УДК 004.492.3

С.М. ЛИСЕНКО, М.П. ЮЖЕКА
Хмельницький національний університет

ДОСЛІДЖЕННЯ МЕТОДУ ОПОРНИХ ВЕКТОРІВ ЯК ЗАСОБУ ІДЕНТИФІКАЦІЇ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В роботі досліджено метод опорних векторів як засіб ідентифікації шкідливого програмного забезпечення. Запропоновано підхід до виявлення шкідливого програмного забезпечення шляхом відслідковування підозрілої поведінки з подальшою класифікацією шкідливих програмних засобів.

Ключові слова: метод опорних векторів, класифікація шкідливого програмного забезпечення, worm-віруси, Trojan-віруси, adware-віруси, лінійний класифікатор, нелінійний класифікатор.

S.M. LYSENKO, M.P. YUGEKA
Khmelnytskyi national university

SUPPORT VECTOR MACHINE APPROACH FOR MALWARE IDENTIFICATION

Abstract. In article the support vector machine for the malware identification is researched. The new approach for malware detection by tracking the suspicious behaviour and further its classification was proposed.

The experiments demonstrate that a small amount for input data all classification methods are able to classify objects. Therefore increase of the points of different classes of linear methods shows the incorrect results of the classification: some objects belonging to a class can be identified as objects of another one. In order to avoid such situations, it is appropriate to use nonlinear classifiers.

Also the non-linear classifier based on B-spline is able to establish the greatest distance between groups of different classes. It means that this classifier sets the longest distinction between classes and has the highest efficiency for the objects' classification.

Keywords: support vector machine, malware classification, worm-viruses, Trojan-viruses, adware-viruses, linear classifier, non-linear classifier.

Вступ

Сьогодні питання захисту інформації є досить актуальним, так як все більше фірм та організацій переходять на електронну форму ведення документації. Зберігання документів в такій формі має ряд переваг над паперовою. До них належить: простота пошуку інформації, зменшення затрат н її зберігання, простота зміни та оновлення інформації та ще ряд інших, які дозволяють суттєво прискорити роботу з даними. Але такий спосіб зберігання має досить суттєвий недолік: електроні документи можуть бути викрадені зловмисниками віддалено, при не залишити жодних слідів за собою. Єдиним ефективним способом вирішення такої проблеми є створення програмного забезпечення, яке б не допускало будь-які спроби отримання секретної інформації [1].

Шкідливе програмне забезпечення (ШПЗ) може спричинити різну шкоду. Результатом його діяльності може бути зниження продуктивності комп'ютерної системи (КС), збої в роботі окремих програмних систем, неадекватність поведінки окремих програм. Це робиться для того, щоб порушити систему захисту і отримати захищену інформацію. Якщо в першому випадку наслідки діяльності програми-шпигуна помітні користувачу, то інший спосіб отримання таємної інформації полягає в тому, щоб не дати себе виявити і таємно збирати та відправляти отримані дані, за допомогою яких можна отримати доступ до захищеної інформації [1, 2].

Для виявлення та знищення ШПЗ розроблено кілька методів. Найбільш поширеним є періодичне сканування оперативної пам'яті та файлів комп'ютера на наявність шкідливих програм. Виявлення відбувається завдяки порівнянню досліджуваного файлу з базою, де зберігаються записи про вже відомі загрози. Основним недоліком такого підходу є те, що база знань про відомі загрози потребує постійного оновлення та є досить велика імовірність того, що в КС присутнє ШПЗ, яке ще не було внесено до списку небезпечного ПЗ, а тому розглядається антивірусною програмою як безпечне [2].

Інший спосіб полягає у відслідковуванні програм, які під час своєї роботи проявляють підозрілу поведінку. Під підозрілою поведінкою мається на увазі періодична відправка даних по мережі через різні канали, дуже часта зміна свого початкового коду, або коду інших програм, зміна параметрів системи для автоматичного запуску без відома користувача, тобто поведінка, яка притаманна програмі, яка намагається запобігти своєму виявленню та знищенню [1, 2].

Постановка задачі

Проблему ідентифікації ШПЗ при обмеженому наборі альтернатив можна сформувати наступним чином: існує множина програм $T = \{t_1,\ldots,t_k\}$ і множина типів програм $A = \{a_1,\ldots,a_n\}$; для деякої підмножини програм $T \subseteq T'$ відомі типи $D = \{(t_i, a_i)\}_{i-1}^{e}$. Необхідно встановити, що з множини A являється справжньою шкідливою програмою серед решти програм (невідомих чи підозрілих за своєю поведінкою) $T'' = \{t_{|T'|+1},\ldots,t_k\} \subseteq T$ [3, 5, 12].

Задачу ідентифікації ШПЗ можна розглядати як задачу класифікації з кількома класами. В цьому випадку множину A складає множина наперед визначених класів та їх міток, D – навчальні приклади, а множина T'' – об'єкти, що класифікуються. Ціллю являється побудова класифікатора, що вирішує задачу знаходження деякої цільової функції $F : T \times A \to [0,1]$, що відносить випадкову програму множини T до відповідного класу. Значення функції інтерпретується як ступінь приналежності об'єкта класу: 1 – відповідає позитивному рішенню, 0 – негативному.

Дану задачу можна звести до вирішення кількох бінарних. Для цього існує кілька стратегій вибору вирішення:

1) *«один проти всіх».* Для вирішення задачі створюється n класифікаторів таким чином, що кожен клас a_j

співвідноситься з рештою (n-1) класів, тобто в кожному з j випадків вибір здійснюється з двох варіантів: «клас a_j» і «не клас a_j». Остаточне рішення щодо приналежності до певного класу приймається за схемою «переможець забирає все» – переможцем вважається клас, який має максимальне значення цільової функції F [4, 5].

2) *«кожен проти кожного»*. Класифікатори будуються для кожної пари класів для того, щоб можна було однозначно розділити будь-які два класи з множини A. Кількість класифікаторів в цьому випадку дорівнює $n\frac{n-1}{2}$. Після подачі на входи кожного з навчених класифікаторів тестового зразка отримуються відповіді, які містять інформацію про його приналежність до одного з двох класів, які брали участь у навчанні. До отриманої множини відповідей застосовується схема мажоритарного голосування, і клас, вибраний більшістю класифікаторів, приймається як підсумкове рішення.

3) *орієнтований ациклічний граф (DNA)*. На етапі навчання стратегія працює аналогічно стратегії «кожен проти кожного». На етапі тестування і безпосередньої класифікації використовується кореневий бінарний орієнтований ациклічний граф (орієнтоване дерево) з $n\frac{n-1}{2}$ внутрішніми вузлами – навченими бінарними класифікаторами , і n листками. Об'єкт, що класифікується, проходить шлях від кореня до одного з листків, при цьому залежно від результатів класифікації в кожному вузлі один з класів відкидається. Подальші дії проходять по гілці, що відповідає другому класу. Після виконання n - 1 подібних операцій , алгоритм досягає листа , який приймається як підсумкове рішення класифікатора [4, 6, 13].

Таким чином, для того, щоб класифікація на декілька класів проходила успішно, необхідно, в першу чергу, добитися високої точності при вирішенні задач бінарної класифікації. Важливими етапами при цьому є вибір алгоритму класифікації і його параметрів, кількості навчальних прикладів, а також вибір характеристик програми для аналізу і необхідного обсягу вибірок.

Основний розділ

Лінійні та нелінійні SVM

Розглянемо задачу класифікації засобами лінійної класифікації методом опорних векторів [5].

Нехай кожен елемент, що потрібно класифікувати, є точкою в n-мірному просторі R^n. Будемо вважати, що точка $x_i, i = 1...m$ в цьому просторі має мітку $y_i = \pm 1$. Задача полягає в тому, щоб розділити дані (n-1)-мірною гіперплощиною, а також знайти дану гіперплощину. Також важливим є те, щоб два розділювачі класи лежали якомога далі від гіперплощини (рис. 1).

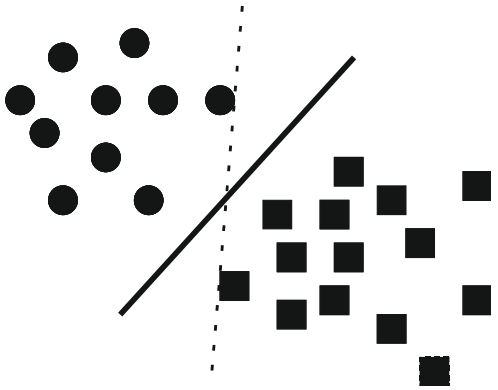


Рис. 1. Лінійна класифікація об'єктів двох класів

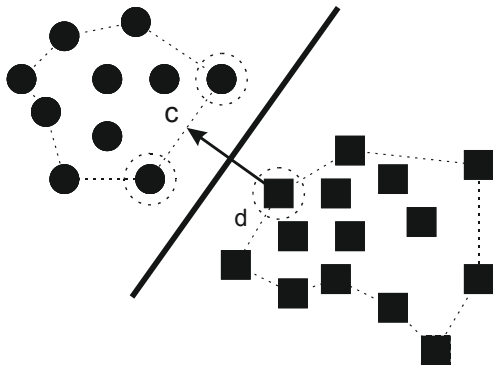


Рис. 2. Перший підхід класифікації

Існує два підходи для такої класифікації. Перший полягає в тому, щоб знайти дві найближчі точки в опуклих оболонках даних, а потім провести розділяючу гіперплощину через середину відрізка (рис. 2). Формально це перетворюється в задачу квадратичної оптимізації:

$$\min_{\alpha}\left\{\|c-d\|^2, \partial e\, c=\sum_{y_i=1}\alpha_i x_i, d=\sum_{y_i=-1}\alpha_i x_i\right\}, \qquad \sum_{y_i=1}\alpha_i=\sum_{y_i=-1}\alpha_i=1, \alpha_i\geq 0. \tag{1}$$

Другий підхід полягає в максимізації зазору між двома паралельними опорними гіперплощинами, потім провести паралельну їм гіперплощину, яка рівновіддалена від опорних [11].

Під опорною гіперплощиною для множини точок X мається на увазі гіперплощина, де всі точки з множини X лежать по одній стороні від неї (рис. 3). Відстань від точки до гіперплощини можна обчислити наступним чином:

$$y(x) = w \perp x + w_0 = 0, \tag{2}$$

що є рівним $\frac{|y(x)|}{\|w\|}$.

Усі точки класифіковані правильно, якщо $t_n y(x_n) > 0, t_n \in \{-1,1\}$ [5].

Таким чином, отримано задачу квадратичної оптимізації:

$$\min_{w,b}\left\{\frac{1}{2}\|w\|^2\right\} \text{ за умови } t_n(w \perp x_n + w_0) \geq 1. \tag{3}$$

Даний підхід дозволяє отримувати стійкі рішення, що надає змогу краще передбачати подальшу класифікацію.

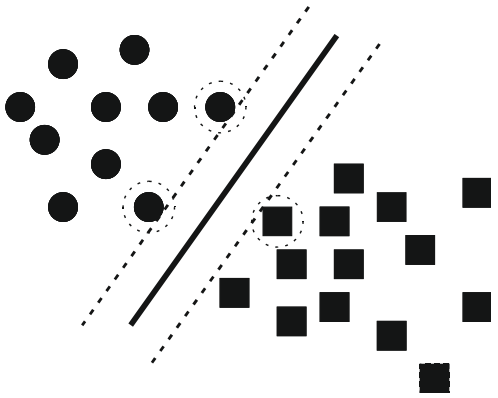


Рис. 3. Максимізація зазору між двома опорними площинами

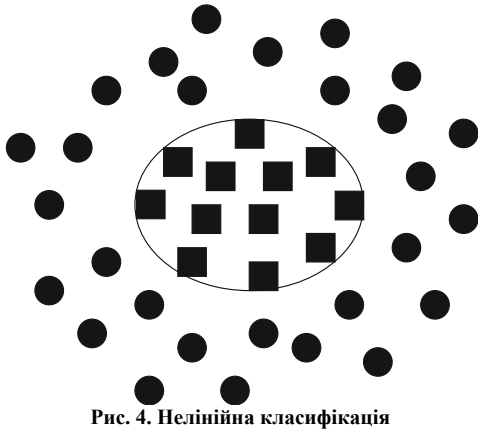


Рис. 4. Нелінійна класифікація

Задача нелінійної класифікації

Досить часто буває необхідно розділити дані нелінійними функціями [4]. Нерідко трапляються випадки, коли об'єкти за своїми властивостями розміщуються в просторі, як показано на рисунку 4. В таких випадках їх розділення лінійною функцією є неможливим.

Класичним підходом є розгортання нелінійної класифікації в простір більшої розмірності, в якому запускається на виконання лінійний класифікатор. Для цього необхідно для кожного об'єкта потрібного ступеня ввести додаткову змінну [6, 9, 14].

Наприклад, для того, щоб в двовимірному просторі $[r,s]$ вирішити задачу класифікації квадратичної функції, потрібно перейти в п'ятимірний простір $[r,s]\to[r,s,rs,r^2,s^2]$. Цей перехід можна формалізувати: визначаємо $\theta: R^2 \to R^5 : \theta(r,s) = (r,s,rs,r^2 s^2)$. Таким чином, вектор, що знаходиться в площині R^5, тепер відповідає квадратичній кривій загального розташування в R^2, а функція класифікації буде виглядати наступним чином:

$$f(x) = sign(\theta(w)\cdot\theta(x) - b). \tag{4}$$

Отже, вирішивши задачу лінійного розділення в новому просторі отримується розв'язок задачі квадратичного розділення в початковому просторі [15].

Ядра

Основною ідеєю ядрового перетворення є відображення даних в простір, в якому поверхня, що розділює ці дані, перетвориться з нелінійної в лінійну. Ядрові перетворення можна швидко обчислювати для скалярних добутків, навіть при нескінченній розмірності цільового простору:

$$K(x,x') = (x\cdot x')^d = \exp(-\gamma\|x\cdot x'\|^2), \gamma > 0. \tag{5}$$

Функція $K: X\times X \to R$ називається ядром (kernel function), якщо її можна представити в вигляді $K(x,x') = \langle\psi(x),\psi(x')\rangle$ при деякому відображенні $\psi: X \to H$, де H – простір зі скалярним добутком.

Теорема Мерсера регламентує функції, що можуть виступати ядрами [6,10].

Теорема. Функція K(x, x′) являється ядром тоді і тільки тоді, коли вона є симетричною, *K(x, x′) = K(x′, x)*, і невід'ємно визначена: $\int_X\int_X K(x,x')g(x)g(x')dxdx' \geq 0$ для будь-якої функції $g: X \to R$ [6, 7].

Для побудови ядра для вирішення практичної задачі потрібно враховувати наступні правила [6]:

1. Довільний скалярний добуток $K(x,x') = \langle x,x'\rangle$ є ядром.
2. Константа $K(x,x') = 1$ є ядром.
3. Добуток ядер $K(x,x') = K_1(x,x')\cdot K_2(x,x')$ є ядром.
4. Для будь-якої функції $\psi: X \to R$ добуток $K(x,x') = \psi(x)\cdot\psi(x')$ є ядром.
5. Лінійна комбінація ядер з невід'ємними коефіцієнтами $K(x,x') = \alpha_1 K_1(x,x') + \alpha_2 K_2(x,x')$ є ядром.
6. Композиція довільної функції $\varphi: X \to X$ і довільного ядра K_0 є ядром: $K(x,x') = K_0(\varphi(x),\varphi(x'))$.
7. Якщо $s: X\times X \to R$ довільна симетрична інтегрована функція, то $K(x,x') = \int_X s(x,z)s(x'z)dz$ є ядром.
8. Функція виду $K(x,x') = k(x-x')$ є ядром тоді і тільки тоді, коли Фур'є-образ $F[k](\omega) = (2\pi)^{\frac{n}{2}}\int_X e^{-i\langle\omega,x\rangle}k(x)dx$ невід'ємний.
9. Межа локально-рівномірно збіжної послідовності ядер є ядром.

10. Композиція довільного ядра K_0 і довільної функції $f: R \to R$ представимо у вигляді степеневого ряду, що сходиться з невід'ємними коефіцієнтами $K(x, x') = f(K_0(x, x'))$ є ядром. Зокрема, функції $f(z) = e^z$ і $f(z) = \frac{1}{1-z}$ від ядра є ядрами.

Навчання

Для навчання системи SVM потрібно привести вхідні дані до відповідного формату. Нехай ШПЗ буде класифікуватись наступним чином:

- Worm;
- Trojan;
- AdWare.

Для того, щоб система була в змозі класифікувати досліджувану програму необхідно сформувати навчальну вибірку. Ця вибірка буде складатись із деякої скінченної послідовності,

$$V_i = \{[a_1, a_2, a_3 ... a_n; s]; [a_1, a_2, a_3 ... a_n; s]; ...\}, \quad (6)$$

де a_n – поведінка, притаманна ШПЗ, s – клас приналежності даного ШПЗ. Приклад типових дій для класів ШПЗ надано в таблиці 1.

Маючи ці дані система буде в змозі побудувати на координатній площині множину точок M:

$$D = \{M_1(m, p); M_2(m, p); ...; M_n(m, p)\}, \quad (7)$$

де n – кількість досліджуваних програм. Змінні m і p відповідають за відсоток приналежності досліджуваної програми до числа ШПЗ, та відсоток приналежності до певного класу ШПЗ відповідно. При чому p div 10 вказуватиме до якого класу ШПЗ належить програма (1 – worm, 2 – trojan і т. д.), а p mod 10 буде вказувати на скільки відсотків вона йому відповідає. Навчання системи потрібно проводити доти, доки вона на вхідні дані не буде давати однозначної відповіді [7, 8].

Таблиця 1

Типова поведінка для певних типів ШПЗ

Клас ШПЗ	Типові дії ШПЗ відповідного класу
Троянські програми	розповсюджуються за ініціативи користувачів; порушення роботи інших програм (зависання комп'ютера, що вирішується лише перезавантаженням, і неможливості їх запуску);незалежне від власника встановлення в якості стартової сторінки спам-посилань, реклами; перетворення мови текстових документів у бінарний код; комунікація через мережу нестандартними портами; відкриття мережевого доступу до ресурсів комп'ютера; ведення та відправка логів комп'ютерної системи; блокування антивірусного ПЗ; запис даних у реєстр; блокування доступу до мережі окремим програмам
Worm-віруси	редагування виконуваних файлів; копіювання в системні каталоги; доступ до системних файлів; сканування комп'ютерної системи на наявність відкритих портів; відкриття доступу до мережних ресурсів; редагування реєстру, автозапуск; надмірне використання ресурсів комп'ютерної системи; перехоплення переривань; запуск під час відкриття документів; редагування завантажувального сектора; створення чи редагування файлів на змінних носіях; редагування таблиці розділів локальних дисків
Рекламне ШПЗ (adware)	завантаження зображень із зовнішніх ресурсів; редагування реєстру, автозапуск; періодичне опитування певного сервера; відстежування дій користувача; перехоплення, підміна сторінок браузера; встановлення додатків у браузер; перенаправлення запитів; завантаження файлів з мережі без відома користувача; відправлення повідомлень на сервер; відслідковування запущених процесів

Отже, для того, щоб навчити систему класифікувати ШПЗ потрібно формувати навчальну вибірку досить великих розмірів. Тому чим ширше коло об'єктів буде охоплювати вибірка, тим якісніше система буде в змозі класифікувати ШПЗ.

Експерименти та дослідження

Експерименти проводилися в середовищі Matlab засобами SVM Toolbox. Для дослідження ефективності запропонованого підходу ідентифікації ШПЗ було згенеровано множину типових поведінок означених вище класів ШПЗ. На вхід системи, що ґрунтується на методі опорних векторів, було подано дану множину. Після виконання процедури навчання, було сформовано базу даних для ідентифікації.

Наступним кроком для оцінки адекватності розробленого підходу було згенеровано множину нових поведінок, що характеризуються властивостями ШПЗ, та відрізняються від наявних в базі даних поведінок.

Для класифікації здійснювалося наступними способами:

1) лінійним;
2) нелінійним (Гауса);
3) нелінійним (поліноміальний);
4) лінійним (bsline);
5) нелінійним (експоненційний).

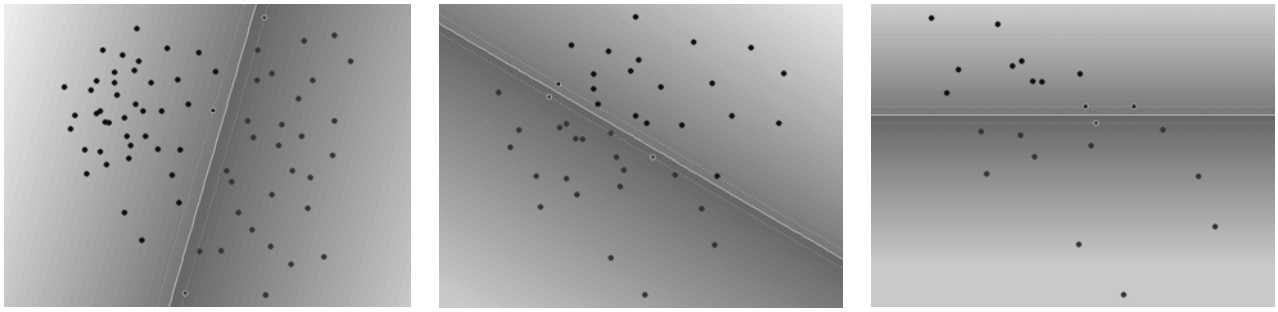
Примітка. В даному дослідженні класифікація здійснювалася для двох класів. Першим етапом була класифікація шкідливого та нешкідливого ПЗ; другий етап полягав в розділенні ШПЗ на класи.

Моделювання класифікації ШПЗ лінійним способом

Розглянемо лінійний спосіб класифікації. На площині розміщуються точки, що відповідають вимогам (6) та (7). Умовою для того, щоб класифікатор даним способом зміг розділити різні класи є те, що групи точок певного класу не повинні перетинатися. Даний підхід є найбільш простим і дозволяє класифікувати об'єкти за значно меншу кількість ітерацій, ніж описані нижче.

Розміщення точок на площині та результати класифікації подано на рисунках 5–8.

Розміщення точок на площині та результати класифікації подано на рисунках 5а–5в. Продуктивність класифікатора представлено в таблиці 2.

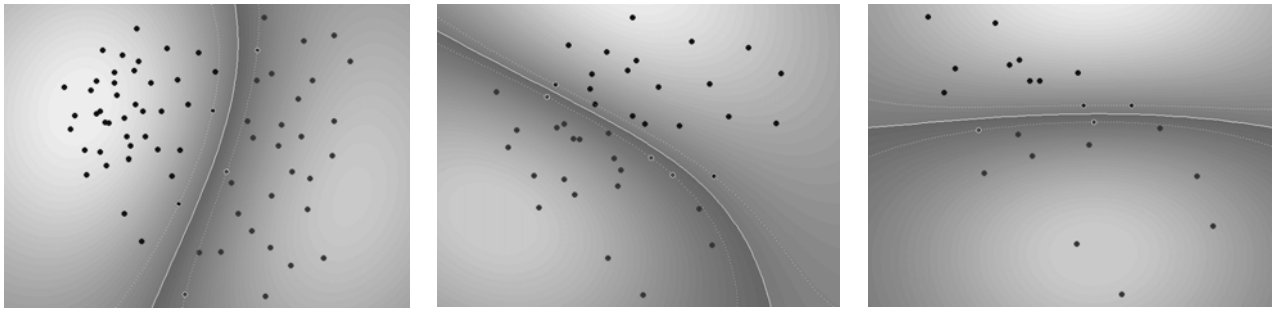


а) б) в)

Рис. 5. Розміщення груп точок на площині і їх лінійне розділення:
а) шкідливе / не шкідливе; б) троянська програма / не троянська програма; в) worm / adware

Моделювання класифікації ШПЗ нелінійним способом (Гауса)

У випадку неможливості лінійного розмежування груп різних класів доцільно використовувати нелінійні класифікатори. До їх числа належить також нелінійний класифікатор на основі функції Гауса. Даний класифікатор дозволяє вирішити задачу, коли групи об'єктів, зображених на рисунку 6, неможливо розмежувати за допомогою лінійної функції. Одним з недоліків такого класифікатора є його невисока точність, що й видно на рисунках 6а–6в.

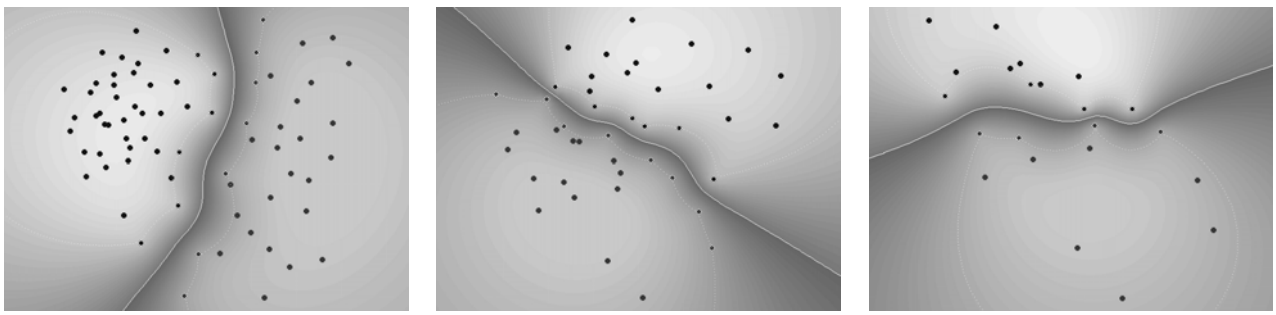


а) б) в)

Рис. 6. Розміщення груп точок на площині та їх розділення гаусівським класифікатором
а) шкідливе / не шкідливе; б) троянська програма / не троянська програма; в) worm / adware

Моделювання класифікації ШПЗ нелінійним способом (експоненційний)

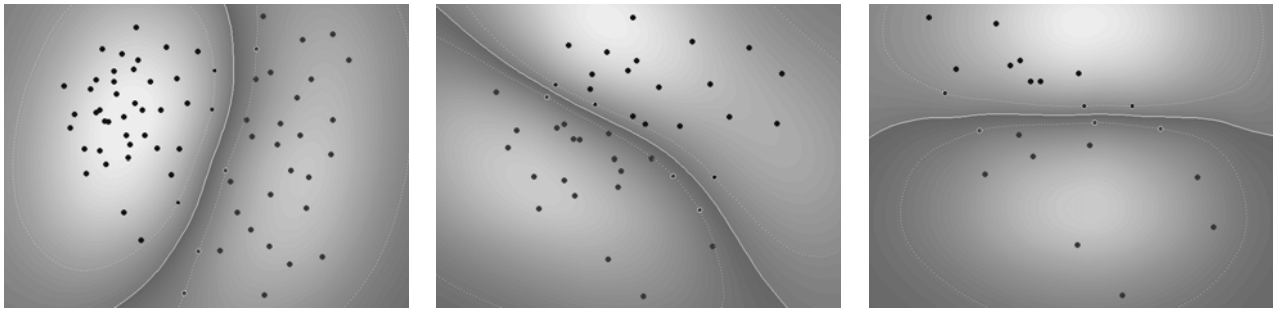
Експоненційний класифікатор є одним з підвидів нелінійних класифікаторів, який дає досить велику точність, в порівнянні з гаусівським. Для наочного відображення різниці між даним і гаусівським класифікаторами розділимо групу об'єктів, що класифікувалась вищеописаним способом, за допомогою експоненційного класифікатора. Якщо порівняти рисунки 6а, 6б, 6в та 7а, 7б, 7в, то можна помітити, що даний класифікатор працює точніше. Це означає, що вектор, що розділяє класи, знаходиться на максимально можливій відстані від кожної точки, через які проходять опорні вектори. Також даний класифікатор характеризується більшою ефективністю. На рисунку 6 видно, що для класифікації груп знадобилось побудувати 116 опорних векторів, в той час як для класифікації тієї ж групи об'єктів експоненційному класифікатору знадобилось 44 опорних вектора.



а) б) в)

Рис. 7. Розміщення груп точок на площині і їх розділення експоненційним класифікатором:
а) шкідливе / не шкідливе; б) троянська програма / не троянська програма; в) worm / adware

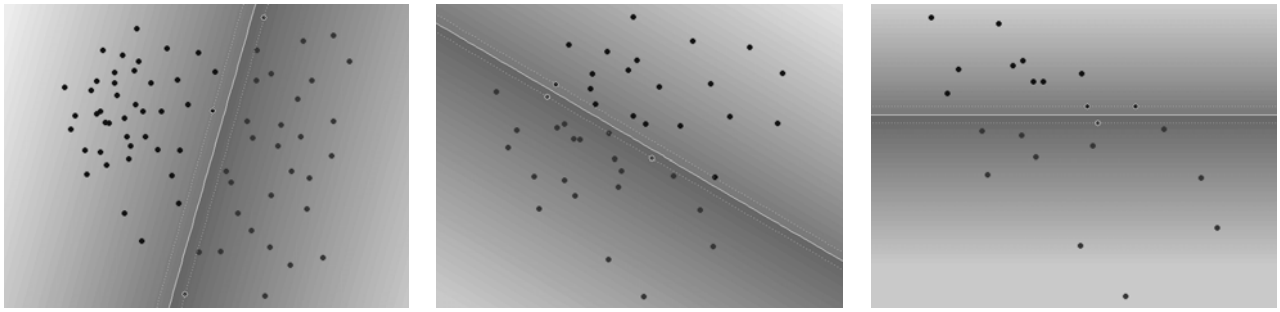
Ще одним представником нелінійних класифікаторів є класифікатор, в основі якого взято функцію B-spline. Даний класифікатор, результати класифікації якого показано на рисунках 8а-8в, є досить точним в порівнянні з використаними, і надає змогу чітко встановити межу між класами. Результати його роботи подано в таблиці 2.



а) б) в)

Рис. 8. Розміщення груп точок на площині і їх розділення B-сплайновим класифікатором: а) шкідливе / не шкідливе; б) троянська програма / не троянська програма; в) worm / adware

Поліноміальний класифікатор відноситься до лінійних класифікаторів і не є на стільки точним, як нелінійні класифікатори. Проте він дозволяє більш точно класифікувати різні групи об'єктів, ніж звичайний лінійний класифікатор. Результати його класифікації показано на рисунках 9а-9в.



а) б) в)

Рис. 9. Розміщення груп точок на площині і їх розділення поліноміальним класифікатором: а) шкідливе / не шкідливе; б) троянська програма / не троянська програма; в) worm / adware

Зведені результати експериментальних досліджень представлено таблицею 2.

Таблиця 2

Результати експериментальних досліджень класифікаторів методом опорних векторів

Параметри / **Класифікатор**	**Ефективність класифікації типу «ШПЗ-не ШПЗ», %**	**Ефективність класифікації Worm-вірусів, %**	**Ефективність класифікації Trojan-вірусів, %**	**Ефективність класифікації вірусів adware, %**	**Статус результату виконання**	**Час виконання, сек**	**Відстань між класифікованими групами**	**Кількість згенерованих векторів**
Лінійний	91	80	87	91	OPTIMAL_SOLUTION	0,5	0.167864	3
Нелінійний (Гауса)	95	96	93	93	OPTIMAL_SOLUTION	0,4	0.209479	5
лінійний (поліноміальний)	98	96	99	93	OPTIMAL_SOLUTION	0,4	0,167775	3
нелінійний (b-spline)	97	98	97	99	OPTIMAL_SOLUTION	0,4	1,293526	7
нелінійний (експоненційний)	98	95	96	98	OPTIMAL_SOLUTION	0,5	0,307337	13

Для експериментальної вибірки найбільш ефективним способом ідентифікації ШПЗ засобами методу опорних векторів є нелінійний B-spline, оскільки він дозволяє отримати найбільшу відстань між опорними векторами за невеликий час, при цьому результати дослідження демонструють найкращі результати виявлення класу ШПЗ – trojans.

Висновки

В роботі досліджено метод опорних векторів як засіб ідентифікації шкідливого програмного забезпечення. Запропоновано підхід до виявлення шкідливого програмного забезпечення шляхом відслідковування підозрілої поведінки з подальшою класифікацією шкідливих програмних засобів.

В результаті дослідження експериментально з'ясовано, що на невеликій кількості вхідних даних усі методи здатні класифікувати об'єкти, проте із збільшенням кількості точок різних класів лінійні методи можуть допускати некоректності у класифікації. Це буде проявлятись в тому, що деякі об'єкти, що належать до певного класу можуть ідентифікуватись як об'єкти іншого класу. Тому для того, щоб уникати таких ситуацій, доцільним є використання нелінійних класифікаторів. В ході експериментальних досліджень було також з'ясовано, що нелінійний класифікатор на основі B-spline здатний встановити найбільшу відстань між групами різних класів. Це значить, що даний класифікатор задає найбільш чітку межу між класами і має найбільшу ефективність серед досліджуваних.

Запропонований підхід в подальшому можу бути основою методу антивірусного діагностування на основі методу опорних векторів.

Література

1. Платонов В.В. Программно-аппаратные средства обеспечения информационной безопасности вычислительных сетей / В.В. Платонов –М. : Академия, 2006. – 121 с.
2. Rootkits, SpyWare/AdWare, Keyloggers & BackDoors. Обнаружение и защита ISBN 5-94157-868-7 / –М. : БХВ-Петербург, 2006. – 304 с.
3. Hsu C.-W., Lin C.-J. A comparison of methods for multi-class support vector machines / IEEE Transactions on Neural Networks, 2003. –425 с.
4. Барсегян А. Технологии анализа данных: Data Mining, Text Mining, Visual Mining, OLAP. 2 изд. / Барсегян А. –М. : БХВ-Петербург, 2008. – 384 с.
5. Support vector machine [Електронний ресурс] - Режим доступу : http://en.wikipedia.org/ wiki/Support_vector_machine
6. Bernhard Schlkopf, Alexander J. Smola Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond ISBN-0262194759 / Bernhard Schlkopf –M. : The MIT Press, 2001. – 644 с.
7. Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995.
8. Suykens, Johan A. K.; Vandewalle, Joos P. L.; Least squares support vector machine classifiers, Neural Processing Letters, vol. 9, no. 3, Jun. 1999, pp. 293–300.
9. Lee, Y.; Lin, Y.; and Wahba, G. (2004). "Multicategory Support Vector Machines, Theory, and Application to the Classification of Microarray Data and Satellite Radiance Data". Journal of the American Statistical Association 99 (465): 67–81.
10. Boser, Bernhard E.; Guyon, Isabelle M.; and Vapnik, Vladimir N.; A training algorithm for optimal margin classifiers. In Haussler, David (editor); 5th Annual ACM Workshop on COLT, pages 144–152, Pittsburgh, PA, 1992. ACM Press
11. Meyer, David; Leisch, Friedrich; and Hornik, Kurt; The support vector machine under test, Neurocomputing 55(1–2): 169–186, 2003
12. Platt, John; Cristianini, N.; and Shawe-Taylor, J. (2000). "Large margin DAGs for multiclass classification". In Solla, Sara A.; Leen, Todd K.; and Müller, Klaus-Robert; eds. Advances in Neural Information Processing Systems. MIT Press. pp. 547–553.
13. Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, B. P. (2007). "Section 16.5. Support Vector Machines". Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.
14. Aizerman, Mark A.; Braverman, Emmanuel M.; and Rozonoer, Lev I. (1964). "Theoretical foundations of the potential function method in pattern recognition learning". Automation and Remote Control 25: 821–837.
15. Hsu, Chih-Wei; Chang, Chih-Chung; and Lin, Chih-Jen (2003). A Practical Guide to Support Vector Classification. Department of Computer Science and Information Engineering, National Taiwan University.

References

1. V.V. Platonov Software and hardware-based security computer networks / Academy, 2006. – 121
2. Rootkits, SpyWare/AdWare, Keyloggers & BackDoors. Detection and Protection ISBN 5-94157-868-7 / Petersburg, 2006. – 304
3. Hsu C.-W., Lin C.-J. A comparison of methods for multi-class support vector machines / IEEE Transactions on Neural Networks, 2003. –425.
4. Barsegyan A. Data mining technology: Data Mining, Text Mining, Visual Mining, OLAP. 2. / Petersburg, 2008. – 384 .
5. Support vector machine / http://en.wikipedia.org/ wiki/Support_vector_machine
6. Bernhard Schlkopf, Alexander J. Smola Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond ISBN-0262194759 / Bernhard Schlkopf –M. : The MIT Press, 2001. – 644 .
7. Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995.
8. Suykens, Johan A. K.; Vandewalle, Joos P. L.; Least squares support vector machine classifiers, Neural Processing Letters, vol. 9, no. 3, Jun. 1999, pp. 293–300.
9. Lee, Y.; Lin, Y.; and Wahba, G. (2004). "Multicategory Support Vector Machines, Theory, and Application to the Classification of Microarray Data and Satellite Radiance Data". Journal of the American Statistical Association 99 (465): 67–81.
10. Boser, Bernhard E.; Guyon, Isabelle M.; and Vapnik, Vladimir N.; A training algorithm for optimal margin classifiers. In Haussler, David (editor); 5th Annual ACM Workshop on COLT, pages 144–152, Pittsburgh, PA, 1992. ACM Press
11. Meyer, David; Leisch, Friedrich; and Hornik, Kurt; The support vector machine under test, Neurocomputing 55(1–2): 169–186, 2003

12. Platt, John; Cristianini, N.; and Shawe-Taylor, J. (2000). "Large margin DAGs for multiclass classification". In Solla, Sara A.; Leen, Todd K.; and Müller, Klaus-Robert; eds. Advances in Neural Information Processing Systems. MIT Press. pp. 547–553.
13. Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, B. P. (2007). "Section 16.5. Support Vector Machines". Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.
14. Aizerman, Mark A.; Braverman, Emmanuel M.; and Rozonoer, Lev I. (1964). "Theoretical foundations of the potential function method in pattern recognition learning". Automation and Remote Control 25: 821–837.
15. Hsu, Chih-Wei; Chang, Chih-Chung; and Lin, Chih-Jen (2003). A Practical Guide to Support Vector Classification. Department of Computer Science and Information Engineering, National Taiwan University.

Рецензія/Peer review : 18.10.2013 р. Надрукована/Printed :24.11.2013 р.
Рецензент: Шалапко Ю.І., д.т.н., проф.

УДК 004.891.3: 004.3

Т.О. ГОВОРУЩЕНКО, А.В. КРАСІЙ
Хмельницький національний університет

ВИЗНАЧЕННЯ ХАРАКТЕРИСТИК ТА ВИБІР МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АНАЛІЗУ СПЕЦИФІКАЦІЙ

Дана робота доводить важливість та можливість прогнозування характеристик ПЗ на ранніх етапах життєвого циклу, а також показує можливість вибору прийнятної моделі життєвого циклу на основі аналізу специфікацій.

У роботі описано всі основні характеристики ПЗ, досліджено формат специфікації вимог до програмного забезпечення та визначено, яку інформацію для визначення (прогнозування) характеристик можна отримати зі специфікації вимог до ПЗ.

Автори провели аналіз існуючих автоматизованих засобів оцінювання характеристик ПЗ та визначили їх неспроможність надавати прогнозовані кількісні значення характеристик ПЗ на основі аналізу специфікацій.

У статті наведено приклад вибору прийнятної моделі життєвого циклу програмного проекту на основі аналізу специфікацій, який доводить можливість прийняття таких рішень на ранніх етапах життєвого циклу на основі ретельного аналізу специфікацій.

Ключові слова: програмне забезпечення (ПЗ), програмний проект, характеристики ПЗ, специфікація вимог до ПЗ, модель життєвого циклу ПЗ.

T. O. HOVORUSHCHENKO, A. V. KRASIY
Khmelnytskiy National University

THE DEFINING OF CHARACTERISTICS AND THE SELECTING OF SOFTWARE LIFE CYCLE MODEL BASED ON THE SPECIFICATIONS ANALYSIS

Abstract - This work proves the importance and possibility of software characteristics prognosis at the early life cycle stages, and proves the possibility of the selecting an appropriate software life cycle model on the basis of the specifications analysis.

In this paper, all main software characteristics are described, format of software requirements specification is analysed and information to defining (prognosis) characteristics ,that can be obtained from the software requirements specification, is defined.

The authors conducted the analysis of known automated tools for software characteristics evaluation and identified their inability to provide the prognosis quantitative values of software characteristics on the basis of software specifications analysis.

The example of selecting acceptable software project life cycle model based on an specifications analysis is in the article. This example proves the possibility of such decision-making at the early life cycle stages on the basis of thorough analysis of software requirements specifications.

Keywords: software, software project, software characteristics, software requirements specification (SRS), software life cycle model.

Вступ

Розроблення програмного забезпечення (ПЗ) – це діяльність, яка вимагає детального вивчення предметної області та повного розуміння цілей розроблюваного продукту [1].

Специфікація вимог до програмного забезпечення (Software Requirements Specification – SRS) – це основа для побудови ПЗ. Вона включає множину функціональних і нефункціональних (додаткових) вимог. Функціональні вимоги описують всі взаємодії користувача з програмним забезпеченням, нефункціональні – накладають обмеження на проект чи реалізацію [2].

Програмний проект – це комплекс взаємозв'язаних заходів, спрямованих на досягнення поставлених задач з чітко визначеними цілями протягом заданого періоду часу та при встановленому бюджеті [3].

Процес розроблення ПЗ тісно пов'язаний з процесом аналізу та оцінювання значущих характеристик ПЗ. До характеристик ПЗ належать: вартість ПЗ, тривалість життєвого циклу ПЗ, модель життєвого циклу ПЗ, ефективність ПЗ, простота або складність ПЗ (найважливіша характеристика ПЗ з точки зору розробника), зручність використання ПЗ, кросплатформеність ПЗ, захист ПЗ, повнота реалізації вимог, обсяг файлів ПЗ, вимоги до системного програмного забезпечення та технічних засобів, обсяг потрібної оперативної та дискової пам'яті, а також безумовно надійність та якість ПЗ (найважливіші характеристики з точки зору користувача).

Наразі все помітнішою стає криза у галузі розроблення ПЗ – великі проекти виконуються з відставанням від графіка або з перевищенням кошторису витрат, розроблений продукт не має необхідних функціональних можливостей, продуктивність його низька, якість програмного забезпечення не влаштовує споживачів. При наявності ряду методів та засобів, залученні кращих фахівців для розроблення технологій та стандартів