

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА МАТЕМАТИЧЕСКИХ МЕТОДОВ ПРОГНОЗИРОВАНИЯ

Курсовая Работа

Поиск скрытых поведенческих паттернов

Вишневский Валерий
группа 317
научный руководитель: Ветров Дмитрий Петрович

Москва, 2009

Аннотация

В работе рассматривается алгоритм поиска скрытых поведенческих закономерностей (hidden T-Patterns), которые сложно обнаружить визуально, или с помощью стандартных статистических методов. Проводится анализ параметров алгоритма. Предложенный метод поиска паттернов основывается на определении взаимосвязи между парами событий, названной критическим отношением (critical interval relation). Поиск производится снизу вверх: алгоритм сначала находит простые закономерности, потом, путем соединения простых, образуются более сложные паттерны. На каждом шаге проводится отбор самых существенных и полных паттернов. Данный алгоритм был реализован на языке Си в виде консольного приложения и модуля Matlab.

Содержание

1	Введение	3
2	Теоретическое обоснование. Определения	3
2.1	Наблюдаемый временной ряд	3
2.2	Понятие критического интервала	3
2.3	Определение Т-Паттерна	4
3	Алгоритм поиска Т-Паттернов	5
3.1	Формальное описание	6
3.2	О параметрах α и N_{min}	7
3.3	Случайные паттерны	7
4	Анализ результатов работы алгоритма	8
4.1	Стратегия выбора критического интервала	8
4.2	Запрет паттернов с одинаковыми событиями	9
5	Документация модуля к Matlab	9
5.1	Описание функций	10
5.2	Пример использования	12
6	Заключение	13

1 Введение

Скрытые, или неочевидные закономерности в поведении давно представляют интерес и исследуются учеными. Такие закономерности присутствуют и в нашей повседневной жизни, например, человек обычно перед тем как приступить к обеду, сначала подходит к столу, отодвигает стул, садится, берет столовые приборы. Поведенческие паттерны присутствуют и в командных играх таких, как футбол [2]. Проводятся исследования и на животных: некоторое время наблюдается поведение отдельного зверька, или группы животных. Определяются некоторые интересные исследователей действия, которые подопытные могут совершить (например, зверек облизал мордочку, или куда-то побежал). Моменты времени, когда эти действия были начаты, фиксируются и записываются. В результате наблюдений получается множество пар: момент времени, совершенное действие. Далее исследователи могут, например, изменить окружающие воздействия на зверьков, или скормить им какое-то вещество, после чего снова провести наблюдения, и затем проанализировать изменение поведения животных.

2 Теоретическое обоснование. Определения

2.1 Наблюдаемый временной ряд

Пусть время наблюдения разбито на N_t интервалов. В каждый момент *периода наблюдения* (*observation period*) $[1, N_t]$ может произойти некоторое событие (*действие, event*)¹ из множества допустимых событий \mathcal{E} (*event types*). Соответственно, каждому типу событий сопоставляется множество моментов времени $TS(\mathbf{A})$

$$TS(\mathbf{A}) = \{T_{\mathbf{A},1}, \dots, T_{\mathbf{A},N_{\mathbf{A}}}\}, \quad \mathbf{A} \in \mathcal{E}, \quad 0 \leq T_{\mathbf{A},i} \leq N_t \quad (i = 1, \dots, N_{\mathbf{A}})$$

2.2 Понятие критического интервала

Во время поиска закономерностей в данных, нас интересуют отношения между распределениями отдельных событий. Если предположить, что в исходных не существует никаких закономерностей, то каждое событие должно появляться независимо, от других; то есть распределение компонентов должно быть независимым. Но паттерн характеризуется появлением своих компонентов в одинаковом порядке, более того, временные интервалы, разделяющие компоненты, должны быть примерно одинаковыми.

¹ Чаще всего понимается, что в этот момент времени имеет место *начало* действия

Будем говорить, что события \mathbf{A} и \mathbf{B} связаны отношением *критического интервала* (*Critical Interval, CI*), если, после появления события \mathbf{A} в момент времени t , существует интервал $[t + d_1, t + d_2]$, ($d_2 \geq d_1 \geq 0$), содержащий \mathbf{B} , чаще, чем это ожидается из предположения о независимости событий. Данную взаимосвязь будем обозначать, как $\mathbf{A}[d_1, d_2]\mathbf{B}$, или, короче, (\mathbf{AB}) .

Далее раскроем понятие «чаще чем это ожидается». Пусть $N_{\mathbf{A}}$ и $N_{\mathbf{B}}$ — количество возникновений \mathbf{A} и \mathbf{B} , соответственно, в течение $[1, N_t]$. $P(\mathbf{A}) = N_{\mathbf{A}}/N_t$ — вероятность появления события \mathbf{A} в некоторый момент времени; $P(\neg\mathbf{A}) = 1 - P(\mathbf{A})$. $P(\neg\mathbf{A})^d$ — вероятность, что \mathbf{A} не появится в течение какого-либо интервала $[d_1, d_2]$, ($d = d_2 - d_1 + 1$), длины d . Вероятность наблюдать \mathbf{A} на интервале длины d один или более раз, равна $1 - P(\neg\mathbf{A})^d$.

Зафиксируем события \mathbf{A} , \mathbf{B} , и длину интервала d . Приняв гипотезу о независимом распределении событий, событие \mathbf{B} содержится² в интервале длины d , после события \mathbf{A} , $N_{\mathbf{A}} * (1 - P(\neg\mathbf{B})^d)$ раз.

$$\rho = P(\geq N_{\mathbf{AB}}) = 1 - P(< N_{\mathbf{AB}})$$

— априорная вероятность того, что $N_{\mathbf{AB}}$ из $N_{\mathbf{A}}$ интервалов содержат вхождения \mathbf{B} . Очевидно, что $P(< N_{\mathbf{AB}})$ распределено по биномиальному закону, где $N_{\mathbf{A}}$ — количество «испытаний», $1 - P(\neg\mathbf{B})^d$ — вероятность «успеха». Следовательно,

$$\rho = P(\geq N_{\mathbf{AB}}) = 1 - \sum_{i=0}^{N_{\mathbf{AB}}-1} C_{N_{\mathbf{A}}}^i (1 - P(\neg\mathbf{B})^d)^i P(\neg\mathbf{B})^{N_{\mathbf{A}}-i}$$

Полученная вероятность ρ сравнивается с пороговым значением α , являющимся структурным параметром поиска: если $\rho \leq \alpha$, то заданный интервал признается критическим. Заметим, что ρ зависит от $N_{\mathbf{A}}, N_{\mathbf{B}}, N_t, N_{\mathbf{AB}}, d$:

$$\rho = \rho(N_{\mathbf{A}}, N_{\mathbf{B}}, N_t, N_{\mathbf{AB}}, d)$$

2.3 Определение Т-Паттерна

Дадим рекурсивное определение Т-Паттерна. Договоримся называть каждое допустимое событие *псевдопаттерном*. Тогда Т-Паттерн \mathbf{Q} можно определить как:

$$\mathbf{Q} = \mathbf{X}_1[dL_1, dR_1]\mathbf{X}_2[dL_2, dR_2] \dots \mathbf{X}_i[dL_i, dR_i]\mathbf{X}_{i+1} \dots \mathbf{X}_m,$$

$$Events(\mathbf{Q}) = \begin{cases} \{Events(\mathbf{X}_1), \dots, Events(\mathbf{X}_m)\}, & \text{если } \mathbf{Q} \text{ — Т-Паттерн} \\ \mathbf{Q}, & \text{если } \mathbf{Q} \text{ — событие (псевдопаттерн)} \end{cases}$$

²Один или более раз.

где \mathbf{X}_i , ($i = 1 \dots m$) — Т-Паттерн, или псевдопаттерн. Для паттернов отношение критического интервала $\mathbf{Q}_L[dL, dR]\mathbf{Q}_R$ вводится, как и для событий, с учетом того, что интервал $[dL, dR]$ отсчитывается от последнего элемента \mathbf{Q}_L , и вхождение \mathbf{Q}_R определяется его первым элементом. Будем называть *двойными сериями* (*double series, DS*) паттерна \mathbf{Q} , множество $\{\{Left_i, Right_i\}_{i=1 \dots N_Q}\}$, где N_Q — количество появлений паттерна \mathbf{Q} , $Left_i, Right_i$ — индексы начального и конечного событий i -го появления паттерна \mathbf{Q} .

3 Алгоритм поиска Т-Паттернов

Алгоритм поиска Т-Паттернов, описанный ниже, заключается в итеративном повторении двух стадий: конструирование новых паттернов и удаление неполных паттернов. На выходе алгоритм выдает множество Т-Паттернов с их сопутствующими характеристиками: критическим интервалом, уровнем значимости, частотой встречаемости.

Пусть \mathcal{D}_i — множество паттернов, обнаруженных к i -ой итерации. Фактически, множество

$$\mathcal{D}_m \setminus \mathcal{E},$$

где m — номер последней итерации, и будет результатом работы алгоритма.

Стадия 1: Поиск и конструирование:

На данном шаге, для любой упорядоченной пары

$$(\mathbf{Q}', \mathbf{Q}'') : \mathbf{Q}', \mathbf{Q}'' \in \mathcal{D}_i,$$

проверяется существование критической связи. Если критическая связь $[dL, dR]$ была найдена, и паттерн $(\mathbf{Q}'\mathbf{Q}'')$ встречается чаще, чем N_{min} раз, то в множество \mathcal{D}_{i+1} добавляется новый паттерн $\mathbf{Q}'[dL, dR]\mathbf{Q}''$.

Однако такие действия приводят к тому, что один и тот же паттерн **ABCD** может быть сконструирован разными способами (например как **(A(BCD))** и **((AB)(CD))**), что ведет к заполнению множества \mathcal{D} лишними паттернами, и как следствие, к замедлению работы программы. Для избежания данной проблемы предлагается следующее решение: паттерн $(\mathbf{Q}'\mathbf{Q}'')$ добавляется в множество \mathcal{D} тогда и только тогда, когда не существует паттерна \mathbf{P} из \mathcal{D} такого, что $Events(\mathbf{P}) = Events((\mathbf{Q}'\mathbf{Q}''))$ и $DS(\mathbf{P}) = DS((\mathbf{Q}'\mathbf{Q}''))$. Другими словами, найденный паттерн, перед добавлением, сравнивается с уже существующими паттернами и проверяется, не является ли он дубликатом.

Стадия 2: Удаление неполных паттернов:

На данной стадии алгоритм стремится удалить найденные паттерны, являющиеся меньшими частями, или неполными версиями других обнаруженных паттернов. Для

индикации таких паттернов можно применять разные эвристики. Ниже опишем условие, взятое из статьи [1].

Итак, паттерн Q_x считается менее полным, чем Q_y , если Q_x и Q_y появляются одинаково часто, и все события возникающие в Q_x , также возникают в Q_y .

3.1 Формальное описание

Require: \mathcal{E} — допустимые события,

N_t — продолжительность наблюдения,

α — минимальный уровень значимости,

N_{min} — минимальное количество вхождений паттерна.

```

1:  $\mathcal{D}_{-1} = \emptyset$ 
2:  $\mathcal{D}_0 = \mathcal{E}$  ▷ Инициализируем множество паттернов
3:  $t = 0$ 
4: while  $\mathcal{D}_t \neq \mathcal{D}_{t-1}$  do
5:    $t = t + 1$ 
6:    $\mathcal{D}_t = \mathcal{D}_{t-1}$ 
7:   for  $P_L \in \mathcal{D}_{t-1}$  do ▷ Стадия 1
8:     for  $P_R \in \mathcal{D}_{t-1}$  do
9:       for all  $d_L, d_R \in [1, N_t]$  таких, что  $d_L \leq d_R$  do
10:        if  $\rho(N_{P_L}, N_{P_R}, N_t, N_{P_L P_R}, d_R - d_L + 1) < \alpha$  then
11:          if  $\text{isUnique}(P_L \cup P_R, \mathcal{D}_t)$  then
12:             $\mathcal{D}_t = \mathcal{D}_t \cup \{P_L \cup P_R\}$ 
13:   for  $P_L \in \mathcal{D}_t \setminus \mathcal{E}$  do ▷ Стадия 2
14:     for  $P_R \in \mathcal{D}_t$  do
15:       if  $P_L \subset P_R^3$  and  $|\text{DS}(P_L)| = |\text{DS}(P_R)|$  and  $\text{isIntersect}(P_L, P_R)$  then
16:         удалить  $P_L$  из  $\mathcal{D}_t$ 
17: function  $\text{ISUNIQUE}(Q, \mathcal{D})$ 
18:   for  $P \in \mathcal{D}$  do
19:     if  $\text{DS}(Q) = \text{DS}(P)$  and  $\text{Events}(Q) = \text{Events}(P)$  then
20:       return false
21:   return true
22: function  $\text{ISINTERSECT}(P_L, P_R)$ 
23:   for  $i = 1 \dots |\text{DS}(P_L)|$  do
24:     if  $\text{DS}_{i, \text{Left}}(P_L) > \text{DS}_{i, \text{Right}}(P_R)$  or  $\text{DS}_{i, \text{Right}}(P_L) < \text{DS}_{i, \text{Left}}(P_R)$  then
25:       return false
26:   return true

```

³Имеется в виду упорядоченная вложенность множеств. Т.е. $abd \subset abcd$, но $bdc \not\subset abcd$

27: `return true`

3.2 О параметрах α и N_{min}

Так как для задачи поиска Т-Паттернов не вводится определение функционала качества найденных паттернов, то алгоритм требует ручной настройки параметров α и N_{min} . Выбор значений параметров должен основываться на специфике наблюдаемых процессов и ожидаемых результатов. Однако выбор значений $\alpha = 0.005$ и $N_{min} = 3$ обычно удовлетворителен [1, с. 99]. Для более тонкой настройки поиска, можно использовать разные α и N_{min} для паттернов разной длины. Например, при значениях $\alpha = 0.00001$, $N_{min} = 7$ для паттернов длины 2, и $\alpha = 0.005$, $N_{min} = 3$ для паттернов длины 3 и более, алгоритм найдет только несколько самых ярко выраженных паттернов длины 2, и уже потом будет конструировать множество более длинных паттернов.

3.3 Случайные паттерны

Когда исследования проводятся на больших наборах данных, то Т-Паттерны могут возникать даже, если выборка была сгенерирована случайно. Поэтому, для найденного множества паттернов было бы полезно оценить, являются ли они случайными, или «структурными». Одним из подходов к решению данной задачи, является анализ рандомизированных данных:

По множеству \mathcal{E} строится множество \mathcal{E}' :

$$\mathcal{E}' = \{rand(\mathbf{A}) | \mathbf{A} \in \mathcal{E}\}, \text{ где}$$

$$rand(\mathbf{A}) = \mathbf{A}' : TS(\mathbf{A}') = \{\xi_1, \dots, \xi_{N_A}\}$$

$$\xi_1, \dots, \xi_{N_A} \sim \mathbf{U}[1, N_t].$$

Проще говоря, в исходных данных времена появления всех событий заменяются на случайные. Причем, число появлений любого события в исходных и рандомизированных данных одинаковое.

Для полученного множества \mathcal{E}' применяется процесс поиска паттернов с теми же параметрами, которые применялись на исходных данных. Описанные действия исполняются несколько раз, после чего, результаты поиска на рандомизированных данных сравниваются с исходными.

Считается, что поиск Т-Паттернов прошел успешно, если в исходных данных было выявлено значительно больше Т-Паттернов, чем в рандомизированных, или они оказались длиннее.

4 Анализ результатов работы алгоритма

4.1 Стратегия выбора критического интервала

Во время поиска связи критического интервала $Q_L[dL, dR]Q_R$ между двумя паттернами, вообще говоря, проверяются все возможные интервалы $[dL, dR]$. На практике, не является редкостью случай, когда для двух паттернов существует несколько пар dL и dR , удовлетворяющих отношению критического интервала. Для выбора конкретных значений dL и dR , предлагается использовать одну из нижеописанных стратегий. Для каждой стратегии представлен результат работы алгоритма на тестовых данных. Данные содержат паттерн длины 7, встречающийся 8 раз.

Выбор кратчайшего критического интервала: При использовании данной стратегии, на каждом шаге выбирается критический интервал, имеющий наименьшую длину d ($d = dR - dL + 1$). Такой подход позволяет уменьшить длину критических связей, тем самым выявляя более выраженные и «стройные» паттерны. Одним из недостатков данного метода является эффект «расщепления». Поясним данный эффект на примере: пусть в исходных данных существует критическая связь $A[4, 20]B$, наблюдаемая 15 раз. Данная связь настолько ярко выражена, что алгоритму не требуется подбирать границы критического интервала, чтобы B появлялось после A все 15 раз. Алгоритм, скорее, выделит два(или даже больше) критических интервала: $A[4, 12]B$, наблюдаемый 6 раз, и $A[13, 18]B$, наблюдаемый 7 раз. Таким образом, один ярко выраженный паттерн будет распадаться на несколько более редких, что в свою очередь, может помешать дальнейшему выявлению закономерностей. Ниже описанный метод позволяет избавиться от такой проблемы.

Выбор самого длинного критического интервала: Как следует из названия, при использовании этой стратегии, среди всех значимых $[dL, dR]$, выбирается интервал, соответствующий наибольшему значению d . В результате некоторые паттерны «загрубляются», и алгоритм стремится найти максимальное количество вхождений каждого критического интервала.

Выбор самого значимого критического интервала: При этом подходе выбирается критический интервал, имеющий максимальный уровень значимости ρ .

В общем случае, нету каких-либо рекомендаций по выбору стратегии поиска критического интервала. Выбор должен основываться на априорных сведениях о наблюдаемом процессе и ожидаемых результатов эксперимента.

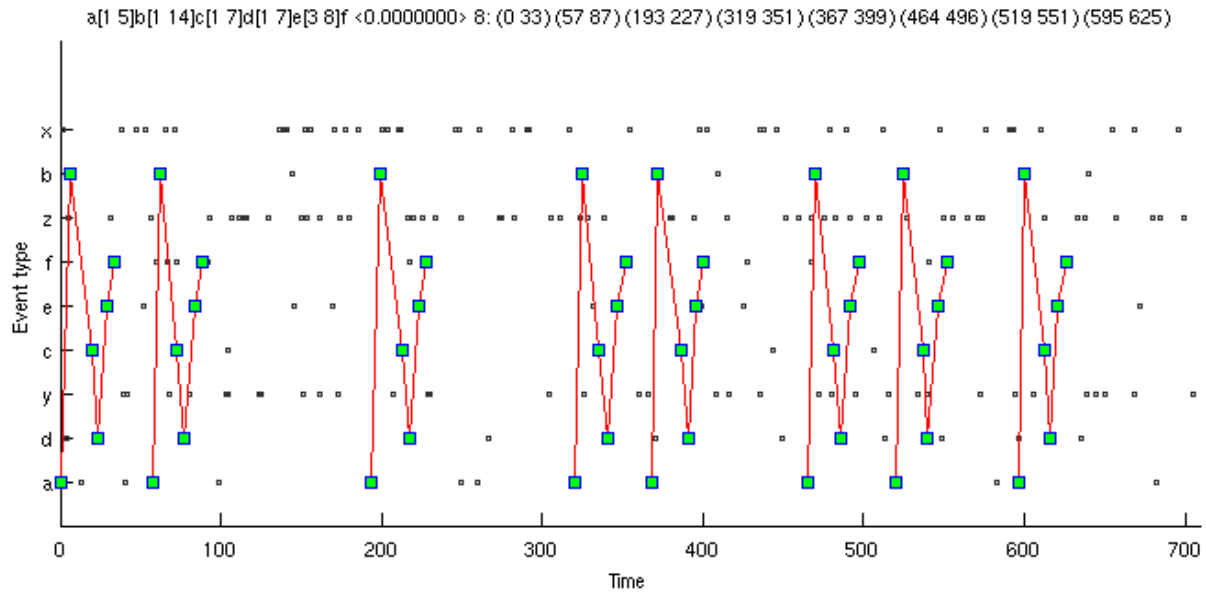


Рис. 1: Визуализация исходного паттерна abcdef.

Стратегия	Найдено паттернов	Найдено вхождений целевого паттерна
Короткий	40	Найдено 3 представления целевого паттерна. Для каждого представления 4 вхождения.
Длинный	5	8
Значимый	7	7

4.2 Запрет паттернов с одинаковыми событиями

Если при поиске поведенческих закономерностей известно, что в один и тот же паттерн не могут входить два одинаковых события, то существует возможность сообщить это алгоритму. Таким образом, если, $Events(Q_L) \cap Events(Q_R) \neq \emptyset$, то паттерн $(Q_L Q_R)$, просто не будет создан. Это условие позволяет отбросить множество лишних паттернов, и найти более длинные и ценные закономерности.

5 Документация модуля к Matlab

Matlab-реализация алгоритма состоит из следующих файлов:

mexPattern.mex*: откомпилированный mex-файл, реализующий алгоритм поиска паттернов.

mexPattern.m: объявление mex-функции.

T_DRAW_PATTERNS.m: графический вывод найденных паттернов.

T_GENERATE_PATTERNS.m: создание искусственных паттернов во временных рядах.

T_LOAD_FILE.m: загрузка временного ряда из файла для дальнейшей работы с ним.

T_STAT_VALIDATE.m: процедура статистической валидации.

test.m: пример использования модуля.

5.1 Описание функций

```
function [events, Nt, ts] = T_GENERATE_PATTERN( pat_sym, noise_sym, CIs, Npat,  
        dist_b_patterns, Pnoise1, Pnoise2 )
```

Создает временной ряд, содержащий один искусственный паттерн.

Параметр	Описание
<i>Вход:</i>	
pat_sym	Матрица $1 \times N_p$ типа char, определяющая паттерн. Каждый символ — событие.
noise_sym	Матрица $1 \times N$ типа char. Определяет события которые будут генерироваться случайно.
CIs	Матрица $(N_p - 1) \times 2$ типа int. В i -й строке которой, записан соответствующий критический интервал.
Npat	Количество паттернов, которые требуется сгенерировать.
dist_b_patterns	Максимальное расстояние между двумя появлениями паттерна.
P_noise1	Частота встречаемости шумовых символов.
P_noise2	Вероятность того, что символ из паттерна будет зашумлен.
<i>Выход:</i>	
events	Массив структур $1 \times N$, где N — количество событий. Каждая структура состоит из двух полей: event_name — строка названия события, и indexes — матрица $1 \times N$ типа int. Определяет время появления событий.
Nt	Продолжительность получившегося периода наблюдений.
ts	Символьная матрица $1 \times Nt$.

```
function patterns = mexPattern(events, Nt, levels, allow_same_events, ci_strategy);
```

Реализует поиск паттернов во временных рядах.

Параметр	Описание
<i>Вход:</i>	
texttvents, Nt	См. определение T_GENERATE_PATTERN.
levels	Матрица Nx3 уровней значимости. Каждая строка содержит длину паттернов, к которым должны применяться следующие параметры; минимальный уровень значимости α ; минимальное количество вхождений паттерна N_{min} .
allow_same_events	1 если, разрешается появление одинаковых событий в паттерне, 0 иначе.
ci_strategy	Стратегия выбора критического интервала. 1 — стратегия выбора длиннейшего интервала, 2 — кратчайшего, 3 — самого значимого.
<i>Выход:</i>	
patterns	Массив структур. Каждая структура описывает найденный паттерн. Поля структуры: Events — индексы событий, которые составляют паттерн. CIs — интервалы, соответствующие критическим связям между событиями в паттерне. Sign — уровень значимости найденного паттерна. Nab — количество появлений паттерна. DS — двойные серии (<i>double series</i>) паттерна. String — строка, описывающая паттерн в следующем формате: $Event_1[dL_1, dR_1]Event_2 \dots Event_m < \text{уровень значимости} > \{N_{ab}\} : (DS_1)(DS_2) \dots (DS_{N_{ab}})$

```
function T_DRAW_PATTERNS(patterns, events, Nt, np)
```

Строит диаграмму найденных паттернов.

Параметр	Описание
<i>Вход:</i>	
patterns, events, Nt	См. предыдущие определения.
np	Номер паттерна, который нужно представить. Или -1 для последовательного вывода всех паттернов.

```
function [events, Nt] = T_LOAD_FILE(fname)
```

Загружает временной ряд из файла для последующей работы с ним. Формат входного файла:

```

Time    Event
0      :
time    event
...
time    event
time    &

```

```
function [p] = T_STAT_VALIDATE( Nt, events, levels, nvalidations )
```

Процедура статистической валидации.

Параметр	Описание
<i>Вход:</i>	
Nt, events, levels	См. предыдущие определения.
nvalidations	Количество повторений процедуры рандомизации.
<i>Выход:</i>	
p	Целочисленная матрица 1 x nvalidations. В каждой ячейке — количество паттернов, найденных в рандомизированных данных.

5.2 Пример использования

Продемонстрируем работу с данным модулем на двух примерах. Следующий код сгенерирует временной ряд, содержащий паттерн $a[4, 5]b[10, 14]c[3, 6]d[5, 7]e[3, 6]f$, и потом запустит процедуру поиска паттернов.

```

1  CIs = [ 4  5 ;
2         10 14;
3         3  6 ;
4         5  7 ;
5         3  6 ];
6  levels = [ 2 0.00005 4 ;
7            4  0.0005 3 ];
8  [events, Nt, ts] = T_GENERATE_PATTERN('abcdef', 'xyz', CIs, 8, 120, 0.2, 0.05);
9  patterns = mexPattern(events, Nt, levels', 0, 1) %Поиск паттернов без повторяющихся
10                                           %событий. Стратегия поиска
11                                           %длиннейшего интервала.

```

Далее показана возможность загрузки данных из файла и применение процедуры статистической валидации. Каждый паттерн будет представлен в виде диаграммы.

```
1 levels = [ 1 0.005 3 ];
2 [events2, Nt] = T\_LOAD\_FILE('../real_data/grooming_hamsters/auratus1.txt');
3 patterns = mexPattern(events2, Nt, levels', 0, 2);
4 T_STAT_VALIDATE(Nt, events2, levels, 7) %Те же параметры, что и в поиске!
5 T_DRAW_PATTERNS(patterns, events2, Nt, -1 );
```

6 Заключение

Задача поиска поведенческих закономерностей не может быть решена автоматически. Этот процесс подразумевает тесную связь с экспертом. Именно человек должен анализировать результаты работы алгоритма, проводить эксперименты с разными параметрами и анализировать результаты. На основании результатов работы алгоритма некоторые гипотезы могут быть подтверждены, или опровергнуты. Разработанный алгоритм является лишь инструментом для исследований.

На данный момент описанный метод поиска Т-Паттернов полностью реализован. В будущем планируется создать полноценное приложение с богатыми возможностями для анализа результатов и удобным интерфейсом. Рассматривается возможность распараллеливания алгоритма, с целью переноса вычислительно-сложных частей программы на видеокарты Nvidia Cuda. Поддержка многопроцессорных систем позволит применять более сложные эвристики и проводить более глубокий перебор, что в свою очередь положительно скажется на качестве алгоритма.

Список литературы

- [1] Magnus Magnusson (2000) «Discovering hidden time patterns in behavior: T-Patterns and their detection» University of Iceland/
- [2] Gudberg K. Jonsson, Sigridur H. Bjarkadottir, «Detection of real-time patterns in sports interaction in football»