\}\}$	x_2^c Дополнение со-бытия x_2	$\neg x_2$ Отрицание x_2
	$\{\emptyset,\{x_2\}\}$	x_1^c Дополнение со-бытия x_1	$\neg x_1$ Отрицание x_1
	$\{\emptyset,\{x_1, x_2\}\}$	$x_1 \Delta x_2^c$ Эквивалентность со-бытий	$x_1 \leftrightarrow x_2, x_1 \equiv x_2,$ x_1 IFF x_2 Если и только если Эквивалентность
	$\{\emptyset,\{x_1\},\{x_2\}\}$	$x_1^c \cup x_2^c$ Антипересечение со-бытий	$x_1 \uparrow x_2,$ x_1 NAND x_2 Штрих Шеффера Антиконъюнкция
	$\{\emptyset,\{x_1\},\{x_1, x_2\}\}$	$x_1 \cup x_2^c, x_1 \not\leftarrow x_2$ Прямая антиразность со-бытий	$x_1 \longleftarrow x_2$ Обратная импликация
	$\{\emptyset,\{x_2\},\{x_1, x_2\}\}$	$x_1^c \cup x_2, x_1 \not\rightarrow x_2$ Обратная антиразностьсо-бытий	$x_1 \rightarrow x_2$ Прямая импликация
	$\{\emptyset,\{x_1\},\{x_2\},\{x_1, x_2\}\}$	Ω Достоверное со-бытие	Истина, Тавтология

	$\emptyset$ Невозможное со-бытие	F Ложь, противоречие

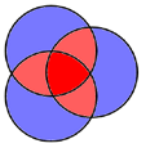
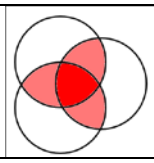
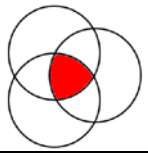
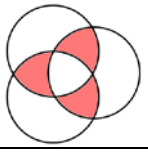
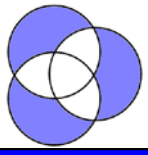
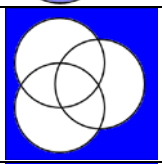
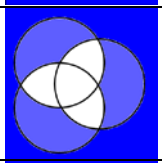
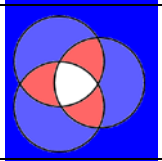
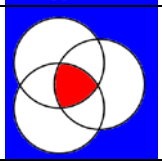
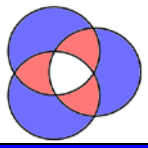
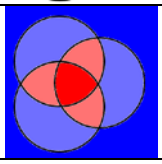
	$x_1 \cup x_2 \cup x_3$ 1-чердак, 3-подвал, объединение	$x_1 \vee x_2 \vee x_3$ OR
	2-чердак, 2-подвал	
	$x_1 \cap x_2 \cap x_3$ 3-слой, 3-чердак, 1-подвал, пересечение	$x_1 \wedge x_2 \wedge x_3$ AND
	2-слой	
	1-слой	
	0-слой, 1-античердак, 3-антиподвал	$x_1 \downarrow x_2 \downarrow x_3$ NOR Стрелка Пирса
	2-античердак, 2-антиподвал	
	3-античердак, 1-антиподвал	$x_1 \uparrow x_2 \uparrow x_3$ NAND Штрих Шефера
	Совпадение	$\equiv (x_1, x_2, x_3)$
	Отличие	$\not\equiv (x_1, x_2, x_3)$
	Ω Достоверное со-бытие	T Истина, тавтология

Рисунок 1. и Рисунок 2.

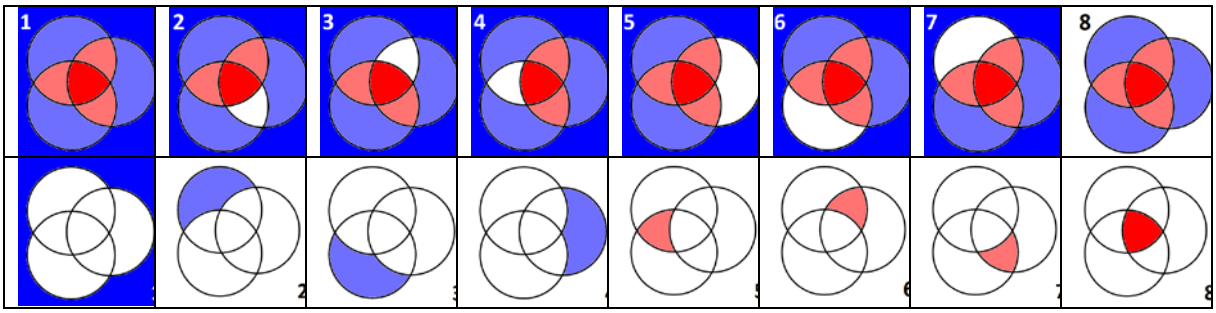


Рисунок 3. Диаграмма Венна 16 «общих импликаций» и их дополнений (отрицаний) – тернарных теоретико-множественных операций над триплетом со́-бытий $\mathfrak{X} = \{x, y, z\}$*, включая вверху: штрих Шеффера (1) и дизъюнкцию (8); - внизу: стрелку Пирса (1) и конъюнкцию (8); обще-импликационные террасные представления которых приведены ниже.*

где операция общей импликации «→» выполняется последней и использована аббревиатура $Ter(X) = ter(X//\{x, y, z\})$,когда $X \subseteq \{x, y, z\}$.

Дополнение (отрицание) общей импликации может быть представлено террасными событиями 1-го рода:

$$\Omega \nrightarrow x \cup y \cup z \quad = x^c \cap y^c \cap z^c \nrightarrow \emptyset \quad = x^c \cap y^c \cap z^c \quad = ter(\emptyset),$$
$$x \nrightarrow y \cup z \quad = y^c \cap z^c \nrightarrow x^c \quad = x \cap y^c \cap z^c \quad = ter(x),$$
$$y \nrightarrow x \cup z \quad = x^c \cap z^c \nrightarrow y^c \quad = x^c \cap y \cap z^c \quad = ter(y),$$
$$z \nrightarrow x \cup y \quad = x^c \cap y^c \nrightarrow z^c \quad = x^c \cap y^c \cap z \quad = ter(z),$$
$$x \cap y \nrightarrow z \quad = z^c \nrightarrow x^c \cup y^c \quad = x \cap y \cap z^c \quad = ter(xy),$$
$$x \cap z \nrightarrow y \quad = y^c \nrightarrow x^c \cup z^c \quad = x \cap y^c \cap z \quad = ter(xz),$$
$$y \cap z \nrightarrow x \quad = x^c \nrightarrow y^c \cup z^c \quad = x^c \cap y \cap z \quad = ter(yz),$$
$$x \cap y \cap z \nrightarrow \emptyset \quad = \Omega \nrightarrow x^c \cup y^c \cup z^c \quad = x \cap y \cap z \quad = ter(xyz),$$

где операция отрицания общей импликации «↛» выполняется последней и использована аббревиатура $ter(X) = ter(X//\{x, y, z\})$, когда $X \subseteq \{x, y, z\}$.

Заключение.Аналогии между булевыми алгебрами множеств, событий и высказываний добавили в логику высказываний операции сет-чердак, сет-подвал и сет-слой множества логических высказываний. А предложенное на основе этих же аналогий семейство необычных для эвентологической теории [1] операций над множеством событий, названных «общая импликация событий», позволило построить «обще-импликационное» представления террасных распределений 1-го и 4-го рода дуплета и триплета событий. Эти представления, по-видимому, являются простейшими иллюстрациями нового понятия *со́-бытия*, как имени колмогоровского события, введенного в.

Список литературы

[1] О. Ю. Воробьев. *Эвентология.* Сибирский федеральный университет, Красноярск, 2007, 435с., https://www.academia.edu/179393/Vorobyev_0.Yu._Eventology._-_Krasnoyarsk_Siberian_Federal_University._-_2007._-_435p.

[2] M.H. Stone. The theory of representations of Boolean algebras. *Transactions of the American Mathematical Society*, 40:37-111, 1936.

[3]A. Tarski. Der Wahrheitsbegriff in den formalisiertenSprachen. *StudiaPhilosophica*, 1:261-405, 1935, Translation of Tarski 1933 by L. Blaustein, with a postscript added.

[4] О. Ю. Воробьев. *Сет-суммирование.* Наука, Новосибирск, 1993, 137с.