DOI: 10.24143/2072-9502-2017-2-62-68
УДК 004.023

Р. Ю. Демина

ОСОБЕННОСТИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМОВ МЕТОДИКИ ФОРМИРОВАНИЯ ОБУЧАЮЩЕГО МНОЖЕСТВА ДЛЯ БИНАРНЫХ КЛАССИФИКАТОРОВ, ИСПОЛЬЗУЕМЫХ В АНТИВИРУСНОМ ЭВРИСТИЧЕСКОМ СТАТИЧЕСКОМ АНАЛИЗЕ

В связи со стремительным распространением средств вычислительной техники в качестве объектов бинарной классификации все чаще выступают компьютерные файлы. Особую роль бинарная классификация файлов играет в антивирусном эвристическом статическом анализе. Процесс классификации состоит из двух этапов: обучения и распознавания. На этапе обучения формируется обучающее множество объектов. Важно проводить данный отбор не случайным образом, а целенаправленно, с учетом разнообразия объектов. Поскольку введение дополнительной процедуры формирования обучающего множества приведет к увеличению общего времени обучения, необходимо учесть все особенности программной реализации, чтобы данный этап прошел максимально быстро. Рассмотрена методика формирования обучающего множества и описаны основные нюансы, которые необходимо учесть для сокращения времени вычислений. Представлен алгоритм расширенного бинарного поиска, предназначенный для формирования отсортированной последовательности уникальных элементов. Рассмотрена основная особенность (способ хранения данных), которая может повлиять на время выполнения алгоритма. Приведен пример кода, реализующего функцию расширенного бинарного поиска на языке высокого уровня C++. Результаты исследования позволят перейти к программной реализации предложенных подходов для их дальнейшего внедрения в системы антивирусной защиты.

Ключевые слова: бинарная классификация, обучающее множество, бинарный поиск, программная реализация.

Введение

Широкий класс систем поддержки принятия решений (поиск неисправностей в сложных технических система, обнаружение вторжений несанкционированных пользователей и т. п.) опирается на задачу бинарной классификации, которая предусматривает отнесение объекта к одному из двух заранее определенных классов [1–3]. В связи со стремительным распространением средств вычислительной техники в качестве объектов бинарной классификации все чаще выступают компьютерные файлы (в системах распознавания изображений или текста) [4]. Особую роль бинарная классификация файлов играет в антивирусном эвристическом статическом анализе. В данном случае результатом классификации является отнесение файла к одному из классов: вредоносных или легитимных. Это позволяет с определенной вероятностью обнаруживать вирусы «нулевого дня», сигнатуры которых еще не выделены специалистами антивирусных лабораторий и которые, соответственно, не успели попасть в базы сигнатур конечных пользователей.

Процесс любой классификации, в том числе бинарной, представляет собой «обучение с учителем» и состоит, как правило, из двух этапов: обучения и распознавания (отнесения к классу).

На *первом шаге этапа обучения* формируется множество объектов, для каждого из которых указано, к какому классу он относится. Обычно это множество формируется случайным образом [5, 6]. При этом должны отсутствовать ошибки в отнесении объектов к определенному классу, поскольку объекты обучающего множества становятся эталоном (образцом) для классификатора. Кроме того, объекты, отобранные для каждого класса, должны образовывать репрезентативную выборку, наиболее полно характеризующую данный класс.

На следующем шаге этапа обучения из объектов обучающего множества извлекаются значения признаков, из которых, в свою очередь, отбираются наиболее информативные (т. е. дающие наибольшее количество информации о классе рассматриваемого объекта, из которого извлечен подобный признак). На отобранных значениях признаков и происходит обучение классификатора.

После того как классификатор обучен, он используется для *распознавания*. На этом этапе на вход классификатора поступает объект, класс которого необходимо определить. Из объекта

извлекаются значения признаков и оценивается, сколько признаков указывают на его принадлежность к какому-либо из классов. Это позволяет сделать вывод о том, к какому классу относится рассматриваемый объект.

Выше уже отмечалось, что процесс классификации начинается с формирования обучающего множества, отбор в которое ведется обычно случайным образом. Однако, как было показано в [7], качество обучающего множества влияет на верность – долю правильно распознанных объектов. Правильный целенаправленный отбор может значительно увеличить качество обучения классификатора. В основе подхода, изложенного в работе [7], лежит следующая идея: общие признаки, найденные среди разнообразных объектов одного класса, в большей степени будут полезны для распознавания, чем общие признаки, найденные среди похожих друг на друга объектов.

Например, пусть возникла задача сформировать обучающее множество вредоносных и доброкачественных файлов. При формировании подмножества вирусных файлов необходимо учитывать не только процентное соотношение троянов, червей, криптолокеров и пр., но и то, что отобранные трояны, черви, криптолокеры должны различаться между собой. Так, например, если в обучающей выборке будет несколько версий одного и того же вируса, это негативно скажется на распознавании других вирусов. Бесконечно наращивать обучающее множество не представляется возможным, потому что время обучения будет соответственно увеличиваться, а для систем реального времени это может быть весьма критичным. Кроме того, подобное дублирование может негативно сказаться на статистическом распределении значений признаков.

Постановка задачи

Для повышения верности бинарного классификатора целесообразно формировать обучающее множество не случайным образом, а целенаправленно, с учетом разнообразия объектов. Следует отметить, что введение дополнительного этапа отбора файлов в обучающее множество увеличивает общее время обучения. Следовательно, необходимо учесть все нюансы программной реализации, чтобы это время сократить до минимума.

Исходя из этого, мы сформулировал основную ***задачу исследования*** – рассмотреть особенности программной реализации методики формирования обучающего множества.

Методы и результаты исследования

Методика формирования обучающего множества. Любой файл представляет собой последовательность байт, поэтому, вне зависимости от его расширения, файл можно представить в виде набора *n*-грамм любой последовательности расположенных подряд *n* байт. Для сравнения файлов в [8] было предложено ввести меру схожести файлов. Мера схожести файла *A* с файлом *B* определяется как отношение числа уникальных *n*-грамм файла *A*, которые встречаются в наборе *n*-грамм файла *B*, к количеству уникальных *n*-грамм файла *A*:

$$\rho(A,B) = \frac{\left|\overline{A} \cap \overline{B}\right|}{\overline{A}},$$

где $\overline{A}$ – множество *n*-грамм файла *A*; $\overline{B}$ – множество *n*-грамм файла *B*.

Пусть *F* – исходное множество файлов, а *F′* – обучающее множество файлов, которое необходимо сформировать из множества *F*. Тогда введенная таким образом мера позволяет построить матрицу схожести для множества файлов *F*: квадратную матрицу, состоящую из элементов ρ_{ij} – мер схожести *i*-го файла с *j*-м.

Матрица схожести служит основой алгоритма отбора максимально различающихся между собой по составу *n*-грамм файлов во множество *F′* для использования на этапе обучения классификаторов ($|F| \geq |F'|$):

1. Исходя из допустимого времени обучения классификатора, целей обучения и алгоритма классификации задать *M* – мощность множества *F′*.

2. Задать пороговое значение меры схожести *K*, при превышении которого сравниваемые файлы считаются схожими.

3. Для каждого *i*-го файла подсчитать параметр L_i – количество *j*-х файлов ($i \neq j$), для которых $\rho_{ij} \geq K$, и тем самым выделить группы взаимно схожих файлов. Файлы с $L_i = 0$ считаются уникальными в рамках данного множества.

4. Из каждой группы произвольно выбрать и оставить только один файл.

5. Включать файлы, отобранные на шаге 4, в итоговую обучающую выборку F' в порядке убывания параметра L до тех пор, пока $|F'| \le M$.

6. Если после выполнения шага 5 $|F'| < M$, то необходимо понизить пороговое значение K и повторить шаги 2–5. В противном случае множество F' считается сформированным и работа алгоритма заканчивается.

Таким образом, методика отбора представляет собой механизм кластеризации, т. е. обучение без учителя. Она позволяет разделить исходный набор вирусных файлов на взаимно сходные классы по определенным признакам в отсутствие информации об используемых ими механизмах вредоносного воздействия.

Отметим, что наиболее затратными по времени являются следующие операции: расчет матрицы схожести и составление отсортированного перечня признаков. В связи с этим необходимо рассмотреть способы сокращения времени работы соответствующих алгоритмов.

Особенности программной реализации алгоритма расчета матрицы схожести. Расчет матрицы схожести характеризуется рядом особенностей, учет которых позволяет сократить время вычислений:

1. Целесообразно предварительно составить отсортированный перечень уникальных n-грамм для каждого файла.

Для вычисления меры схожести двух файлов необходимо считать каждый из них побайтово в оперативную память, представить в виде перечня n-грамм, отсортировать и удалить дубликаты. Нерационально проводить данные операции перед каждым сравнением, поскольку это занимает много времени. Время построения матрицы схожести сократится, если считать все файлы описанным выше образом в оперативную память и при вычислении меры схожести обращаться к уже сформированным массивам, хранящимся в ОЗУ.

Если при разработке программного обеспечения данная рекомендация учитывается, то процедура считывания файла с последующим составлением отсортированного перечня уникальных n-грамм будет происходить s раз, где s – количество файлов. В противном случае – s^2 раз.

Для проверки целесообразности данной рекомендации был проведен вычислительный эксперимент с помощью программного продукта, разработанного на языке C++ в среде Microsoft Visual Studio 2012. Отсчет времени производился с помощью разницы значений, возвращаемых функцией языка C++ «clock» до начала и после окончания вычислений в каждом из экспериментов. При этом использовалась ПЭВМ со следующими характеристиками: процессор Intel(R) Core(TM) i3 CPU M330 @2.13GHz 2.13GHz; ОЗУ 3,00 Гб.

Было разработано две версии программного продукта, реализующего описанную выше методику формирования обучающего множества: с учетом данной особенности и без учета каких-либо нюансов. На вход обеих программ поступало фиксированное множество файлов объемом 1 000 шт., из которого было необходимо сформировать обучающее множество объемом 700 файлов.

Версия, не учитывающая никаких рекомендаций, выполняла отбор в течение 13 минут 29 секунд. Версия, которая предварительно считывала в память файлы и формировала для каждого из них отсортированный перечень уникальных n-грамм, затратила на работу 10 минут 48 секунд.

2. Целесообразно одновременно рассчитывать $\rho(A, B)$ и $\rho(B, A)$.

Для составления матрицы схожести необходимо провести s^2 расчетов мер схожести. Каждый раз, когда для определения меры схожести берутся два файла – A и B, необходимо рассчитать для них количество общих n-грамм. Подсчет количества общих n-грамм – самая затратная по времени операция, при этом полученное значение применяется и для расчета $\rho(A, B)$, и для расчета $\rho(B, A)$, поэтому будет рациональным после подсчета количества общих n-грамм файлов A и B вычислять сразу и $\rho(A, B)$, и $\rho(B, A)$.

Для проверки целесообразности рекомендации об одновременном расчете $\rho(A, B)$ и $\rho(B, A)$ был проведен вычислительный эксперимент, аналогичный описанному выше. Версия, учитывающая данную рекомендацию, выполняла отбор в течение 11 минут 33 секунд, а версия, не учитывающая никакие рекомендации, – в течение 13 минут 29 секунд.

3. Целесообразно использовать многопоточность.

Матрицу схожести $M = \{m_{ij}\}$ можно представить в виде 4 частей.

$$M = \begin{matrix} M_1 & M_2, \\ M_3 & M_4, \end{matrix}$$

где $M_1 : m_{ij},\ i = 0...\frac{s}{2} - 1,\ j = 0...\frac{s}{2} - 1;$ $M_2 : m_{ij},\ i = 0...\frac{s}{2} - 1,\ j = \frac{s}{2}...s;$ $M_3 : m_{ij},\ i = \frac{s}{2}...s,\ j = 0...\frac{s}{2} - 1;$

$M_4 : m_{ij},\ i = \frac{s}{2}...s,\ j = \frac{s}{2}....s.$

Для расчета значений мер схожести из M_1 и M_4 целесообразно выделить два отдельных потока – для расчета каждой из частей в отдельности. Это связано с тем, что для M_1 и M_4 справедливо следующее: m_{ij} и m_{ji} принадлежат одной и той же части матрицы схожести при любых i и j. При этом достаточно рассчитывать меры схожести только для файлов выше/ниже главной диагонали соответствующей части матрицы, т. к. остальные значения определятся автоматически при соблюдении пункта об одновременном расчете $\rho(A, B)$ и $\rho(B, A)$.

Части M_2 и M_3 взаимосвязаны, т. к. m_{ij} и m_{ji} принадлежат разным частям матрицы схожести при любых i и j. При этом рассчитывать меры схожести можно только для одной из частей, т. к. соответствующие значения из другой части определятся автоматически при соблюдении пункта об одновременном расчете $\rho(A, B)$ и $\rho(B, A)$.

Таким образом, для ускорения расчета матрицы схожести необходимо выделить три потока: для M_1, для M_2 и M_3 и для M_4.

Для проверки целесообразности внедрения многопоточности вычислений был проведен эксперимент в условиях, аналогичных описанным выше. Версия программного обеспечения, в рамках которой было реализовано три потока, потратила на отбор 10 минут 17 секунд, версия, не учитывающая никакие рекомендации, – 13 минут 29 секунд.

Таким образом, применение вышеизложенных приемов позволяет сократить время расчета матрицы схожести.

Использование программы, в которой были реализованы все предложенные приемы, позволило сократить совокупное время расчетов, проведенных в описанных выше условиях, в 3,16 раза (с 13 минут 29 секунд до 4 минут 16 секунд).

Для сокращения времени формирования отсортированного перечня уникальных элементов, который необходимо получать как на этапе формирования обучающего множества, так и на этапе составления исходного перечня значений признаков, целесообразно применить алгоритм расширенного бинарного поиска (РБП) [9].

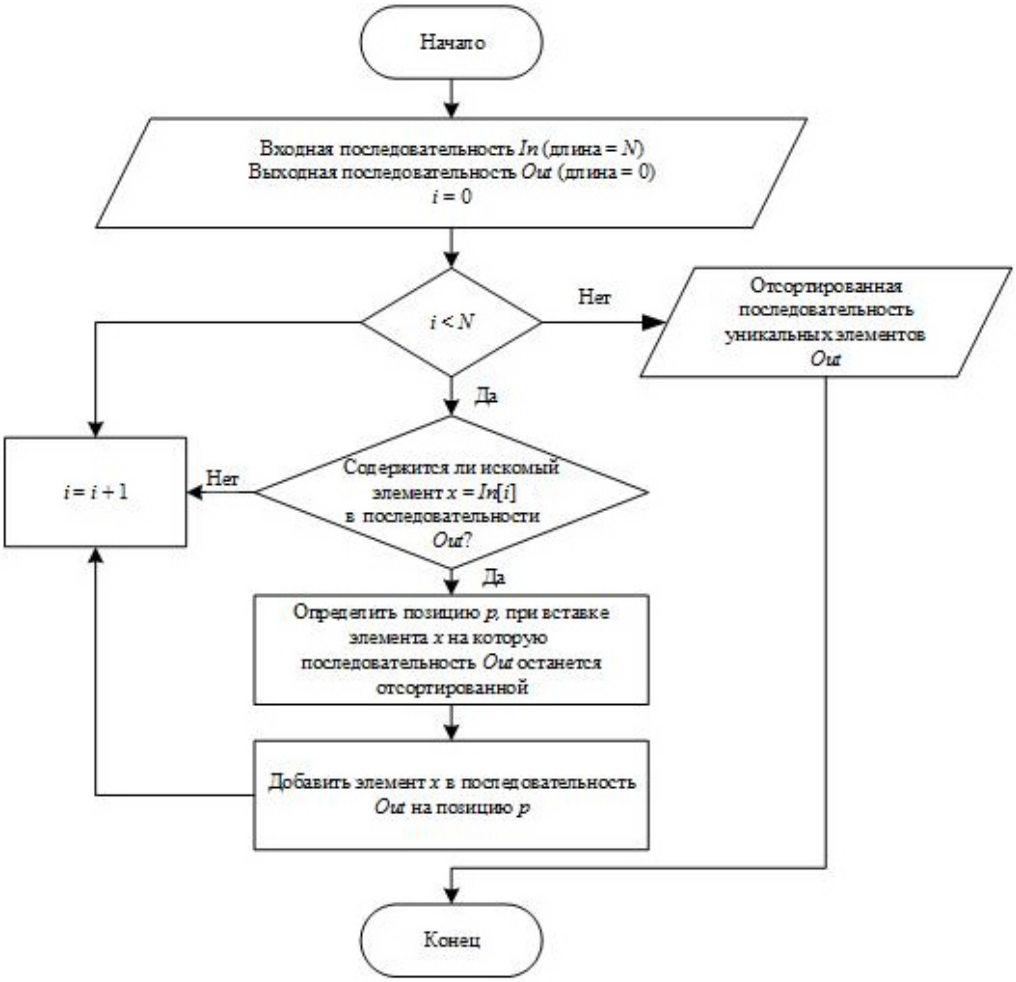
Алгоритм расширенного бинарного поиска. Алгоритм РБП предназначен для формирования отсортированного перечня уникальных элементов в режиме постоянного поступления новых значений.

Бинарный поиск, лежащий в основе РБП, работает только на отсортированной последовательности и проверяет наличие элемента в списке. Однако, в случае его отсутствия, он может дополнительно возвращать номер позиции, после вставки искомого элемента на которую список остается отсортированным. Данный факт и лег в основу алгоритма РБП.

Суть его заключается в следующем: пусть имеется случайная последовательность, возможно содержащая в себе повторяющиеся элементы *In*. Будем «перекладывать» элементы в новую последовательность *Out*, которая изначально не содержит ни одного элемента. Берем первый элемент исходной последовательности *In* и переносим его в искомую (новую) последовательность *Out*. Последовательность из одного элемента является отсортированной, в связи с этим по отношению к ней можно применить бинарный поиск. Берем следующий элемент *x* из исходной последовательности *In*. Проверяем его наличие в выходной последовательности *Out*. Если *x* присутствует, то вставлять его повторно не требуется. Если *x* отсутствует, то РБП позволит определить, на какую позицию *p* его необходимо поставить в выходной последовательности *Out* так, чтобы она продолжала оставаться отсортированной. Аналогично осуществляется проверка-вставка всех элементов исходной последовательности.

В итоге, без реализации отдельного этапа сортировки, получается отсортированная выходная последовательность, состоящая из уникальных элементов.

На рисунке представлена блок-схема данного алгоритма.



Алгоритм расширенного бинарного поиска

Особенности программной реализации алгоритма РБП. Скорость РБП во многом зависит от того, каким образом хранятся и обрабатываются объекты в памяти ЭВМ. Так, например, если данные хранятся в массиве, то существенную часть времени будет занимать операция вставки элементов. Каждый раз, при необходимости вставить новый элемент в массив, потребуется все элементы, стоящие правее позиции для вставки, перенести на одну позицию вправо.

От этого недостатка свободен, например, способ хранения данных в виде связного списка. В этом случае для вставки нового элемента достаточно лишь выделить для него память и изменить ссылки, указывающие на порядок элементов в списке [10].

Для проверки эффективности использования связных списков при реализации расширенного бинарного поиска был разработан программный продукт на языке C++ в среде Microsoft Visual Studio 2012, в рамках которого РБП был реализован двумя различными способами. Первый предусматривал обработку поступающих на вход значений в виде массива, другой – в виде связного списка. Для проведения экспериментов использовалась описанная выше ПЭВМ. Результаты показали, что функция, на вход которой поступил массив размером 50 000 элементов, потратила на формирование отсортированного перечня уникальных элементов 116 мс, а функция, на вход которой поступил связный список того же размера, – 74 мс. Таким образом, выигрыш во времени составил более 36 %.

Заключение

В основе антивирусного эвристического статического анализа лежит задача бинарной классификации файлов, которая в данном случае предусматривает возможность отделения вредоносных файлов от легитимных. Качество обучения классификатора во многом зависит от эффективного подбора обучающего множества. Для решения этой задачи предложена методика,

позволяющая целенаправленно отбирать файлы в обучающее множество, при условии обеспечения максимальной информативности при минимальном объеме выборки. Анализ особенностей программной реализации данной методики позволил предложить алгоритмы, наиболее эффективные с точки зрения времени расчета. Это дает возможность перейти к программной реализации предложенных подходов в рамках конкретной антивирусной лаборатории для их дальнейшего внедрения в системы антивирусной защиты.

СПИСОК ЛИТЕРАТУРЫ

1. *Потапов А. С.* Распознавание образов и машинное восприятие. СПб.: Политехника, 2007. 552 с.
2. *Флах П.* Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. М.: ДМК Пресс, 2015. 400 с.
3. *Ажмухамедов И. М., Запорожец К. В.* Усовершенствованный метод фильтрации нежелательного трафика // Вестн. Астрахан. гос. техн. ун-та. Сер.: Управление, вычислительная техника и информатика. 2014. № 1. С. 98–104.
4. *Ажмухамедов И. М., Князева О. М.* Унификация подходов к управлению уровнем информационной безопасности в организациях различного профиля // Вестн. Астрахан. гос. техн. ун-та. Сер.: Управление, вычислительная техника и информатика. 2015. № 1. С. 66–77.
5. *Harrington P.* Machine Learning in Action. Manning Publications Co, 2012. 382 p.
6. *Marsland S.* Machine Learning: An Algorithmic Perspective. Chapman and Hall/CRC, 2014. 457 p.
7. *Демина Р. Ю., Ажмухамедов И. М.* Зависимость эффективности обнаружения вредоносного программного обеспечения от качества обучающей выборки в алгоритмах классификации // Математические методы в технике и технологиях – ММТТ-28: сб. тр. XXVIII Междунар. науч. конф.: в 12 т. Т. 3. Саратов: СГТУ, 2015. С. 64–66.
8. *Демина Р. Ю., Ажмухамедов И. М.* Методика формирования обучающего множества при использовании статических антивирусных методов эвристического анализа // Инженерный вестн. Дона. 2015. № 3. URL: http://ivdon.ru/uploads/article/pdf/IVD_204_demina_azhmuhamedov.pdf_0b8ea4a2fc.pdf (дата обращения: 16.03.2015).
9. *Демина Р. Ю., Ажмухамедов И. М.* Использование бинарного поиска для формирования отсортированного перечня уникальных элементов // Математические методы в технике и технологиях – ММТТ-27: сб. тр. XXVII Междунар. науч. конф.: Тамбов: Изд-во ТГТУ, 2014. С. 124–126.
10. *Кнут Д. Э.* Искусство программирования. Т. 3. Сортировка и поиск. М.: Вильямс, 2012. 824 с.

Статья поступила в редакцию 16.03.2017

ИНФОРМАЦИЯ ОБ АВТОРЕ

Демина Раиса Юрьевна – Россия, 414056, Астрахань; Астраханский государственный технический университет; аспирант кафедры информационной безопасности; raisapereverzeva@gmail.com.

R. Yu. Demina

FEATURES OF SOTWARE IMPLEMENTATION OF ALGORITHMS OF THE TECHNIQUE OF THE TRAINING SET FOR THE BINARY QUALIFIERS USED IN THE ANTI-VIRUS HEURISTIC STATIC ANALYSIS

Abstract. In the context of rapid expansion of means of computation, computer files are increasingly used as objects of binary classification. Binary classification of files plays a specific role in the anti-virus heuristic static analysis. Classification process consists of two stages: training and recognition. At the training stage a training set of objects is formed. The selection is important to be carried out not randomly, but in a targeted manner, taking into account a variety of objects. As in-

troduction of the additional procedure of formation of the training set will lead to the increase in total training hours, it is necessary to consider all features of the software implementation, so that this stage would last the least time possible. The article presents the technique of formation of the training set and describes the main nuances needed to reduce the time of calculations. There has been determined the algorithm of advanced binary search designed to form the sorted sequence of unique elements. There is considered the main feature (a method of data storage) which can influence the time of algorithm execution. The article gives an example of a code realizing the function of advanced binary search in the high level language C++. The research results allow attempting program realization of the offered approaches for their further implementation into anti-virus systems.

Key words: binary classification, training set, binary search, software implementation.

REFERENCES

1. Potapov A. S. *Raspoznavanie obrazov i mashinnoe vospriiatie* [Image recognition and computer vision]. Saint-Petersburg, Politekhnika Publ., 2007. 552 p.

2. Flash P. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* Cambridge University Press, 2012. 396 p. (Russ. ed.: Flakh P. *Mashinnoe obuchenie. Nauka i iskusstvo postroeniia algoritmov, kotorye izvlekaiut znaniia iz dannykh*. Moscow, DMK Press, 2015. 400 p.).

3. Azhmukhamedov I. M., Zaporozhets K. V. Usovershenstvovannyi metod fil'tratsii nezhelatel'nogo trafika [Improved method of filtration of undesired traffic]. *Vestnik Astrakhanskogo gosudarstvennogo tekhnicheskogo universiteta. Seriia: Upravlenie, vychislitel'naia tekhnika i informatika*, 2014, no. 1, pp. 98-104.

4. Azhmukhamedov I. M., Kniazeva O. M. Unifikatsiia podkhodov k upravleniiu urovnem informatsionnoi bezopasnosti v organizatsiiakh razlichnogo profilia [Unification of the approaches to control of the level of information security in different organizations]. *Vestnik Astrakhanskogo gosudarstvennogo tekhnicheskogo universiteta. Seriia: Upravlenie, vychislitel'naia tekhnika i informatika*, 2015, no. 1, pp. 66–77.

5. Harrington P. *Machine Learning in Action*. Manning Publications Co, 2012. 382 p.

6. Marsland S. *Machine Learning: An Algorithmic Perspective*. Chapman and Hall/CRC, 2014. 457 p.

7. Demina R. Iu., Azhmukhamedov I. M. Zavisimost' effektivnosti obnaruzheniia vredonosnogo programmnogo obespecheniia ot kachestva obuchaiushchei vyborki v algoritmakh klassifikatsii [Dependence of detecting malware efficiency on the quality of training sampling in classification algorithms]. *Matematicheskie metody v tekhnike i tekhnologiiakh – MMTT-28: sbornik trudov XXVIII Mezhdunarodnoi nauchnoi konferentsii: v 12 t. T. 3* [Mathematical Methods in Engineering and Technology - MMTT-28: Proceedings of the XXVIII International Scientific Conference: 12 volumes. Vol. 3]. Saratov, SGTU, 2015. P. 64-66.

8. Demina R. Iu., Azhmukhamedov I. M. Metodika formirovaniia obuchaiushchego mnozhestva pri ispol'zovanii staticheskikh antivirusnykh metodov evristicheskogo analiza [The technique of a training set formation in using statistic anti-virus methods of heuristic analysis]. *Inzhenernyi vestnik Dona, 2015, no. 3.* Available at: http://ivdon.ru/uploads/article/pdf/IVD_204_demina_azhmuhamedov.pdf_0b8ea4a2fc.pdf (accessed: 16.03.2015).

9. Demina R. Iu., Azhmukhamedov I. M. Ispol'zovanie binarnogo poiska dlia formirovaniia otsortirovannogo perechnia unikal'nykh elementov [Binary search in formation of sorted list of the unique elements]. *Matematicheskie metody v tekhnike i tekhnologiiakh – MMTT-27* [Mathematical Methods in Engineering and Technology - MMTT-27: Proceedings of the XXVII International Scientific Conference]. Tambov, Izd-vo TGTU, 2014. P. 124-126.

10. Knuth D. E. *The Art of Computer Programming. Vol. 3. Sorting and Searching.* Addison-Wesley Professional, 1998. 780 p. (Russ. ed.: Knut D. E. *Iskusstvo programmirovaniia. Sortirovka i poisk.* Moscow, Vil'iams Publ., 2012. 824 p.).

The article submitted to the editors 16.03.2017

INFORMATION ABOUT THE AUTHOR

Demina Raisa Yurevna – Russia, 414056, Astrakhan; Astrakhan State Technical University; Postgraduate Student of the Department of Information Security; raisapereverzeva@gmail.com.