УДК 519.9

Шликов М. П., аспірант

**Про алгоритм побудови довірчого еліпсоїда Петуніна**

Київський національний університет імені Тараса Шевченка, факультет кібернетики, 03680, Україна, м. Київ, пр. Академіка Глушкова, 4д.
e-mail: maksym_shlykov@ukr.net

M. P. Shlykov, post-graduate student

**On the algorithm of Petunin confidence ellipsoid construction**

Taras Shevchenko National University of Kyiv, cybernetics faculty 03680, Kyiv, Glushkova st., 4d,
e-mail: maksym_shlykov@ukr.net

*Довірчі еліпсоїди та сфери мають широке застосування у сучасному машинному навчанні. Їх використовують для багатовимірного ранжування, знаходження медіани, знаходження викидів та оцінки квантилів певних рівнів. На сьогодні існує чимало алгоритмів, що використовують еліпсоїди та сфери. Зокрема така конструкція як еліпсоїд мінімального об'єму, що містить дану множину точок використовується для побудови класифікатора, що обмежує область вхідних даних (gap-tolerant classifier). Ще одним прикладом є застосування сфер для виявлення викидів, запропоноване Таксом та Дуйном. Більшість існуючих алгоритмів мають статистичну природу або ж їх створення було пов'язане з широковідомим методом опорних векторів. Їх основною метою є побудова структури, яка б якомога щільніше містила або відділяла тренувальні дані. Існує довірчий еліпсоїд, запропонований Ю. І. Петуніним. Його головною перевагою є відомий рівень значущості. У роботі запропоновано новий алгоритм побудови еліпсоїда Петуніна. Оригінальний алгоритм, запропонований автором, важко застосовувати у просторах із розмірністю більшою за три. Дана роботи містить модифікацію вихідного алгоритму, що дозволяє легко будувати довірчий еліпсоїд Петуніна у довільних просторах скінченної розмірності.*

*Ключові слова: довірчі еліпсоїди, багатовимірне ранжування даних, класифікація*

*Confidence ellipsoids and spheres have wide application in modern machine learning. They can be used for multivariate ranking, determining median, detecting outliers, determining quantile of prescribed level, resolving uncertainties etc. A lot of algorithms that use ellipsoids and spheres have been introduced. Such construction as minimum volume enclosing ellipsoid is used to build gap-tolerant classifier. Another example is support vector data description introduced by Tax and Duin that uses a hypersphere to detect outliers. Most of existing algorithm have statistical origin or were inspired by well-known support vector machine algorithm. Their purpose is to build a structure that tightly contains or separates training data. There exists a confidence ellipsoid introduced by Y. I. Petunin. Its main advantage is known confidence level. In this article we introduce a new algorithm of Petunin ellipsoid construction. The original algorithm has some constraints in its application for finite-dimensional space whose dimension is higher than three. This article contains a modification of the algorithm that allows easy construction of Petunin ellipsoid construction in any finite-dimensional space.*

*Key Words: confidence ellipsoids, data-ranking, classification*

Статтю представив д.ф.-м.н., проф. Буй Д.Б.

## Вступ

Машинне навчання сьогодні динамічно розвивається. Це зумовлено можливістю накопичувати та зберігати великі об'єми даних. Однією з актуальних задач машинного навчання є задача класифікації. Зокрема набирає популярності так звана задача однокласової класифікації (one-class classification), тобто задача відсіювання аномальних об'єктів. Одним з найпопулярніших методів класифікації є метод опорних векторів (support vector machine) [1]. Він будує гіперплощини у просторі ознак, що забезпечують найбільшу ширину проміжку, що розділяє об'єкти тренувальної вибірки. Своєю популярністю метод опорних векторів завдячує

відносній простоті. Задача оптимізації, що лежить в його основі, зводиться до задачі квадратичного програмування.

Метод опорних векторів будує «просту» лінійну поверхню. Сьогодні зроблено чимало спроб ускладнити форму цієї поверхні. Так у роботах [2] та [8] розглядаються методи відсіювання викидів за допомогою сфер. Автори назвали свій алгоритм Support vector data description через його подібність до методу опорних векторів. Основною ідеєю методу є побудова сфери найменшого об'єму, що містила б усі точки, що відмічені я точки заданої множини. При цьому усі тренувальні об'єкти, що є викидами, мають знаходитися ззовні побудованої сфери.

Ще однією широко застосовною конструкцією є еліпсоїд мінімального об'єму, що містить дану множину точок (minimum volume enclosing ellipsoid). У [6] досліджено ймовірність того, що об'єкт, який має бути віднесений до множини, буде невірно класифікований даним еліпсоїдом. Ще одним застосуванням є обмеження області, в якій лежать дані. У роботі [7] еліпсоїд мінімального об'єму застосовується разом із методом опорних векторів для побудови так званого gap-tolerant classifier.

Взагалі застосування еліпсоїдів дуже широке: їх використовують для оцінки медіани, оцінки квантилів певних рівнів а також для побудови багатовимірного ранжування даних. Автори роботи [5] пропонують алгоритм побудови еліпсоїда з найменшим ефективним радіусом (least effective radius) на основі коваріаційної матриці отриманих даних та застосовують його для виявлення викидів у бездротовій сенсорній мережі. Зокрема цікавою є процедура оновлення коваріаційної матриці та переходу до нового еліпсоїда на основі даних, що надходять.

Існує алгоритм, запропонований Ю. І Петуніним та наведений у роботах [3] та [4], який дозволяє побудувати еліпсоїд, що містить задану множину об'єктів. Спочатку він розглядався як наближення еліпсоїда мінімального об'єму, проте пізніше було виявлено, що даний еліпсоїд має фіксовану ймовірність попадання в нього, яка залежить лише від об'єму вибірки, за якою його будували: $\frac{n}{n+1}$. Основними застосуваннями даного еліпсоїда на даний момент є багатовимірне ранжування даних ([3]) та еліптичний пілінг ([4]).

Оригінальний алгоритм, запропонований Ю. І. Петуніним, має суттєвий недолік: його важко застосовувати у просторах із розмірністю більшою за три.

Метою даної роботи є запропонувати модифікацію вихідного алгоритму, що дозволяє будувати еліпсоїд Петуніна у просторах довільної скінченної розмірності. Асимптотична складність модифікованого алгоритму співпадає зі складністю оригінального.

Структура статті є наступною. У другій частині ми наведемо оригінальний алгоритм та вкажемо його недоліки. Третя частина буде присвячена модифікації даного алгоритму. Далі буде оцінено складність отриманого алгоритму ті наведено його порівняння з вихідним.

## Еліпсоїд Петуніна

У даній частині ми наведемо алгоритм побудови еліпсоїда Петуніна. Як і в роботах [3] та [4] для наочності опишемо побудову спочатку для простору $R^2$, а потім сформулюємо загальний алгоритм для довільного скінченновимірного простору. Отже, нехай ми маємо набір точок простору $R^m$ (позначимо його $M_n = \{\vec{x}_1, ..., \vec{x}_n\}$).

**Випадок *m*=2.** Знайдемо діаметр множини $M_n$. Нехай його визначають точки $(x_k, y_k)$ і $(x_l, y_l)$. Проведемо через точки $(x_k, y_k)$ і $(x_l, y_l)$ пряму $L$. Знайдемо вершини опуклої оболонки $(x_r, y_r)$ і $(x_q, y_q)$, що є найвіддаленішими від прямої $L$ та лежать по різні сторони від неї. Проведемо через точки $(x_r, y_r)$ і $(x_q, y_q)$ прямі $L_1$ і $L_2$, паралельні до $L$. Через точки $(x_k, y_k)$ і $(x_l, y_l)$ будуємо дві прямі $L_3$ і $L_4$, перпендикулярні до $L$. Перетин прямих $L_1$, $L_2$, $L_3$ і $L_4$ утворюють прямокутник, сторони якого мають довжини $a$ і $b$ (нехай, для визначеності, $a \le b$). Здійснимо поворот та перенос системи координат, так щоб лівий нижній кут прямокутника був розташований на початку нової системи координат з осями $Ox'$ і $Oy'$, а точки $(x_1, y_1)$, $(x_2, y_2)$, ..., $(x_n, y_n)$ перейшли в точки $(x_1', y_1')$, $(x_2', y_2')$, ..., $(x_n', y_n')$. Виконаємо стискання абсцис усіх точок $(x_1', y_1')$, $(x_2', y_2')$, ..., $(x_n', y_n')$ з коефіцієнтом $\alpha = \frac{a}{b}$ і одержимо сукупність точок $(\alpha x_1', y_1')$, $(\alpha x_2', y_2')$, ...,

$(\alpha x'_n, y'_n)$, що лежать у квадраті $S$. Знайдемо центр $(x'_0, y'_0)$ квадрата $S$ й обчислимо відстані від нього до кожної точки $r_1, r_2, ..., r_n$. Знайдемо найбільше число $R = \max(r_1, r_2, ..., r_n)$. Побудуємо коло з центром у точці $(x'_0, y'_0)$ і радіусом $R$. (Усі точки $(\alpha x'_1, y'_1)$, $(\alpha x'_2, y'_2)$, ..., $(\alpha x'_n, y'_n)$ тепер лежать всередині цього кола.) Застосовуємо до цього кола операцію розтягування уздовж осі $Ox'$ з коефіцієнтом $\beta = \frac{1}{\alpha}$ і зворотні перетворення повороту і переносу, одержуючи шуканий довірчий еліпс. Легко бачити, что середня складність цього алгоритму визначається складністю побудови опуклої оболонки і дорівнює $O(n \lg n)$.

**Випадок *m*>2.** Знайдемо діаметр множини $M_n$. Нехай $\vec{x}_k$ і $\vec{x}_l$ — дві вершини, що лежать на ньому. Проведемо через $\vec{x}_k$ та $\vec{x}_l$ пряму $L$. Здійснимо поворот і перенос системи координат, так щоб діаметр опуклої оболонки лежав на осі $Ox'_1$. Спроектуємо точки на підпростір, перпендикулярний $Ox'_1$. У даному підпросторі здійснемо аналогічні дії (побудову опуклої оболонки, перенос та поворот). Повторюватимемо дані кроки, доки розмірність підпростору, на який необхідно проектувати точки, не стане рівною одиниці. Після цього побудуємо найменший прямокутний паралелепіпед, сторони якого напрямлен вздовж координатних вісей, що містить точки $\vec{x}'_1, ..., \vec{x}'_n$. Стиснемо простір, перетворюючи прямокутний паралелепіпед у гіперкуб. Знайдемо центр $\vec{x}_0$ гіперкуба й обчислимо відстані $r_1, r_2, ..., r_n$ від нього до кожної точки. Знайдемо найбільше число $R = \max(r_1, r_2, ..., r_n)$. Побудуємо гіперкулю з центром у точці $\vec{x}_0$ і радіусом $R$. Застосуємо до цієї гіперкулі зворотні операції розтягування, повороту і переносу, одержуючи необхідний еліпсоїд у вихідному просторі.

Основним недоліком даного алгоритму є складність його застосування у випадку *m*>3. Так зокрема складною є побудова повороту на кожному кроці. У зв'язку з цим зменшення розмірності підпростору також створює складнощі при реалізації алгоритму. Практичне застосування алгоритму є зручним лише у дво- та тривимірному просторах.

### Альтернативний алгоритм

У цій частині ми дещо модифікуємо вихідний алгоритм побудови довірчого еліпсоїда з метою його зручного застосування для точок простору $R^m$, $m$>2. Помітимо, що основною метою алгоритму в цьому випадку є побудова такого лінійного перетворення простору, при якому обрані на кожному кроці діаметри множин (або їх опуклих оболонок) будуть паралельними до векторів базису простору. Після цього все зводиться до простого стискання та знаходження найбільш віддаленої від центру точки.

Нашою основною метою є чітко описати побудову зазначеного вище лінійного перетворення. Уявімо, що в процесі побудови ми на кожному кроці не виконували повороти, а просто проектували та зсували наші вектори. Тоді в кінці нам відомі усі вектори діаметрів опуклих оболонок. З побудови очевидно, що вони утворюють ортогональну систему векторів. Нормуємо її та позначимо $B_m = \{\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_m\}$.

Після цього нашим завданням є знайти унітарне перетворення простору, за якого дана ортонормована система векторів стане базисом. Ця задача стає тривіальною, якщо врахувати, що при оберненому перетворенні образами базисних векторів будуть вектори нашої системи $B_m$. Тобто фактично нам відома матриця оберненого перетворення:

$$U^{-1} = (\vec{a}_1 \mid \vec{a}_2 \mid \ldots \mid \vec{a}_m)$$

або ж

$$U = (\vec{a}_1 \mid \vec{a}_2 \mid \ldots \mid \vec{a}_m)^{-1}$$

На цьому кроці дуже зручним є те, що $U$ — унітарний оператор, а тому $U^{-1} = U^T$. Наступним кроком до спрощення процесу реалізації нашого алгоритму буде модифікація переходу до підпростору меншої розмірності. Нашою метою тут є зробити так, щоб зсув простору був одним. Для цього розглянемо перший крок алгоритму. Нехай ми знайшли вектори діаметра опуклої оболонки нашої множини точок (позначимо їх $\vec{x}_k$ та $\vec{x}_l$). Якщо не робити поворот та зсув, на першому кроці нам необхідно спроектувати наші точки на лінійний многовид $\vec{x}_k + L$, де $L$ — ортогональне доповнення прямої, визначеної вектором $\vec{x}_k - \vec{x}_l$. Фактично це гіперплощина, що проходить через $\vec{x}_k$ та ортогональна до $\vec{x}_k - \vec{x}_l$. Позначимо точки, отримані після проектування $M_n^{(1)} = \{\vec{x}_i^{(1)}, i = \overline{1, n-1}\}$. Зауважимо, що кількість різних точок після проектування зменшиться на

одну, бо за побудовою проекцією точки $\vec{x_k}$ буде $\vec{x_l}$. Покажемо, що для подальших кроків ми можемо не переходити до підпросторів меншої розмірності. Замість цього нам достатньо на кожному кроці виконувати проектування на лінійний многовид, підпростір якого є гіперплощиною, ортогональною до вектора діаметру опуклої множини. Нехай на деякому кроці ми маємо множину $M_n^{(k)} = \{\vec{x}_i^{(k)}, i = \overline{1, n-k}\}$ векторів простору $R^m$, що належать лінійному многовиду $y+L_1$, розмірність якого дорівнює $p$. Припустимо також, що ми знайшли вектор діаметру даної множини і він дорівнює $\vec{x}_l^{(k)} - \vec{x}_t^{(k)}$. Розглянемо тепер гіперплощину, ортогональну до цього вектора. Її рівняння має вигляд:

$$(\vec{x}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)}) = (\vec{x}_t^{(k)}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)})$$

При цьому проекція точки на дану гіперплощину має вигляд

$$P\vec{x} = \vec{x} - \frac{(\vec{x}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)}) - (\vec{x}_t^{(k)}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)})}{\| \vec{x}_l^{(k)} - \vec{x}_t^{(k)} \|^2} (\vec{x}_l^{(k)} - \vec{x}_t^{(k)})$$

Оскільки $\vec{x}_t^{(k)}$ лежить в $L_1$, то $\vec{y} + L_1 = \vec{x}_l^{(k)} + L_1$.

Тому якщо $\vec{x} \in \vec{x_t}^{(k)} + L_1$, то $\vec{x} = \vec{x_t}^{(k)} + \vec{y}, \vec{y} \in L_1$

$$P\vec{x} = \vec{x} - \alpha(\vec{x}_l^{(k)} - \vec{x}_t^{(k)}) = \vec{x_t}^{(k)} + \vec{y} - \alpha(\vec{x}_l^{(k)} - \vec{x}_t^{(k)})$$

де $\alpha = \frac{(\vec{x}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)}) - (\vec{x}_t^{(k)}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)})}{\| \vec{x}_l^{(k)} - \vec{x}_t^{(k)} \|^2}$. Також ми знаємо, що $\alpha(\vec{x}_l^{(k)} - \vec{x}_t^{(k)}) \in L_1$, а тому $P\vec{x} \in \vec{x_t}^{(k)} + L_1$. При цьому за означенням проекції $Px$ належить гіперплощині $(\vec{x}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)}) = (\vec{x}_t^{(k)}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)})$, яка теж є лінійним многовидом. Тому проекції наших точок лежатимуть у перетині двох лінійних многовидів, причому наша гіперплощина є лінійним многовидом $\vec{x}_t^{(k)} + L_2$, де $L_2$ — це підпростір розмірності $m$-1. Перетин цих лінійних многовидів має вигляд $\vec{x}_t^{(k)} + L_1 \cap L_2$ При цьому оскільки $\vec{x}_l^{(k)} - \vec{x}_t^{(k)}$ ортогональний до $L_2$, то $L_1 + L_2 = R^m$. Запишемо тепер формулу Грасмана, щоб знайти розмірність $L_1 \cap L_2$:

$$m = m - 1 + p - \dim L_1 \cap L_2$$

звідки

$$\dim L_1 \cap L_2 = p - 1$$

З урахуванням цього ми на кожному кроці можемо виконувати проекцію на гіперплощину перпендикулярну вектору діаметра, що проходить через одну з точок цього діаметра. При цьому насправді на кожному кроці розмірність многовида, у якому лежатимуть точки, зменшуватиметься. В оригінальному алгоритмі це перехід до підпростору, що перпендикулярний черговій осі координат, отриманій після повороту.

Таким чином у підсумку наш алгоритм набуває вигляду:

*Вхідні дані:* $M_n = \{\vec{x}_1, ..., \vec{x}_n\}$

*Алгоритм*

1. $M_n^{(0)} \leftarrow M_n, B = \varnothing, k = 0$;
2. Доки $k < m-1$ виконувати:
   2.1. Знайти $\vec{x_l}^{(k)}$ та $\vec{x_t}^{(k)}$ — точки, що визначають діаметр множини $M_n^{(k)}$;
   2.2. $B \leftarrow B \cup \left\{ \frac{\vec{x}_l^{(k)} - \vec{x}_t^{(k)}}{\| \vec{x}_l^{(k)} - \vec{x}_t^{(k)} \|} \right\}$;
   2.3. $M_n^{(k+1)} \leftarrow P_L M_n^{(k)}$, де $L$ — це гіперплощина, визначена рівнянням $(\vec{x}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)}) = (\vec{x}_t^{(k)}, \vec{x}_l^{(k)} - \vec{x}_t^{(k)})$;
   2.4. $k \leftarrow k+1$;
3. Сформувати матрицю $U$, стовпчиками якої є вектори множини $B$;
4. $M_n^{'} \leftarrow U^T M_n$;
5. Знайти мінімальний паралелепіпед зі сторонами, напрямленими вздовж координатних осей, що містить множину $M_n^{'}$. Нехай $\vec{c}$ — центр цього паралелепіпеда;
6. Побудувати перетворення $S$, що переводить цей паралелепіпед в куб;
7. $M_n^{''} \leftarrow SM_n^{'}$, $\vec{c}^{'} \leftarrow S\vec{c}$;
8. $R \leftarrow \max \left\{ \| \vec{x} - \vec{c}^{'} \|, x \in M_n^{''} \right\}$;
9. $E \leftarrow \frac{1}{R^2} \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix}$;
10. $E \leftarrow US^T ESU^T$, $\vec{c} \leftarrow U\vec{c}$;

*Вихідні дані:* $E$ — матриця вихідного еліпсоїда, $\vec{c}$ — центр.

Застосування запропонованого алгоритму можливе лише у випадку, коли шуканий еліпсоїд існує та є не виродженим. Іншими словами не

має існувати підпростору вихідного простору $R^m$, що містить вхідну множину точок.

### Складність

У даній частині ми вважатимемо операції проектування точки на афінний підпростір, лінійного перетворення простору та порівняння двох точок елементарними, оскільки

Асимптотично найскладнішою частиною обох алгоритмів є знаходження діаметра скінченної множини точок. Кількість операцій, необхідна для пошуку діаметра перебором, дорівнює $\frac{n(n-1)}{2}$. Проте цей результат може бути покращений для дво- та тривимірних просторів за допомогою опуклої оболонки. Можна побудувати опуклу оболонку та здійснювати перебір по її вершинах. Асимптотична складність таких дій буде $O(n\lg n+\frac{z(z-1)}{2})$, де $z$ — кількість вершин опуклої оболонки. В найгіршому випадку складність залишиться квадратичною, проте для більшості вибірок кількість вершин опуклої оболонки є незначно у порівнянні з розміром вибірки, тому складність для таких вибірок буде $O(n\lg n)$. Для просторів $R^m$, $m$>3 складність алгоритмів буде $O(n^2)$

### Порівняння на практиці

Порівняння алгоритмів проводилося для дво-тривимірного випадку. Обидва алгоритми було реалізовано мовою R. Було згенеровано 100 вибірок, кожна з яких містила по 200 векторів. Кожна вибірка містила по 100 векторів з двох нормальних розподілів з однаковими діагональними матрицями коваріації та різними векторами математичних сподівань. Для кожного алгоритму розглядався сумарний час роботи на всіх 100 вибірках. Прийнявши сумарний час роботи оригінального алгоритму у двовимірному випадку за одиницю, результати мають наступний вигляд:

| | Оригінальний алгоритм | Модифікований алгоритм |
|---|---|---|
| $R^2$ | 1 | 2,0082 |
| $R^3$ | 1,9951 | 2,9404 |

Таблиця 1. Порівняння часу роботи алгоритмів.

Слід зауважити, що для тривимірного простору після першого проектування в оригінальному алгоритмі задача зводиться до випадку $m$=2. Це означає, що після повороту ми можемо на площині, перпендикулярній, $Ox_1'$ застосувати алгоритм для випадку $m$=2 та побудувати шуканий прямокутник, а потім за ним побудувати паралелепіпед у просторі. Це було враховано при програмній реалізації оригінального алгоритму. Таке спрощення зменшує кількість необхідних пошуків діаметру множини на 1 і нам необхідно буде їх здійснювати $m-1$ разів. Модифікований алгоритм при цьому вимагає здійснювати пошук діаметру множини $m$ разів. В іншому алгоритми здійснюють подібні дії. Таким чином співвідношення між часом роботи двох алгоритмів на одних і тих самих вхідних даних має бути близьким до $\frac{m-1}{m}$, що і відображено у результатах.

Як випливає з результатів порівняння, модифікований алгоритм має дещо більший час роботи. Проте зі збільшенням розмірності простору дана різниця нівелюється. Крім того, як ми вже казали раніше, застосування оригінального алгоритму у просторих з розмірністю, більшою за 3, є проблематичним у зв'язку зі складністю побудови поворотів

### Висновки

У статті запропоновано новий алгоритм побудови еліпсоїда Петуніна. Даний еліпсоїд застосовується для багатовимірного ранжування та пілінгу даних для оцінки медіани. Наведено складність отриманого алгоритму та проведено порівняння з оригінальним алгоритмом. Основною його перевагою є легкість застосування для просторів довільної скінченної розмірності. Недоліком є дещо більший час роботи порівняно з оригінальним алгоритмом на просторах розмірності 2 і 3. Подальшим покращенням могло б стати вдосконалення процесу знаходження діаметру множини.

Ще однією проблемою, яка виникає при побудові даного еліпсоїда є досягнення наперед визначеного рівня значущості. Зі зростанням кількості тренувальних даних рівень значущості згідно ймовірності потрапляння до еліпсоїда буде прямувати до нуля. Для отримання, наприклад, рівня значущості 0,05 необхідно 19 тренувальних точок. Тому постає питання вибору цих 19 точок так, щоб отриманий еліпсоїд містив тренувальну

вибірку та деякою мірою відображав сукупність, з якої було взято дані.

**Список використаних джерел**


1. *Herbrich R.,* Learning Kernel Classifiers: Theory and Algorithms. / R. Herbrich — MIT Press — Cambridge, MA, 2001 — 382 P.
2. *Tax D.M.J.,* Support vector data description. / D.M.J. Tax, R.P.W. Duin // Machine learning — 2004. — 54(1). — P. 45-66.
3. *Клюшин Д.А.,* Еліптична функція статистичної глибини даних. / Д.А. Клюшин, М.В. Присяжна // Вісник Київського національного університету імені Тараса Шевченка, Серія фізико-математичні науки. — 2014. — № 1. — С.179-184.
4. *Lyashko S.I.,* Multivariate ranking using elliptical peeling. / S.I. Lyashko, D.A. Klyushin, V.V. Alexeyenko // Cybernetics and Systems Analysis — 2013. — 49(4). — P. 511-516.
5. *Zhang Y.,* Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine. / Y. Zhang, N. Meratnia, P.J.M. Havinga // Ad hoc networks. — 2012. — 11(3). — P. 1062-1074.
6. *Dolia A.N.,* The minimum volume covering ellipsoid estimation in kernel-defined feature spaces. / A.N. Dolia, T. Bie, C.J. Harris // ECML, 18-22 September 2006, Berlin, Germany. — 2006. — P. 630-637.
7. *Shivaswamy, P.,* Ellipsoidal kernel machines. / P.Shivaswamy, T. Jebara // AISTATS, 21-24 March 2007, San Juan, Puerto Rico. — 2007 — P. 484-491.
8. *Wang J.,* Pattern classification via single spheres. / J. Wang, P. Neskovic, L.N. Cooper // Discovery Science, 8-11 October 2005, Singapore, — 2005 — P. 241–252.


**References**


1. HERBRICH R. (2001) *Learning Kernel Classifiers: Theory and Algorithms* MIT Press, Cambridge, MA
2. TAX D.M.J., DUIN R.P.W. (2004) Support vector data description. *Machine learning* 54(1). P. 45-66.
3. KLYUSHIN D.A., PRYSIAZHNA M.V. (2014) Elliptical statistical data depth function. *Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics*, 1. P.179-184.
4. LYASHKO S.I., KLYUSHIN D.A., ALEXEYENKO V.V. (2013) Multivariate ranking using elliptical peeling. *Cybernetics and Systems Analysis* 49(4). P. 511-516.
5. ZHANG Y., MERATNIA N., HAVINGA P.J.M. (2012) Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine. *Ad hoc networks.* 11(3). P. 1062-1074.
6. DOLIA A. N., BIE T., HARRIS C. J. (2006) The minimum volume covering ellipsoid estimation in kernel-defined feature spaces. In *ECML*. Berlin, Germany, 18th to 22nd September 2006. P. 630-637.
7. SHIVASWAMY P., JEBARA T. (2007). Ellipsoidal kernel machines. In *AISTATS*. San Juan, Puerto Rico, 21st to 24th March 2007 . P. 484-491.
8. WANG J., NESKOVIC P., COOPER L. N. (2005) Pattern classification via single spheres. In *Discovery Science*. Singapore, 8th to 11th October 2005. P. 241–252.


Надійшла до редколегії 29.05.2015