Наука и Образование
МГТУ им. Н.Э. Баумана
Сетевое научное издание
ISSN 1994-0408

Наука и Образование. МГТУ им. Н.Э. Баумана.
Электрон. журн. 2015. № 11. С. 382–405.

DOI: **10.7463/1115.0827396**

Представлена в редакцию: 03.10.2015
Исправлена: 20.10.2015

УДК 519.6

Эффективность классификации многомерных временных рядов с помощью шейплетов

Карпенко А. П.[1,*], Кострубин М. С.[2], Чернышев А. С.[1]

[*]apkarpenko@mail.ru

[1]МГТУ им. Н.Э. Баумана, Москва, Россия
[2] ООО "ЭКСПЕРТЕК ИБС", Москва. Россия

Рассматриваем один из методов классификации временных рядов, не требующий выделения характерных признаков сигналов. Метод основан на использовании шейплетов временного ряда (time series shapelets) – небольших фрагментов этого ряда, максимально отражающих свойства одного из его классов. Публикация имеет целью исследовать эффективность ряда модификаций исходного метода шейплетов применительно к классификации многомерных рядов, что представляет собой малоисследованную задачу. Результаты исследования показывают, что разработанный алгоритм и программная система обеспечивают точность классификации, составляющую, в лучших случаях, около 85%. Время, затрачиваемое на поиск шейплета, возрастает пропорционально размерности входных данных.

Ключевые слова: классификация многомерных временных рядов; шейплет временного ряда

Введение

Задачи классификации многомерных временных рядов возникают во многих областях науки и техники – в системах ориентации и стабилизации, в нейрокомпьютерных интерфейсах, системах распознавании речи, при идентификации динамических систем и прочее. Обычно классификаторы временных рядов предполагают выполнение следующих этапов [1].

- Предобработка сигналов (фильтрация сигналов от шумов, устранение артефактов и т.д.).
- Выделение характерных признаков сигнала (например, амплитуда, частота, спектр, мощности спектральных диапазонов, спектральная плотность энергии, корреляция и автокорреляция) с целью перехода от исходного пространства динамических данных к пространству многомерных статических данных.
- Классификация в пространстве характерных признаков сигналов с использованием классических методов и алгоритмов классификации многомерных данных (алгоритмы линейной и нелинейной классификации, классификаторы, основанные

на методе ближайших соседей и на основе искусственных нейронных сетей, комбинированные классификаторы).

Один из методов классификации временных рядов, не требующий выделения характерных признаков сигналов, предложен в 2009 году. Метод основан на использовании шейплетов временного ряда (*time series shapelets*) – небольших фрагментов этого ряда, максимально отражающих свойства одного из его классов [2].

Исходный вариант метода шейплетов позволяет разделить набор одномерных временных рядов на два класса по критерию наличия или отсутствия в них данного шейплета. В настоящее время предложено большое число модификаций метода, которые позволяют увеличить скорость классификации, осуществлять классификацию многомерных временных рядов, многоклассовую классификацию, использовать в качестве шейплетов агрегацию фрагментов временного ряда (так называемые логические шейплеты) и т.д. [3 - 10]. Известен ряд примеров эффективного использования шейплетов для решения практических задач классификации временных радов (см. библиографию на сайте [11]).

Можно выделить следующие преимущества метода классификации временных рядов на основе шейплетов.

- В отличие от большинства классификаторов временных рядов, данные классификаторы позволяют получить интерпретируемые результаты, что обеспечивает лучшее понимание и визуализацию данных.
- Для некоторых временных рядов шейплеты могут быть существенно точнее и эффективнее, поскольку способны «подмечать» их локальные характерные признаки.
- По сравнению со многими другими методами классификации шейплеты могут обеспечить значительно более высокую скорость классификации, поскольку обладают временной сложностью O(ml), где m – число рядов в наборе данных, l – длина шейплета.

Заметим, что близкий подход использован в отечественной работе [12] с целью экстраполяции временных рядов. Аналогичный шейплету фрагмент временного ряда называется в этой работе *выборкой максимального подобия*.

Не смотря на значительное число публикаций по теории и приложениям шейплетов для классификации временных рядов, актуальной остается задача оценки эффективности этой техники. Данная публикация имеет целью исследовать эффективность ряда модификаций исходного метода шейплетов применительно к классификации многомерных рядов, что представляет собой малоисследованную задачу.

Публикация организована следующим образом. В первом разделе даны основные используемые определения и обозначения, представлена постановка задачи классификации многомерного временного ряда с помощью шейплетов. Второй раздел содержит изложение базового метода бинарной классификации на основе шейплетов, третий раздел - различные обобщения и модификации метода. В четвертом разделе представлено разработанное программное обеспечение, реализующее модификации, рассмотренные в предыдущем разделе. В пятом разделе приведены результаты вычислительных экспериментов, подтверждающие эффективность принятых алгоритмических и программных решений.

Полагаем далее, что $|A|$ есть число элементов (мощность) счётного множества (набора) элементов A; a_i - i-й элемент этого множества (набора).

1. Постановка задачи

Приведем прежде основные используемые определения и обозначения.

Временной ряд (*time series*) обозначаем $T = t_1, t_2, ..., t_{|T|}$, где $t_1, t_2, ..., t_{|T|}$ - действительные числа. Полагаем, что точки ряда $t_1, t_2, ..., t_{|T|}$ расположены во времени в прямом порядке через равные промежутки.

Фрагментом временного ряда T (*subsequence*) называем набор $S_i^{|S|} = t_i, t_{i+1}, ..., t_{i+|S|-1}$ последовательных точек этого ряда; $|S| \le |T|$. Здесь $i \in [1:|T| - |S| + 1]$ – начальная позиция фрагмента. Все возможные фрагменты $\mathbf{S}^{|S|} = \left\{ S_i^{|S|}, i \in [1:|T| - |S| + 1] \right\}$ ряда T могут быть извлечены из этого ряда с помощью *скользящего окна* (*sliding window*) длиной $|S|$. Там, где это не вызывает опасности неверного чтения, индексы $i, |S|$ в обозначении фрагмента временного ряда будем опускать.

Меру отличия между рядами T_1, T_2 одинаковой длины называем *расстоянием* между этими рядами и обозначаем $d(T_1, T_2) \ge 0$. Полагаем, что функция расстояния является симметричной: $d(T_1, T_2) = d(T_2, T_1)$. Аналогично определяем расстояние $d(S_1, S_2) \ge 0$ между фрагментами S_1, S_2 одинаковой длины.

Расстояние между рядом T и его фрагментом S обозначаем $d(T, S) \ge 0$ и определяем как

$$d(T, S) = \min_{S'} d(S', S), \quad S' \in \mathbf{S}^{|S|},$$

где $|S'| = |S|$. Другими словами, расстояние $d(T, S)$ интерпретируем как расстояние между фрагментом S и его лучшим представлением в ряду T.

Для оценки качества разделения набора временных рядов на классы используем метрики энтропии и «прибыли» информации, применяющиеся в классическом алгоритме дерева решений [13].

Пусть исходное множество рядов $\mathbf{T} = \{T_i, i \in [1:|\mathbf{T}|]\}$ состоит из рядов, принадлежащих набору классов $\mathbf{C} = \{C_j, j \in [1:|\mathbf{C}|]\}$, и класс C_j формирует подмножество $\mathbf{T}_{C,j}$, так что $\bigcup_{j=1}^{|\mathbf{C}|}\mathbf{T}_{C,j}=\mathbf{T}$ и $\mathbf{T}_{C,k}\bigcap\mathbf{T}_{C,l}=\varnothing$ для любых $k,l\in[1:|\mathbf{C}|]$, $k\neq l$. *Энтропию* множества $\mathbf{T}$ определяем как

$$I(\mathbf{T})=-\sum_{j=1}^{|\mathbf{C}|}\frac{|\mathbf{T}_{C,j}|}{|\mathbf{T}|}\log\left(\frac{|\mathbf{T}_{C,j}|}{|\mathbf{T}|}\right), \tag{1}$$

где $\frac{|\mathbf{T}_{C,j}|}{|\mathbf{T}|}$ – доля объектов класса C_j в множестве $\mathbf{T}$; $|\mathbf{T}_{C,j}|$ - число рядов в подмножестве $\mathbf{T}_{C,j}$.

В случае бинарной классификации каждая стратегия классификации Sp разделяет множество данных $\mathbf{T}$ на подмножества $\mathbf{T}_1,\mathbf{T}_2$. Энтропия множества $\mathbf{T}$ после его разделения на эти подмножества определяется по формуле, аналогичной формуле (1), как средневзвешенная энтропия подмножеств $\mathbf{T}_1,\mathbf{T}_2$:

$$\hat{I}(\mathbf{T})=\frac{|\mathbf{T}_i|}{|\mathbf{T}|}I(\mathbf{T}_1)+\frac{|\mathbf{T}_2|}{|\mathbf{T}|}I(\mathbf{T}_2).$$

Прибыль (*information gain*) для бинарной стратегии классификации *Sp* определяем формулой

$$g(Sp)=I(\mathbf{T})-\hat{I}(\mathbf{T})=I(\mathbf{T})-\frac{|\mathbf{T}_i|}{|\mathbf{T}|}I(\mathbf{T}_1)-\frac{|\mathbf{T}_2|}{|\mathbf{T}|}I(\mathbf{T}_2).$$

Не формально, *шейплетом S* называется такой фрагмент ряда, для которого большинство объектов одного класса исходного множества $\mathbf{T}$ имеют малое значение расстояния $d(T,S)$ («содержат» в себе шейплет), а большинство объектов другого класса имеют большой значение этого расстояния («не содержат» шейплет).

Одним из основных понятий в методе шейплетов является понятие *оптимальная точка разделения* (*Optimal Split Point, OSP*). Пусть исходное множество рядов $\mathbf{T}$ состоит из рядов, принадлежащих классом C_1,C_2, то есть пусть $|\mathbf{C}|=2$. Пусть также тем или иным способом выбран кандидат *S* и задано некоторое пороговое значения расстояния $\hat{d}$. Разделим множество $\mathbf{T}$ на подмножества $\mathbf{T}_1,\mathbf{T}_2$ так, чтобы для каждого ряда $T_{1,i}\in\mathbf{T}_1$, расстояние $d(T_{1,i},S)$ было не больше величины $\hat{d}$, а для каждого ряда $T_{2,i}\in\mathbf{T}_2$, расстояние $d(T_{2,i},S)$ было не меньше этой величины. Оптимальная точка разделения $OSP(\mathbf{T},S)$ для кандидата *S* – это пороговое расстояние $d_{OSP(\mathbf{T},S)}$, для соответствующей прибыли которого справедливо неравенство

$$g(S,d_{OSP(\mathbf{T},S)})\geq g(S,\hat{d}).$$

Шейплетом $shapelet(\mathbf{T})$ множества данных $\mathbf{T}$, состоящего из классов C_1, C_2, называется фрагмент ряда, для оптимальной точки разделения которого $d_{OSP(\mathbf{T},\, shapelet(\mathbf{T}))}$ имеет место неравенство

$$g(shapelet(\mathbf{T}), d_{OSP(\mathbf{T},\, shapelet(\mathbf{T}))}) \geq g(S, d_{OSP(\mathbf{T},\, S)}),$$

где S - любой фрагмент ряда $T_i \in \mathbf{T}$; $i \in [1:|\mathbf{T}|]$.

В общей постановке шейплетом может быть любой временной ряд, длина которого не превышает длину рядов в исходном наборе. Однако мы полагаем, как это обычно делается в теории и приложениях шейплетов, что шейплет является фрагментом некоторого ряда, содержащегося в исходном наборе. Для уменьшения числа возможных кандидатов-шейплетов определим целочисленные константы *MINLEN* и *MAXLEN*, ограничивающие диапазон длин кандидатов.

Таким образом, задача классификации множества рядов $\mathbf{T}$, состоящего из рядов, принадлежащих классам C_1, C_2, заключается в определении в некотором смысле оптимальной стратегии разделения, то есть оптимальных шейплета $shapelet(\mathbf{T})$ и порогового расстояния $d_{OSP(\mathbf{T},\, shapelet(\mathbf{T}))}$.

2. Алгоритм бинарной классификации на основе шейплетов

2.1. Исходный алгоритм

Псевдокод алгоритма бинарной классификации временных рядов (*Brute-Force Algorithm, BFA*), в том виде, как он предложен в работе [2], представлен на рисунке 1 (некоторые детали алгоритма опущены).

```
FindingShapeletBF(dataset T, MAXLEN, MINLEN)
   candidates ← GenerateCandidates(T, MAXLEN, MINLEN) MINLEN)
   bsf_gain ← 0
   for each S in candidates
      gain ← CheckCandidate(T, S)
      if gain > bsf_gain
         bsf_shapelet ← S
         bsf_gain ← gain
      endif
   endfor
return bsf_shapelet, bsf_gain
```

Рисунок 1 - Псевдокод *BFA*-алгоритма

Процедура GenerateCandidates$(\cdot)$ генерирует фрагменты всех возможных длин и сохраняет их в неупорядоченном списке *candidates*. После инициализации лучшей прибыли *bsf_gain* нулем, алгоритм циклически проверяет, насколько хорошо каждый кандидат *S* из списка *candidates* разделяет множество рядов $\mathbf{T}$ на два класса. С этой целью для каждого кандидата *S* вызывается процедура CheckCandidate$(\cdot)$, которая определяет прибыль от разделения множества $\mathbf{T}$ данным кандидатом. Если прибыль от использования текущего кандидата больше, чем наилучшая достигнутая прибыль, то алгоритм обновляет лучший шейплет *bsf_shapelet* и указанную прибыль. По окончании итераций, алгоритм возвращает лучший шейплет и соответствующую прибыль.

Представленный алгоритм крайне расточителен в плане использования как памяти ЭВМ, так и процессорного времени. Объем памяти, требуемый для хранения всех кандидатов, равен $O(\overline{m}^2 k)$, где $\overline{m}$ – средняя длина ряда в множестве $\mathbf{T}$, k – число объектов в этом множестве. Время на обработку одного кандидата составляет $O(\overline{m}k)$.

2.2 Оптимизации исходного алгоритма с помощью кэширования

Подавляющую часть вычислительных затрат *BFA*-алгоритма составляют затраты на вычисление расстояний $d(T, S)$ от кандидата *S* до данного временного ряда *T*, которые необходимы для определения прибыли от разделения множества $\mathbf{T}$ данным кандидатом (с помощью процедуры CheckCandidate$(\cdot)$). Поэтому оптимизация *BFA*-алгоритма сводится, в первую очередь, к ускорению именно этих вычислений.

Алгоритм *BFA* не фиксирует используемые метрики расстояний между рядами и фрагментами. Это обстоятельство позволяет для оптимизации алгоритма использовать наиболее вычислительно эффективные метрики. В работе [3] в качестве такой метрики предложено использовать метрику, основанную на евклидовом расстоянии между рассматриваемыми временным рядом и кандидатам. Такая метрика, кроме того, позволяет обеспечить инвариантность алгоритма к масштабированию и сдвигу исходных временных рядов [3]. Другие возможные метрики рассмотрены, например, в работе [10].

Для простоты записи обозначим сравнимые ряды как $X = (x_i, i \in [1:m])$, $Y = (y_j, j \in [1:m])$, где m – длины рядов. В этих обозначениях нормализованное евклидово расстояние $d(X, Y)$ между этими рядами может быть вычислено по формуле

$$d(X, Y) = \sqrt{2\,(1 - Corr\,(X, Y))},$$

где $Corr\,(X, Y) = \dfrac{Cov(X, Y)}{m\,\sigma_X\ \sigma_Y}$ - коэффициент корреляции;

$$Cov\,(X, Y) = \sum_{i=1}^{m} x_i\, y_i - m\,\mu_X\ \mu_Y$$ - коэффициент ковариации;

$\sigma_V^2 = \frac{1}{m}\sum_{i=1}^{m} v_i^2 - \mu_V^2$, $\mu_V = \frac{1}{m}\sum_{i=1}^{m} v_i$ - среднеквадратичное отклонение и среднее арифметическое ряда V соответственно; $V \in \{X, Y\}$.

Представленные формулы показывают, что для вычисления расстояния $d(X, Y)$ между рядами X, Y необходимо вычислить суммы

$$\sum_{i=1}^{m} x_i \ , \quad \sum_{i=1}^{m} y_i \ , \quad \sum_{i=1}^{m} x_i^2 \ , \quad \sum_{i=1}^{m} y_i^2 \ , \quad \sum_{i=1}^{m} x_i \, y_i \ .$$

Идея ускорения вычислений расстояний между всеми возможными фрагментами рядов X, Y основана на предварительном вычислении и сохранении в соответствующих массивах следующих кумулятивных сумм: одномерные массивы сумм $\mathbf{S}_X, \mathbf{S}_Y$ членов рядов X, Y; аналогичные массивы сумм $\mathbf{S}_{X^2}, \mathbf{S}_{Y^2}$ квадратов членов этих рядов; двумерный массив сумм $\mathbf{S}_{XY}$ всех возможных парных произведений членов рядов.

Инициируем все левые границы указанных массивов нулями:

$$\Sigma_X[0] = 0; \quad \Sigma_Y[0] = 0; \quad \Sigma_{X^2}[0] = 0; \quad \Sigma_{Y^2}[0] = 0;$$

$$\Sigma_{XY}[u, 0] = 0, \ u \in [1:m]; \quad \Sigma_{XY}[0, v] = 0, \ v \in [1:m].$$

Для всех $u \in [1:m]$, $v \in [1:m]$ заполним эти массивы по следующим правилам:

$$\Sigma_X[u] = \sum_{i=0}^{u} x_i; \quad \Sigma_Y[v] = \sum_{i=0}^{v} y_i; \quad \Sigma_{X^2}[u] = \sum_{i=0}^{u} x_i^2; \quad \Sigma_{Y^2}[v] = \sum_{i=0}^{v} y_i^2;$$

$$\Sigma_{XY}[u, v] = \begin{cases} \sum_{i=0}^{v} x_{i+k} y_i, & u > v, \\ \sum_{i=0}^{u} x_i y_{i+k}, & u \le v. \end{cases}$$

Здесь $k = abs\,(u - v)$.

В результате для любых заданных начальных позиций u, v рассматриваемых фрагментов рядов X, Y и заданной длины фрагментов l соответствующие величины $\mu_X, \mu_Y, \sigma_X, \sigma_Y, \ \sum_{i=0}^{l-1} x_{u+i} \, y_{v+i}$ могут быть вычислены по формулам

$$\mu_X(u, l) = \frac{\Sigma_X[u + l - 1] - \Sigma_X[u - 1]}{l}, \quad \mu_Y(v, l) = \frac{\Sigma_Y[v + l - 1] - \Sigma_Y[v - 1]}{l},$$

$$\sigma_X(u, l) = \frac{\Sigma_{X^2}[u + l - 1] - \Sigma_{X^2}[u - 1]}{l} - \mu_X^2,$$

$$\sigma_Y(v, l) = \frac{\Sigma_{Y^2}[v + l - 1] - \Sigma_{Y^2}[v - 1]}{l} - \mu_Y^2,$$

$$\sum_{i=0}^{l-1} x_{u+i}\ y_{v+i} = \Sigma_{XY}[u+l-1, v+l-1] - \Sigma_{XY}[u-1, v-1].$$

В работе [3] показано, что так организованные вычисления расстояний между рядами имеют вычислительную сложность $O(\overline{m})$, то есть позволяют сэкономить целый порядок сложности алгоритма. С другой стороны, кэширование вычислений по рассмотренной схеме влечет квадратичные затраты оперативной памяти компьютера (для хранения массива Σ_{XY}). Однако поскольку в настоящее время память не является дефицитным ресурсом компьютера, это обстоятельство, как правило, не является критичным.

2.3. Оптимизации путем отбрасывания кандидатов

Идея такого подхода к оптимизации *BFA*-алгоритма заключается в следующем: если известно, что кандидат *S* является «плохим» (обеспечивает низкую прибыль), то любой кандидат, «похожий» на *S*, тоже, вероятнее всего, будет иметь низкую эффективность, и ресурсоемкая операция оценки его прибыли может быть опущена. Различаем два варианта этого подхода: отбрасывание кандидата на основе предварительной оценки его прибыли; отбрасывание во время вычисления прибыли [3].

Прежде нам понадобятся следующие сведения. В задаче бинарной классификации всякая точка $\hat{d}$ рассекает объекты $\mathbf{T}_C$ каждого из классов $C \in \{C_1, C_2\}$ на «левую» $\mathbf{T}_{C.left} \in \mathbf{T}_1$ и «правую» $\mathbf{T}_{C.right} \in \mathbf{T}_2$ части. Класс *C* называется *классом с левым смещением* (*left-major class*) и *классом с правым смещением* (*right-major class*), если имеют место неравенства

$$\frac{|\mathbf{T}_{C.left}|}{|\mathbf{T}_1|} > \frac{|\mathbf{T}_{C.right}|}{|\mathbf{T}_2|}, \quad \frac{|\mathbf{T}_{C.left}|}{|\mathbf{T}_1|} \le \frac{|\mathbf{T}_{C.right}|}{|\mathbf{T}_2|}$$

соответственно. Другими словами, класс *C* называется классом с левым (правым) смещением, если относительное число объектов во множестве $\mathbf{T}_{C.left}$ больше (меньше или равно) такому же числу во множестве $\mathbf{T}_{C.right}$

Утверждение 1. Перемещение (*трансфер*) точки разделения в сторону смещения класса гарантированно повышает прибыль соответствующего разделения классов [3].

Предварительная оценка прибыли кандидата. Допустим, что в процессе поиска оптимальной стратегии разделения в какой-то момент времени определена тройка $\langle S_b, d_b, g_b \rangle$, где S_b, d_b, g_d - лучший текущий шейплет, оптимальная точка разделения и прибыль соответственно. Положим, что определен некоторый шейплет *S* и его прибыль g, и пусть имеет место неравенство $g << g_b$, то есть шейплет *S* обеспечивает прибыль, значительно более низкую по сравнению с наилучшей. Рассмотрим в качестве кандидата следующий фрагмент S'. Обозначим $r(S,S')$ расстояние между кандидатами S, S'. Легко видеть, что в сделанных предположениях имеет место утверждение

$$d(T,S') \in [d(T,S) - r(S,S'); d(T,S) + r(S,S')]. \quad (1)$$

На основании информация о том, как шейплет S разделил набор данных, можно сделать предположение о том, какие изменения внесёт в это разделение новый кандидат. Соотношение (1) определяет диапазон расстояний, который может затронуть кандидат. Временные ряды, расстояние которых от шейплета S принадлежит данному диапазону, могут быть подвергнуты трансферу и таким образом принести информационную прибыль.

Верхнюю границу прибыли можно оценить путем перебора всех возможных комбинаций таких трансферов для каждого класса. Легко видеть, что алгоритмическая сложность такой схемы оценки верхней границы прибыли равна $O(2^{|C|}|\mathbf{T}|z)$, где z – число рядов попавших в диапазон расстояний (1).

Псевдокод алгоритма подсчёта верхнего порога информационной прибыли представлен на рисунке 2. Здесь r – расстояние между кандидатом-шейплетом и шейплетом из истории; $T_i.dist = r(S,S')$. Алгоритм можно применить для текущего шейплета и набора шейплетов из истории, которая формируется по принципу последнего использования. Алгоритм, представленный на рисунке 2, предполагает, что похожие шейплеты идут друг за другом.

```
MaxPossibleInformationGain(dataset, r)
   I_gain ← 0
   for T_i, T_i+1 where i in 1 to |T|-1 do
      τ ← (T_i.dist + T_i+1.dist)/2
      for each T in T do
         if T.dist > r + τ or T.dist < r + τ then
            continue
         endif
         if d not in majority part then
            transfer d to majority part
         endif
      endfor
      I_gain ← max(I_gain, InformationGain(dataset))
   endfor
return I_gain
```

Рисунок 2 – Псевдокод алгоритм подсчёта верхнего порога информационной прибыли

Отбрасывание во время подсчета прибыли. Вторая стратегия отбрасывания кандидатов на основе оценки их прибыли также использует трансферы точек разделения.

Отличие заключается лишь в том, что в данном случае оценка максимально возможной прибыли происходит в процессе непосредственно вычислений.

После некоторого числа операций по сравнению кандидата-шейплета с набором данных, вычислим максимальную информационную прибыль, которую принесут дальнейшие итерации, если шейплет идеально разделит оставшиеся временные ряды по заданным классам. Если эта прибыль оказывается ниже текущей максимальной прибыли, то данного кандидата можно отбросить.

Данный приём дает хорошие результаты в случае, если заметно чаще имеют место итерации, в которых получается низкая информационная прибыль. Такая ситуация характерна для данных с длинными «хвостами» плотности распределения вероятности. Псевдокод алгоритма подсчёта верхнего порога будущей прибыли представлен на рисунке 3.

```
MaxPossibleFurtherInformationGain(T)
   for each T in T do
      if T not in majority part then
           transfer T to majority part
      endif
   endfor
return InformationGain(T)
```

Рисунок 3 – Псевдокод алгоритм подсчёта верхнего порога информационной прибыли

3. Расширения алгоритма

Представленные выше оптимизации делают исходный алгоритм достаточно быстрым, однако область его применения по-прежнему ограничивается лишь бинарной классификацией одномерных наборов данных. Рассматриваемые в данном разделе техники позволяют с помощью незначительных доработок существенно расширить функциональность исходного алгоритма, сделать его более пригодным для применения в реальных приложениях.

Разделение многоклассовых наборов. Основное ограничение исходного алгоритма - поддержка только бинарной классификации - легко преодолевается с помощью, например, дерева решений (*decision-tree*). Действительно, поскольку каждый шейплет является бинарным классификатором для исходного набора рядов, для получения многоклассового разделения эти классификаторы можно применять совместно с древовидной структурой, в которой каждый узел является бинарным классификатором [13].

Логические шейплеты. Опыт классификации временных рядов с помощью шейплетов показывает, что существуют наборы рядов, в которых применение одного шейплета может оказаться недостаточным. Этот недостаток метода преодолевается путем агрегирования рассмотренных *моношейплетов* с помощью простейших логические операций [3]. Помимо этого логические шейплеты дают возможность найти выразительные признаки класса в компактных фрагментах ряда и получить агрегированный шейплет меньшей длины, чем моношейплет, найденный по исходной технике.

Для реализации логических шейплетов, в исходном алгоритме требуется всего одна доработка: в качестве расстояния $d(T, S)$ для логической конъюнкции (*AND*) шейплетов $S_1, S_2,.., S_m$ нужно взять максимальное из расстояний $d(T, S_i)$, $i \in [1:m]$, а для логической дизъюнкции (*OR*) – минимальное из этих расстояний

Многомерные шейплеты. В этом случае шейплет содержит не один фрагмент, а массив фрагментов рассматриваемого многомерного ряда. Алгоритм классификации при этом сравнивает по рассмотренным в п. 3 правилам каждый фрагмент указанного массива с соответствующим измерением ряда или его фрагмента. Итоговое значение расстояния рассчитывается на основе полученных одномерных расстояний с помощью какой-либо метрики, например, путем простого суммирования этих расстояний. Очевидно, что временная сложность алгоритма при этом вырастает на порядок размерности многомерных рядов.

4. Программное обеспечение

Программа *Shapelet C#*, реализующая алгоритм классификации многомерных временных рядов, написана на языке программирования высокого уровня *C#* с использованием платформы *.NET* v4.5. Данный язык по сравнению с языками *C* и *C*++ предполагает более высокий уровень абстракции, что ускоряет разработку программного обеспечения. Платформа *.NET* обеспечивает универсальность, портативность, совместимость, тесную интеграцию разработываемого программного обеспечения с используемой средой разработки (*IDE*) *Visual Studio* 2012 *Express*. В настоящее время эта среда является, возможно, лучшей комбинированной средой для программирования. К ее основным преимуществам можно отнести наличие мощного отладчика, продуманный рефакторинг кода (особенно для *C#*), встроенный профилировщик для исследования проблем производительности написанных программ. Кроме того, как и все продукты *Visual Studio* данная среда имеет в своем составе уникальную систему I*ntelliSense* [14], которая производит анализ кода «на лету».

Для визуализации результатов вычислений используется утилита *Gnuplot*, не требующая установки и встраивания в программный продукт. Утилита принимает входные данные в виде текстовых файлов, имеет простой синтаксис и гибкую систему настроек графического вывода.

В качестве аппаратной платформы использован ноутбук среднего уровня, обладающий следующими основными вычислительными характеристиками: процессор архитектуры *Core i*5 с рабочей частотой 2,5 ГГц; оперативная память стандарта *DD*R3 объемом в 4 Гб.

Используемые в программе классы и структуры привдены в приложении.

Интерфейс программы *Shapelet C#*, представлен на рисунке 4. Верхнюю часть окна занимают группы селекторов опций алогритма: оптимизация, настройка логических шейплетов, ограничения поиска. В средней части расположены поля ввода путей рабочих файлов: набор данных, дамп шейплета. Оба поля предусматривают интерактивный ввод путей с помощью стандартного диалогового окна открытия файла операционной системы *Windows* [15]. В нижней части окна расположены группы «статистик» и результатов: набор данных, шейплет, итоговая классификация.

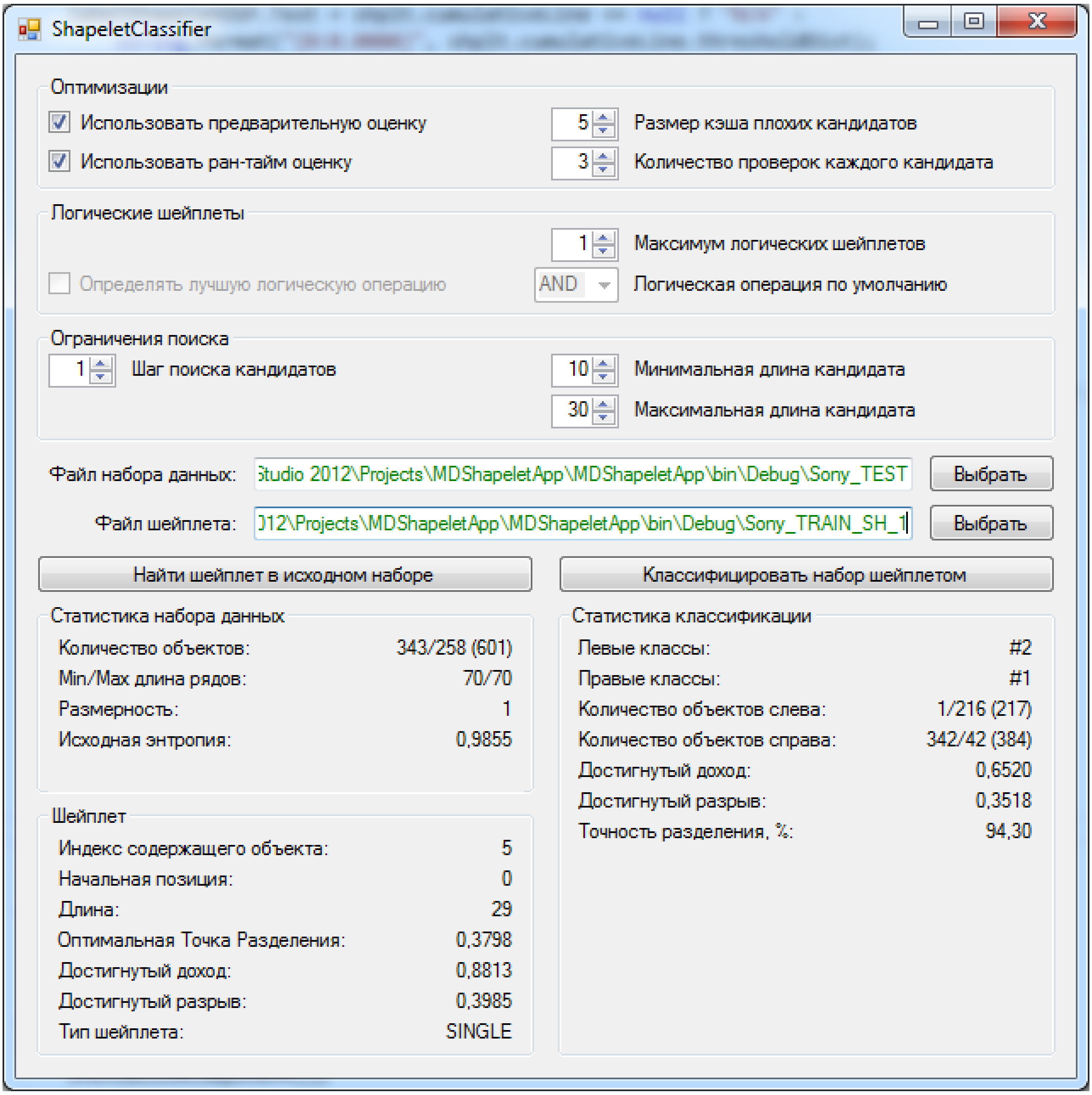

Рисунок 4 - Интерфейс программы *Shapelet C#*

5. Вычислительный эксперимент

Одномерный набор данных. Для исследования эффективности разработанного алгоритмического и программного обеспечения при классификации одномерных временных рядов использован известный набор данных *Sony_TRAIN* [16], который представляет собой записи звуков ходьбы робота-собаки *Sony AIBO* по ковру (*Class* 1) и по бетону (*Class* 2) - рисунок 5.

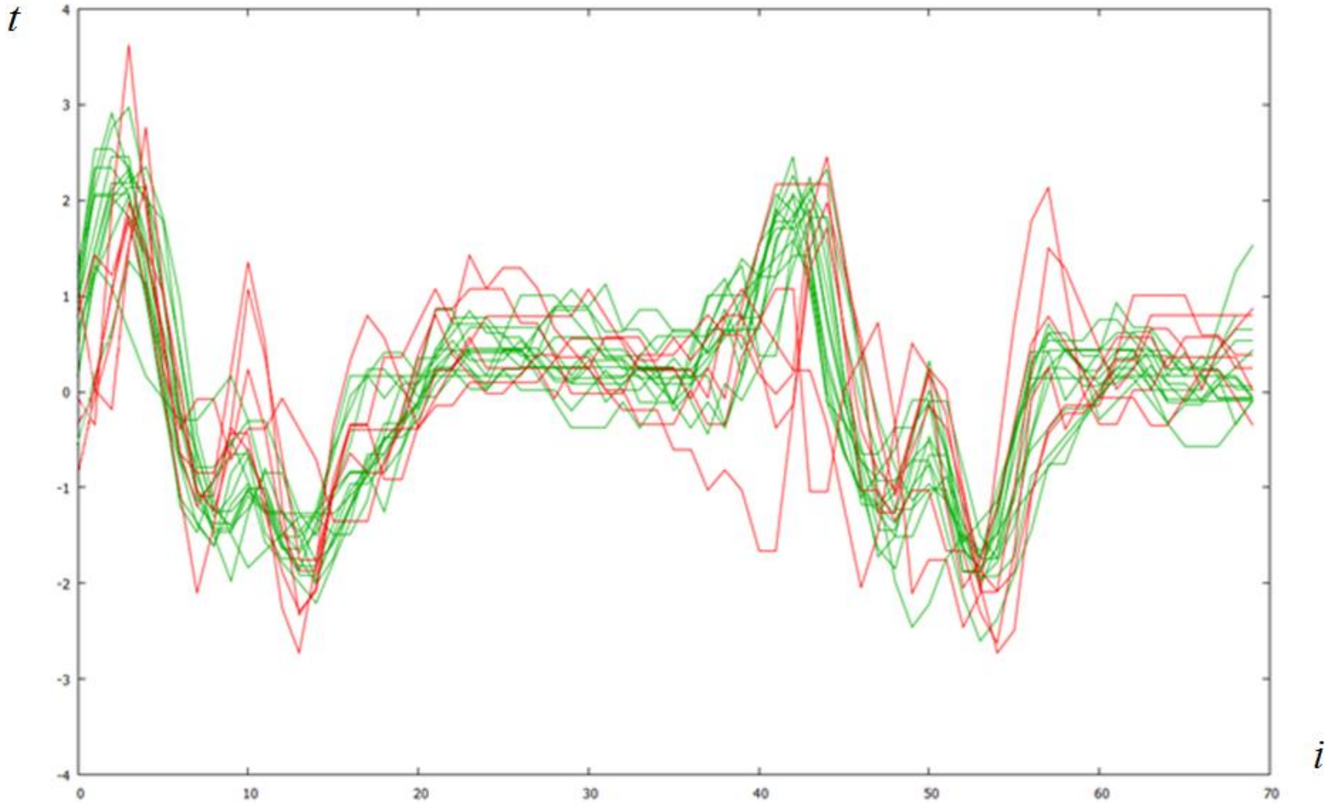

Рисунок 5 - Временные ряды набора *Sony_TRAIN*: зеленое - *Class* 1; красное - *Class* 2

Использованы обучающий и тестовый наборы из 20 и 601 временных рядов соответсвенно (рисунок 6). Из рисунка следует, что шейплет, найденный на обучающем наборе, примерно повторяет форму рядов, относящихся к классу *Class* 2, находясь в середине их разброса.

«Статистику» разделения рядов иллюстрирует таблица 1.

Таблица 1 поазывает, что найденный шейплет обеспечивает «чистое» разделение обучающей выборки на два класса. Результат классификации тестовой выборки также высок - из 601 объектов алгоритм правильно определил 558, из которых 342 и 216 объектов принадлежат первому и второму классам соответственно. Алгоритм продемонстрировал также высокую стойкость к дисперсии входных данных (средний разрыв между классами составлет ~ 0,35).

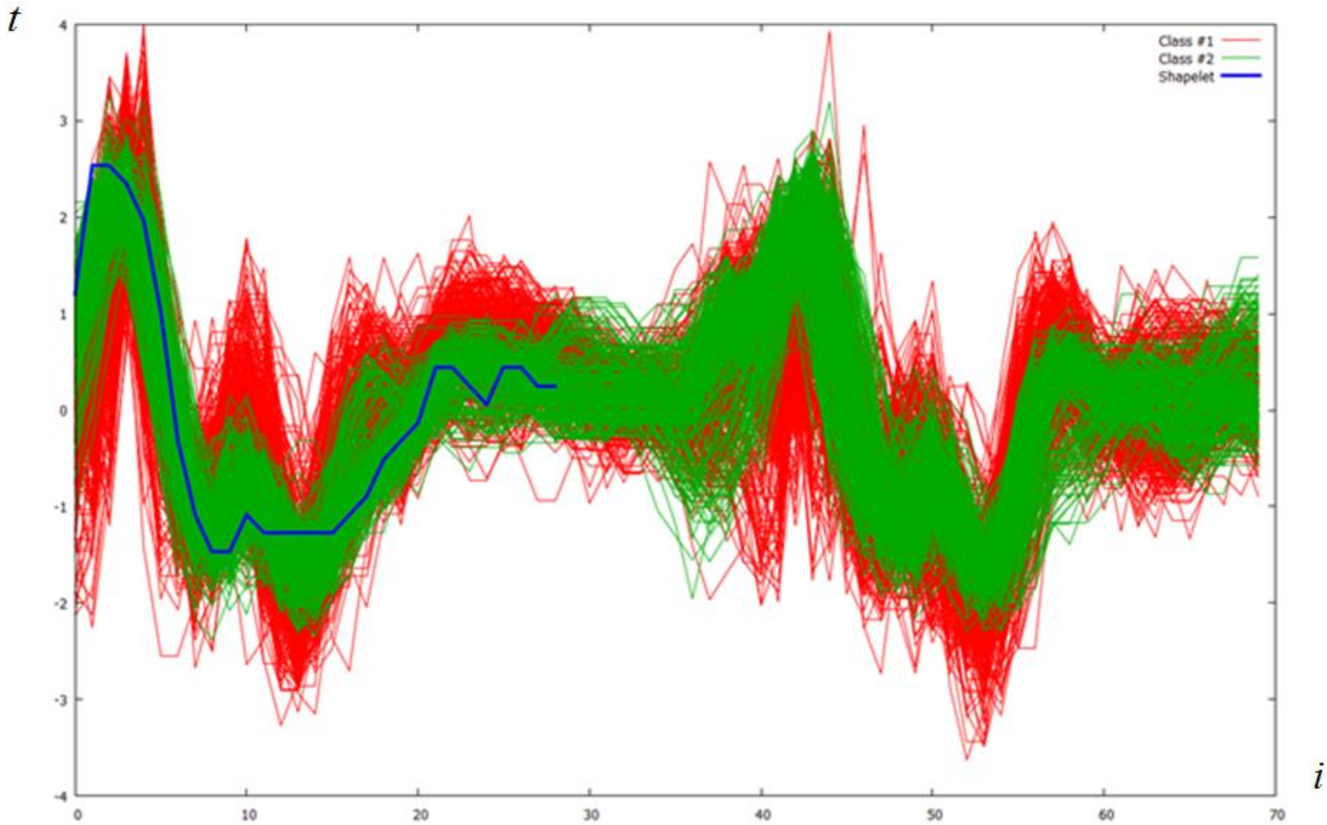

Рисунок 6 - Классификация набора Sony_TEST найденным шейплетом

Таблица 1 – «Статистика» классификации для обучающей и тестовой выборок

Параметр	Обучающая выборка	Тестовая выборка
Число рядов класса Class 1	0/14 (14)	1/216 (217)
Число рядов класса Class 2	6/0 (6)	342/42 (384)
Достигнутый доход	0,88	0,65
Достигнутый разрыв	0,40	0,35
Точность разделения	100,0	94,3

Многомерный набор данных. Использован набор данных ЭЭГ, полученных от 11 датчиков с интервалом в одну минуту в сессии, в ходе которой испытуемому показывались два вида изображений [17]. Поскольку амплитуды потенциалов, снимаемые с различных датчиков, могут сильно отличаться, на рисунке представлены их нормализованные значения.

Протокол обработки программой *Shapelet C#* данного набора представлен на рисунке 7. Рисунок показывает, что из 14 и 16 объектов классов *Class* 1, *Class* 2 верно классифицированы 12 и 14 объектов соответственно. таким образом, точность разделения таким образом составляет около 86%. Время обработки составило 2,5 минуты. Результаты обработки иллюстрирует рисунок 8. Рисунок 9 показывает многомерную ЭЭГ второго класса и найденный шейплет.

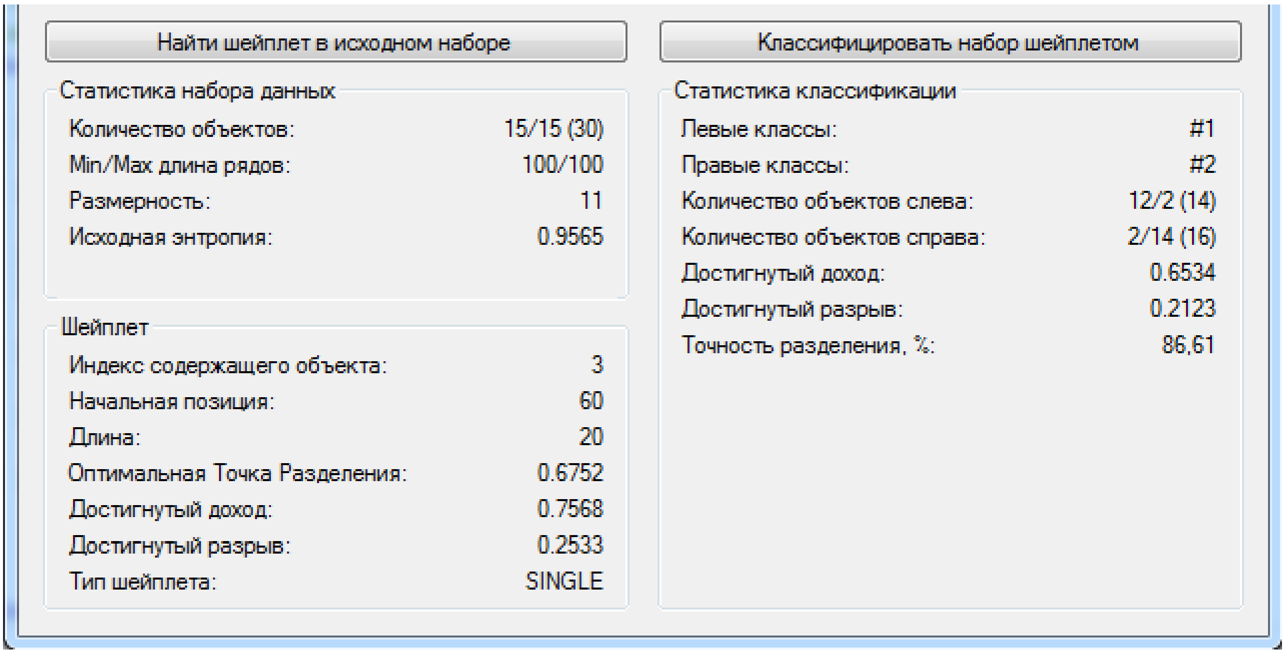

Рисунок 7 – Протокол обработки многомерного набора ЭЭГ

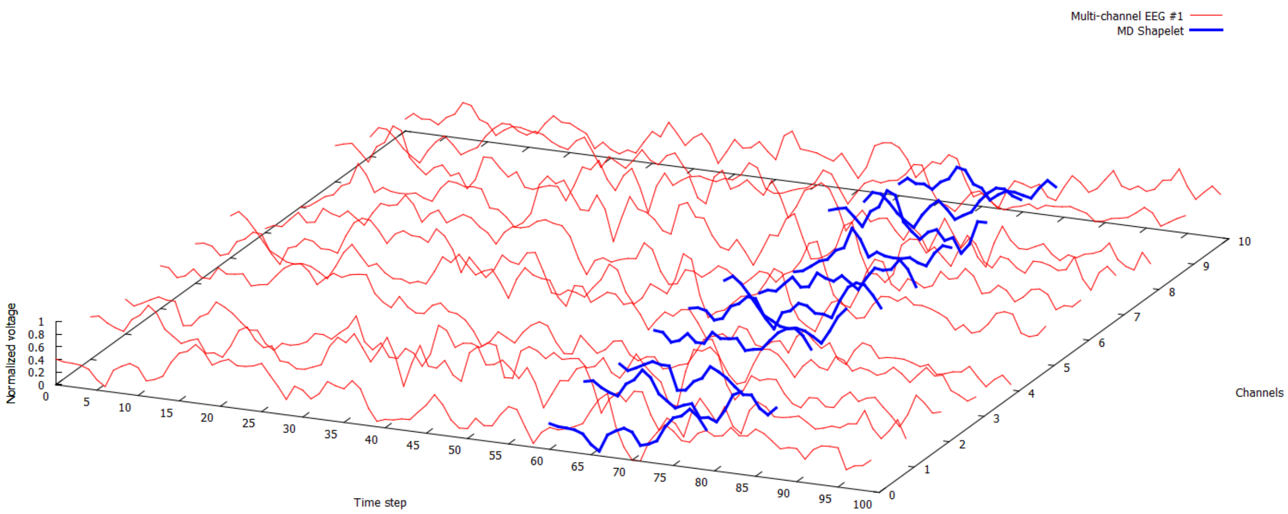

Рисунок 8 - Фрагмент 11-канального временного ряда ЭЭГ и найденный шейплет

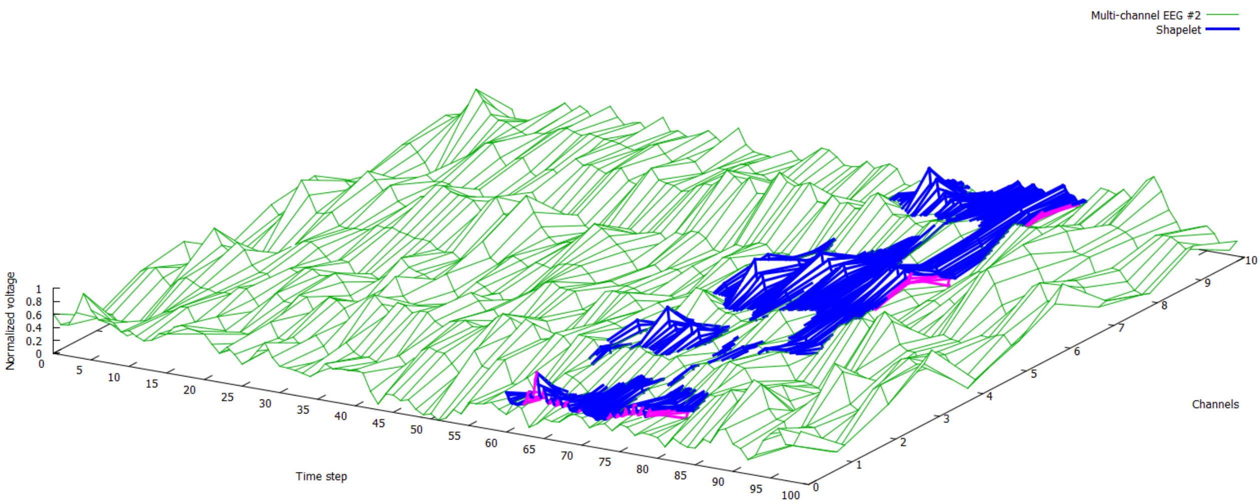

Рисунок 9 - Найденный шейплет и содержащий его ряд

Другие наборы данных [17]. Данные наборы относятся к различным доменам информации и отражают различные физические и статистические явления. В ходе вычислительного эксперимента фиксировались такие параметры, как время поиска шейплета, используемая техника оптимизации, точность классификации набора (таблица 2).

Таблица 2 - Статистика алгоритма для других наборов данных

Набор	Лучшее время	Время, только кэш	Время, только энтропия	Ускор. только кэш	Ускор. только энтропия	Точность, %
CBF	71,5	120,2	89,2	2,2	4,0	83,6
FaceFour	685,8	2599,0	1132,2	3,5	9,3	93,3
Trace	4147,2	20268,5	10843,5	2,1	4,9	80,0
GunPoint	204,6	410,6	320,1	0,8	1,4	81,5
ECG200	469,5	569,6	473,9	3,8	7,6	84,6
Plane	657,7	3120,8	2329,6	4,5	9,4	91,2
Car	1940,3	3426,3	3311,0	2,2	2,5	73,8
Coffee	549,8	1140,4	569,0	2,1	4,9	84,4
Olive Oil	4738,5	22878,0	7029,3	1,2	4,2	75,6
Beef	2660,6	13406,1	4765,6	2,9	9,5	94,8
Symbols	783,2	4832,5	1903,2	4,4	12,8	87,2
Diatom	164,6	1087,4	476,7	1,9	4,9	92,4
Motes	6,2	9,3	8,0	1,8	3,2	92,7
ECG FiveDay	43,7	60,6	43,1	1,0	1,7	81,4
Sony	4,30	5,1	4,3	4,3	7,3	85,1
Haptics	8236,0	24907,1	10858,5	1,9	5,0	75,7

Результаты, представленные в таблице, свидетельствуют о том, что с точки зрения точности классификации рассматриваемый алгоритм может конкурировать с такими известными алгоритмами классификации временных рядов, как *kNN* (алгоритм *k* ближайших соседей) и *МОВ* (алгоритм машины опорных вектров). Имеется ресурс повышения точности классификации за счет подбора значений свободных параметров алгоритма, оптимальных для каждого данного набора временных рядов.

Заключение

Проведенные исследования показали эффективность алгоритма классификации на основе шейплетов как для простых, так и для многомерных данных. Разработанный алгоритм и программная сисетма обеспечивают точность классификации, составляющую, в лучших случаях, около 85%. Время, затрачиваемое на поиск шейплета, возрастает пропорционально размерности входных данных. Точность классификации также зависит от выбора метрики многомерного расстояния между размерностями ряда.

В развитие работы авторы планируют комбинирование метода шейплетов со спайковой нейросетью [18]. Алгоритмы на основе метода шейплетов обладают высоким потенциалом эффективного распараллеливания для различных классов параллельных вычислительных систем [19]. Поэтому авторы также планируют исследования в этой области.

Работа выполнена при поддержке Минобрнауки России (проект RFMEF157714X0135).

Список литературы

1. Трофимов А.Г., Скругин В.И. Адаптивный классификатор многомерных нестационарных сигналов на основе анализа динамических паттернов // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2010. № 8. Режим доступа: http://technomag.bmstu.ru/doc/151934.html (дата обращения 01.10.2015).
2. Ye L., Keogh E. Time series shapelets: a new primitive for data mining // Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD’09). ACM Publ., 2009. P. 947-956. DOI: 10.1145/1557019.1557122
3. Mueen A., Keogh E., Young N. Logical-Shapelets: An Expressive Primitive for Time Series Classification // Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’11). ACM Publ., 2011. P. 1154-1162. DOI: 10.1145/2020408.2020587
4. Gordon D., Hendler D., Rokach L. Fast Randomized Model Generation for Shapelet-Based Time Series Classification. 2012. Available at: http://arxiv.org/abs/1209.5038 , accessed 01.10.2015.
5. He Q., Zhuang F., Shang T., Shi Z. Fast Time Series Classification Based on Infrequent Shapelets // Proceedings of 11th International Conference on Machine Learning and Applications (ICMLA). Vol. 1. IEEE Publ., 2012. P. 215-219. DOI: 10.1109/ICMLA.2012.44

6. Lines J., Bagnall A. Alternative Quality Measures for Time Series Shapelets. Springer Berlin Heidelberg, 2012. P. 475-483. DOI: 10.1007/978-3-642-32639-4_58 (Ser. Lecture Notes in Computer Science; vol. 7435.).
7. Lines J., Davis L., Hills J., Bagnall A. A Shapelet Transform for Time Series Classification // Proceedings of the 18th International Conference on Knowledge Discovery in Data and Data Mining (KDD'12). ACM Publ., 2012. P. 289-297. DOI: 10.1145/2339530.2339579
8. Zakaria J., Mueen A., Keogh E. Clustering Time Series using Unsupervised-Shapelets // Proceedings of the 12th IEEE International Conference on Data Mining (ICDM'2012). IEEE Publ., 2012. P. 785-794. DOI: 10.1109/ICDM.2012.26
9. Rakthanmanon T., Keogh E. Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets // Proceedings of the 13th SIAM International Conference on Data Mining (SDM'13). SIAM Publ., 2013. P. 668-676. DOI: 10.1137/1.9781611972832.74
10. Grabocka J., Schilling N., Wistuba M., Schmidt-Thieme L. Learning Time-Series Shapelets // Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'14). ACM Publ., 2014. P. 392-401. DOI: 10.1145/2623330.2623613
11. Shapelet based time series classification // The University of East Anglia (UEA): website. Available at: https://www.uea.ac.uk/computing/machine-learning/shapelets , accessed 01.10.2015.
12. Чучуева И.А. Модель экстраполяции временных рядов по выборке максимального подобия // Информационные технологии. 2010. № 12. С. 43-47.
13. Левитин А.В. Алгоритмы: введение в разработку и анализ: пер. с англ. М.: Вильямс, 2006. 576 с.
14. Левинсон Дж.Дж. Тестирование ПО с помощью Visual Studio 2010: пер. с англ. М.: ЭКОМ Паблишерз, 2012. 336 с.
15. Нейгел К., Ивьен Б., Глинн Дж., Уотсон К., Скиннер М. C# и платформа .NET 4 для профессионалов: пер. с англ. М.: Вильямс, 2011. 1440 с.
16. Vail D., Veloso M. Learning from accelerometer data on a legged robot // Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, 2004. Available at: http://www.cs.cmu.edu/~coral/old/publinks/mmv/04iav-doug.pdf , accessed 01.10.2015.
17. Software for Brain Computer Interfaces and Real Time Neirosiences // OpenVibe: website. Available at: http://openvibe.inria.fr/datasets-downloads/, accessed 01.10.2015.
18. Paugam-Moisy H., Bohte S.M. Computing with spiking neuron networks // In: Handbook of Natural Computing / ed. by G. Rozenberg, T. Bäck, J.N. Kok. Springer-Verlag Berlin Heidelberg, 2012. P. 335-376. DOI: 10.1007/978-3-540-92910-9_10
19. Chang K-W., Deka, B., Hwu W-M., Roth D. Efficient Pattern-Based Time Series Classification on GPU // Proceedings of the 12th IEEE International Conference on Data Mining (ICDM'2012). IEEE Publ., 2012. P. 131-140. DOI: 10.1109/ICDM.2012.132

Приложение. Используемые классы и структуры

Основой программы *Shapelet C#* является статический класс-интерфейс *ShapeletAlgorithm*, содержащий публичные методы для поиска многомерного логического шейплета (*FindLogicalShapelet*) и для классификации имеющимся шейплетом (*ClassifyBinary*). Также этот класс содержит публичные определения структуры опций алгоритма (*Options*) и структуры данных производительности (*PerformanceData*). С помощью первой структуры осуществляются предварительные настройки алгоритма, с помощью второй – анализ результатов его работы.

Помимо публичных методов, класс *ShapeletAlgorith* содержит приватные методы, запускаемые в теле вышеуказанных методов и предназначенные для выполнения низкоуровневых процедур и структурирования кода.

Основные используемые в программе структуры данных приведены в *UML*-диаграмме (рисунок 9).

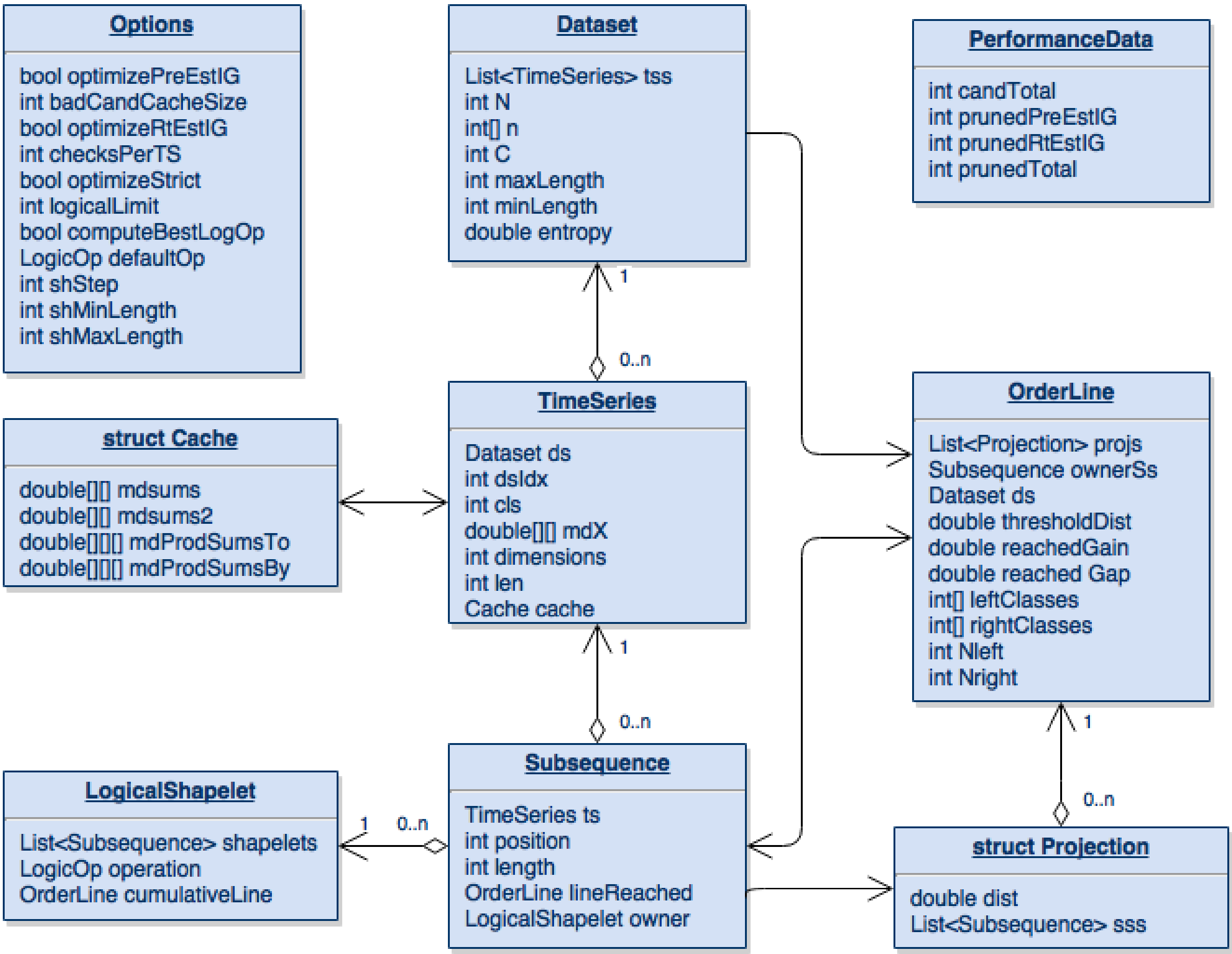

Рисунок 9 - *UML*-диаграмма используемых структур

Класс *TimeSeries* является самой низкоуровневой структурой данных и реализует основной объект обработки – временной ряд. Каждый экземпляр класса *TimeSeries* содержит: сами данные, то есть многомерный массив значений ряда, его длина и размерность; информация о контейнере - ссылка на исходный набор данных, позиция в этом наборе, класс ряда; кэш статистик - многомерные массивы сумм, сумм квадратов и сумм попарных произведений (п. 2). Основными методами класса *TimeSeries* являются: *CalcProdSumsTo* – метод, вычисляющий массивы сумм попарных произведений указанных временных рядов и сохраняющий ссылки на массивы обоих этих рядов; *prepareSumsData* – метод, вычисляющий массивы сумм и сумм квадратов обрабютываемого ряда.

Класс *Subsequence* реализует фрагмент обработываемого временного ряда. Экземпляры класса содержат: данные - ссылка на временной ряд, котлорому принаджлежит фрагмент, длина и начальная позиция фрагмента; ссылки на основной и альтернативный ряды отображений, а также на контейнер этого фрагмента (шейплет). Основные методы класса *Subsequence*: *CanReachBetterIGThan* – метод, оценивающий возможный максимальную прибыль от кандидата и сравнивающий его с текущим кэшем «плохих» кандидатов; *sdistOptimized* – оптимизированный метод, вычисляющий расстояние между фрагментом и рядом. Помимо этого, класс содержит большое число приватных методов для таких низкоуровневых операций, как вычисление расстояния, корреляция, ковариация и т.д. (п. 2).

Класс *LogicalShapelet* реализует логический шейплет (п. 3). Путем установки соответствующего ограничения класс может быть использован для реализации обычных шейплетов. Класс является контейнером для базовых шейплетов. Помимо этого, класс хранит тип логической операции (для логических шейплетов), а также кумулятивный ряд отображений для всех используемых шейплетов. Основные методы класса реализуют функции сохранения и загрузки логического шейплета в файл на жестком диске, а также функция добавления нового шейплета в логический шейплет.

Класс *Dataset* реализует исходный набор данных. Основное назначение класса – хранение временных рядов и их некоторых параметров. Класса включает в себя: данные - массив входящих временных рядов и их число; параметры данных - максимальная и минимальная длины рядов в наборе, энтропия исходного набора (п. 2); информация о классах - чиса классов и экземпляров каждого класса. Единственный метод класса – конструктор, производящий создание исходного набора, начиная с парсинга входного файла с данными и заканчивая вычислением значений указанных параметров.

Класс *OrderLine* включает в себя: данные - ссылки на кандидата и набор данных, относящихся к данной сущности, а также список отображений (проекций) объектов этого набора; параметры текущих вычислений - список объектов с левым и правым смещение и числа этих объектов, списки классов с левым и правым смещением, общее число объектов в наборе; результаты вычислений - пороговое значение расстояния (*OTP*), достигнутую прибыль и разрыв (англ. *gap*) разделения. Методами класса *OrderLine* являются:

конструктор класса, выполняющий всю работу по инициализации отображений; оценщики дохода *MaxPossibleRangedIG*, *MaxPossibleFurtherIG*, позволяющие оценить максимальную прибыль до и во время выполнения инициализации; процедура разделения *MakeBestSplit*, реализующая лучшее разделение, возможное в текущей ситуации, и сохраняющая результаты разделения в переменных класса. Кроме того, класс содержит определение структуры *Projection*, использующейся для реализации отображений (проекций) объектов класса на прямую. Структура содержит в себе позицию отображенного объекта на прямой, а также список фрагментов, которым соответсвует данное отображение.

Структура *Options* содержит значения свободных параметров алгоритма (таблица 3). Вообще говоря, для каждого набора входных данных оптимальные значения указанных параметров индивидуальны. Приведенные умолчательные значения параметров обеспечивают некоторый баланс между универсальностью и качеством достигаемых результатов.

Таблица 3 - Параметры структуры *Options* (свободные параметры алгоритма)

Имя	Тип	Умолч. значение	Описание
optimizePreEstIG	*bool*	True	Использовать предварительную оценку прибыли?
badCandCacheSize	*int*	5	Размер кэша «плохих» кандидатов. Используется вместе с предыдущей опцией.
optimizeRtEstIG	*bool*	true	Использовать оценку прибыли во время вычислений?
checksPerTS	*int*	3	Число проверок во время вычислений. Используется вместе с предыдущей опцией.
optimizeStrict	*bool*	true	Использовать строгие неравенства при оценке?
logicalLimit	*int*	1	Признак логического объединения шейплетов. При значении этой величины большем единицы производится поиск указанного числа логических шейплетов.
computeBestLogOp	*bool*	false	Определять лучшую логическую операцию для логических шейплетов? Используется вместе с предыдущей опцией.
defaultOp	*enum*	AND	Логическая операция по умолчанию. Используется вместе с предыдущей опцией.
shStep	*int*	1	Шаг генерации кандидатов. При использовании значений этой величины бол ьше единицы, скорость поиска пропорционально возрастает, но шейплет отыскивается менее точно.
shMinLength	*int*	-	Минимальная длина генерируемых шейплетов. Рекомендуемое значение зависит от длины исходных данных.
shMaxLength	*int*	-	Максимальная длина генерируемых шейплетов. Рекомендуемое значение зависит от длины исходных данных.

Science & Education of the Bauman MSTU Electronic journal ISSN 1994-0408

Science and Education of the Bauman MSTU, 2015, no. 11, pp. 382–405.

DOI: **10.7463/1115.0827396**

Received: 03.10.2015
Revised: 20.10.2015

Effectiveness of Multivariate Time Series Classification Using Shapelets

A.P. Karpenko[1,*], M.S. Kostrubin2, A.S. Chernyshev[1]

[*]apkarpenko@mail.ru

[1]Bauman Moscow State Technical University, Moscow, Russia
[2]Ltd. "EKSPERTEK IBS", Moscow, Russia

Keywords: classification of time series; time series shapelets

Typically, time series classifiers require signal pre-processing (filtering signals from noise and artifact removal, etc.), enhancement of signal features (amplitude, frequency, spectrum, etc.), classification of signal features in space using the classical techniques and classification algorithms of multivariate data. We consider a method of classifying time series, which does not require enhancement of the signal features. The method uses the shapelets of time series (*time series shapelets*) i.e. small fragments of this series, which reflect properties of one of its classes most of all.

Despite the significant number of publications on the theory and shapelet applications for classification of time series, the task to evaluate the effectiveness of this technique remains relevant. An objective of this publication is to study the effectiveness of a number of modifications of the original shapelet method as applied to the multivariate series classification that is a little-studied problem. The paper presents the problem statement of multivariate time series classification using the shapelets and describes the shapelet–based basic method of binary classification, as well as various generalizations and proposed modification of the method. It also offers the software that implements a modified method and results of computational experiments confirming the effectiveness of the algorithmic and software solutions.

The paper shows that the modified method and the software to use it allow us to reach the classification accuracy of about 85%, at best. The shapelet search time increases in proportion to input data dimension.

References

1. Trofimov A.G., Skrugin V.I. Adaptive classificator of multidimensional non stationary signals based on dynamical patterns analysis. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU*, 2010, no. 8. Available at: http://technomag.bmstu.ru/doc/151934.html , accessed 01.10.2015. (in Russian).

2. Ye L., Keogh E. Time series shapelets: a new primitive for data mining. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'09).* ACM Publ., 2009, pp. 947-956. DOI: 10.1145/1557019.1557122
3. Mueen A., Keogh E., Young N. Logical-Shapelets: An Expressive Primitive for Time Series Classification. *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11).* ACM Publ., 2011, pp. 1154-1162. DOI: 10.1145/2020408.2020587
4. Gordon D., Hendler D., Rokach L. *Fast Randomized Model Generation for Shapelet-Based Time Series Classification*. 2012. Available at: http://arxiv.org/abs/1209.5038 , accessed 01.10.2015.
5. He Q., Zhuang F., Shang T., Shi Z. Fast Time Series Classification Based on Infrequent Shapelets. *Proceedings of 11th International Conference on Machine Learning and Applications (ICMLA). Vol. 1*. IEEE Publ., 2012, pp. 215-219. DOI: 10.1109/ICMLA.2012.44
6. Lines J., Bagnall A. *Alternative Quality Measures for Time Series Shapelets*. Springer Berlin Heidelberg, 2012, pp. 475-483. DOI: 10.1007/978-3-642-32639-4_58 (Ser. *Lecture Notes in Computer Science*; vol. 7435.).
7. Lines J., Davis L., Hills J., Bagnall A. A Shapelet Transform for Time Series Classification. *Proceedings of the 18th International Conference on Knowledge Discovery in Data and Data Mining (KDD'12).* ACM Publ., 2012, pp. 289-297. DOI: 10.1145/2339530.2339579
8. Zakaria J., Mueen A., Keogh E. Clustering Time Series using Unsupervised-Shapelets. *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM'2012).* IEEE Publ., 2012, pp. 785-794. DOI: 10.1109/ICDM.2012.26
9. Rakthanmanon T., Keogh E. Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. *Proceedings of the 13th SIAM International Conference on Data Mining (SDM'13).* SIAM Publ., 2013, pp. 668-676. DOI: 10.1137/1.9781611972832.74
10. Grabocka J., Schilling N., Wistuba M., Schmidt-Thieme L. Learning Time-Series Shapelets. *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'14).* ACM Publ., 2014, pp. 392-401. DOI: 10.1145/2623330.2623613
11. *Shapelet based time series classification*. The University of East Anglia (UEA): website. Available at: https://www.uea.ac.uk/computing/machine-learning/shapelets , accessed 01.10.2015.
12. Chuchueva I.A. The Time Series Extrapolation Model Based on Maximum Likeness Set. *Informacionnye Tehnologii = Information Technologies*, 2010, no. 12, pp. 43-47. (in Russian).
13. Levitin A.V. *Introduction to the Design and Analysis of Algorithms.* Addison-Wesley, 2003. (Russ. ed.: Levitin A.V. *Algoritmy: vvedenie v razrabotku i analiz*. Moscow, Vil'iams Publ., 2006. 576 p.).

14. Levinson J. *Software Testing with Visual Studio 2010*. Addison-Wesley, 2011. 326 p. (Russ. ed.: Levinson J. *Testirovanie PO s pomoshch'yu Visual Studio 2010*. Moscow, EKOM Publishers, 2012. 336 p.).
15. Nagel C.., Evjen B., Glynn J., Watson K., Skinner M. *Professional C# 4.0 and .NET 4*. Wrox, 2010. 1536 p. (Russ. ed.: Nagel C., Evjen B., Glynn J., Watson K., Skinner M. *C# i platforma .NET 4 dlya professionalov*. Moscow, Vil'yams Publ., 2011. 1440 p.).
16. Vail D., Veloso M. Learning from accelerometer data on a legged robot. *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004. Available at: http://www.cs.cmu.edu/~coral/old/publinks/mmv/04iav-doug.pdf , accessed 01.10.2015.
17. *Software for Brain Computer Interfaces and Real Time Neirosiences*. OpenVibe: website. Available at: http://openvibe.inria.fr/datasets-downloads/, accessed 01.10.2015.
18. Paugam-Moisy H., Bohte S.M. Computing with spiking neuron networks. In book: Rozenberg G., Bäck T., Kok J.N., eds. *Handbook of Natural Computing*. Springer-Verlag Berlin Heidelberg, 2012, pp. 335-376. DOI: 10.1007/978-3-540-92910-9_10
19. Chang K-W., Deka, B., Hwu W-M., Roth D. Efficient Pattern-Based Time Series Classification on GPU. *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM'2012).* IEEE Publ., 2012, pp. 131-140. DOI: 10.1109/ICDM.2012.132