УДК61:004.651 (075.8)

АЛГОРИТМ КЛАСИФІКАЦІЇ ПОЛІТРАВМ МЕТОДОМ ІНДУКЦІЇ ДЕРЕВА РІШЕНЬ

Р. М. Борис, В. П. Марценюк[1]

ДП "Український науково-дослідний інститут медицини транспорту МОЗ України"
Тернопільський державний медичний університет імені І. Я. Горбачевського[1]

У роботі розроблено і програмно реалізовано метод індукції дерева рішень для задачі класифікації політравм на основі ряду біохімічних показників.

Алгоритм вибору атрибуту використовує значення приросту інформації. Проект реалізовано в середовищі Netbeans на основі Java-класів.

Ключові слова: політравма, прийняття рішень, дерево рішень, Java, SQL

АЛГОРИТМ КЛАССИФИКАЦИИ ПОЛИТРАВМ МЕТОДОМ ИНДУКЦИИ ДЕРЕВА РЕШЕНИЙ

Р. М. Борис, В. П. Марценюк[1]

ГП "Украинский научно-исследовательский институт медицины транспорта МЗ Украины"
Тернопольский государственный медицинский университет имени И. Я. Горбачевского[1]

В работе разработан и программно реализован метод индукции дерева решений для задачи классификации политравм на основании ряда биохимических показателей.

Алгоритм выбора атрибута использует значение прироста информации. Проект реализован в среде Netbeans на основе Java- классов.

Ключевые слова: политравма, принятие решений, дерево решений Java, SQL.

CLASSIFICATION ALGORITHM POLYTRAUMA BY INDUCTION OF DECISION TREES

R. M. Borys, V. P. Martsenyuk[1]

SE "Ukrainian Scientific Research Institute of Transport Medicine of MPH of Ukraine
Ternopil State Medical University by I. Ya. Horbachevsky[1]

The work program is developed and implemented induction decision tree method for classification of polytrauma based on a number of biochemical parameters.

The selection algorithm uses the value of the attribute information gain. The project was implemented in the medium Netbeans Java- based classes.

Key words: polytrauma, decision making, decision tree Java, SQL.

Вступ. Під політравмою мають на увазі складний патологічний процес, зумовлений пошкодженням кількох анатомічних ділянок або сегментів кінцівок. Проблему становить правильне та своєчасне діагностування політравм, особливо в умовах надзвичайних ситуацій або військових дій, коли пацієнт знаходиться у стані без свідомості.

Мета даної роботи - розробити і програмно реалізувати алгоритм класифікації політравм з використанням методу індукції дерева рішень.

Вирішувана проблема належить до широкого класу задач диференціальної діагностики. В медицині поняття «диференціальної діагностики» означає системний підхід, який грунтується на доказовості, для визначення причини симптомів, що спостерігаються, у випадку, коли є кілька альтернативних пояс-

нень, а також для зменшення переліку можливих діагнозів.

Сьогодні медичне діагностування може виконуватися автоматично з використанням комп'ютеризованих систем та алгоритмів. Такі системи переважно називаються діагностичними системами підтримки прийняття рішень або медичними діагностичними системами. Вони належать до загальнішого класу клінічних систем підтримки прийняття рішень [Марценюк, 2004—2012]. Метою таких систем є системний супровід лікаря в процесі диференційної діагностики. Багато з таких систем можуть надавати результати навіть коли не вистачає даних, тобто в умовах невизначеності, і що найважливіше - вони не обмежені щодо кількості інформації, яку можуть зберігати та обробляти.

Одним з підходів, що відображає природний процес мислення при диференційній діагностиці, є метод індукції дерева рішень. У період кінця 1970 - початку 1980 років J. R. Quinlan [Quinlan, 1986] розробив алгоритм побудови дерева рішень ID3 (ітеративний дихотомайзер). Пізніше J. R. Quinlan представив алгоритм C4.5 (наступник ID3), який став еталоном, з яким часто порівнюються новітні алгоритми в галузі машинних знань. У 1984 році група статистиків (L. Breiman, J. Friedman, R. Olshen, C. Stone) опублікували роботу щодо Classification and Regression Trees (CART) [Breiman, 1984], в якій описали побудову бінарних дерев рішень. Алгоритми ID3 та CART, не зважаючи на те, що були розроблені незалежно і приблизно у той же час, реалізують подібний підхід до навчання дерев рішень на основі навчальних даних. При цьому дерева рішень будуються в результаті рекурсивної процедури типу «зверху-вниз». Більшість алгоритмів індукції дерев рішень також відповідають цьому загальному підходу. При цьому навчальна множина рекурсивно поділяється на менші підмножини по мірі того, як будується дерево.

Задача індукції дерева рішень формулюється таким чином. Маємо множину D), що містить *N* наборів навчальних даних. При цьому кожен *i*-й набір $(A_1^i, A_2^i, ..., A_p^i, C^i)$ складається з вхідних даних - атрибутів $A_1, ..., A_p$ та вихідних даних - атрибуту класу *C*. Атрибути $A_1, ..., A_p$ можуть приймати як чисельні, так і категоріальні значення. Атрибут класу *C* приймає одне з *K* дискретних значень: $C \in \{1, ..., K\}$. Метою є прогнозування деревом рішень значення атрибуту класу *C* на основі значень атрибутів $A_1, ..., A_p$. При цьому слід максимізувати точність прогнозування атрибуту класу, а саме $P\{C = c\}$ на термінальних вузлах для довільного $C \in \{1, ..., K\}$. Алгоритми індукції дерев рішень автоматично розбивають на вузлах значення чисельних атрибутів A_j на два інтервали: $A_i \le x_i$ та $A_i > x_i$, а категоріальних атрибутів A_i - на дві під множини: $A_j \in S_j$, $A_j \notin S_j$. Розбиття чисельних атрибутів грунтується, як правило, на мірах на основі ентропії, або індексі Джині [Han, 2001]. Процес розбиття рекурсивно повторюється до тих пір, поки не спостерігатиметься покращення точності прогнозування. Останній крок включає вилучення вузлів для уникнення оверфітінгу моделі. У результаті ми повинні отримати множину правил, що йдуть від кореня до кожного термінального вузла, містять нерівності для чисельних атрибутів та умови включення для категоріальних атрибутів.

Метод індукції дерева рішень. За основу взято таку рекурсивну процедуру роботи [Han, 2001].

Генерація дерева рішень

Вхідні дані: *D* - множина навчальних наборів даних $(A_1^i, A_2^i, ..., A_p^i, C^i)$

Вихідні дані: дерево рішень

Метод:

1. Створити вузол *N*.
2. Якщо усі набори в *D* належать до спільного класу *C*, тоді повернути вузол *N* як листок із назвою класу *C*.
3. Якщо список атрибутів (а отже і *D)* є порожнім, тоді повернути вузол *N* як листок із назвою найпоширенішого класу в *D*.
4. Застосувати *Алгоритм відбору атрибуту* із списку атрибутів і для множини *D* з метою відшукання «найкращого» атрибуту поділу.
5. Вилучити атрибут поділу із списку атрибутів.
6. Для кожної умови поділу *j* для атрибуту поділу розглянемо D_j - множину наборів з *D,* що задовольняють умову поділу j.
7. Якщо D_j - порожня, тоді приєднати до вузла *N* листок під заголовком найпоширенішого класу в D_j, інакше - приєднати до *N* вузол, що повертається рекурсивним викликом методу *Генерація дерева рішень* з вхідними даними D_j та список атрибутів.
8. Кінець циклу кроку 6.
9. Повернути вузол *N*

В основу *Алгоритму відбору атрибуту* на *j*-му кроці рекурсії покладено такий інформаційний показник

$$Gain(A_i) = Info(D_j) - Info_{A_i}(D_j). \quad (1)$$

Тут

$$Info(D_j) = -\sum_{k=1}^{K} p_k^j \log_2(p_k^j) \quad (2)$$

- інформація, потрібна для класифікації набору $(A_1, A_2, ..., A_p)$ в D_i,

$$Info_{A_i}(D_j) = \sum_{l=1}^{K_i} \frac{\#(D_j^l)}{\#(D_j)} Info(D_l) \quad (3)$$

- інформація, потрібна для класифікації $(A_1, A_2, ..., A_p)$ в D_j після поділу D_j на підмножини D^l_j відповідно до значень атрибуту A_j

У формулі (2) ймовірність того, що довільний набір з D_i належить множині C_{k,D_j} оцінюється як $p_k^j = \frac{\#(C_{k,D_j})}{\#(D_j)}$, де C_{k,D_j} - множина наборів з D_j для яких атрибут класу $C = k$. Тут $\#(\bullet)$ – кількість елементів в множині.

У формулі (3) $\frac{\#(D_j^l)}{\#(D_j)}$ - оцінка ймовірності того, що довільний набір з D_j належить множині D^l_j, де D^l_j множина наборів з D_j, для яких атрибут $A_i = a_i^l$. Тут атрибут $A_i \in \{a_i^1, a_i^2, ..., a_i^{K_i}\}$.

Отже, $Gain(A_i)$ оцінює зменшення інформації, необхідної для класифікації довільного набору даних в D_j за рахунок відомого значення атрибуту A_i. Таким чином, з наявних атрибутів на кожному вузлі дерева рішень для умови поділу слід відбирати атрибут A_{i^*} з найбільшим значенням $Gain(A_{i^*})$. У результаті такого вибору для завершення процесу класифікації набору даних в D_j вимагатиметься найменше інформації.

Програмна реалізація. Метод реалізовано в середовищі розробки Netbeans на мові програмування Java. Базу навчальних даних розгорнуто на сервері MySQL. На рисунку 1 представлено концептуальну модель інформаційної системи. У класі DecisionTree безпосередньо реалізовано метод індукції дерева рішень. У клас DataManager надходять виклики від DecisionTree на виконання запитів до бази даних mysql щодо отримання навчальних даних.

База даних mysql складається з двох таблиць - таблиці attribute, призначеної для зберігання інформації про атрибути, та таблиці categorized_data - для

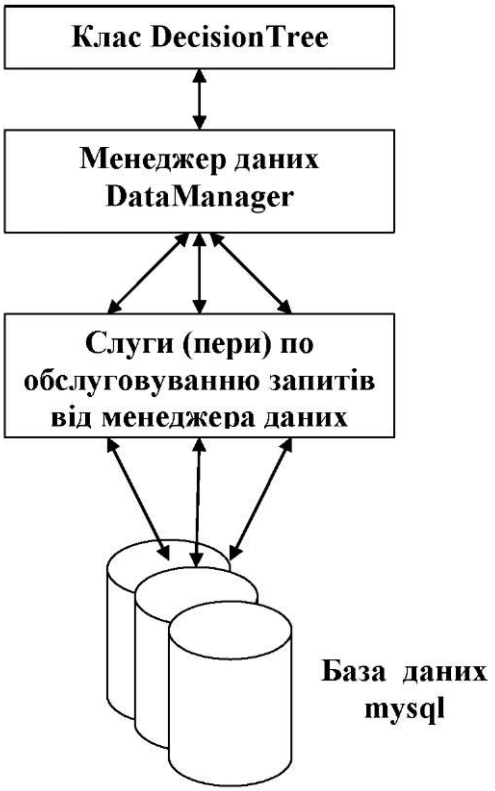

Рис.1. Концептуальна модель інформаційної системи індукції дерева рішень.

наборів навчальних даних. Структура таблиць намові SQL для класифікації політравм наведена нижче:

```
CREATE TABLE mysql.attribute (
        id integer not null unique,
        attribute_name varchar(25),
        attribute_field_name varchar(25),
    primary key (id)
) ENGINE=InnoDB;
CREATE TABLE mysql.categorised_data (
        id integer not null unique,
        A1 varchar(12),
        A2 varchar(8),
        A3 varchar(7),
  ……..
        A21 varchar(7),
        class varchar(28),
    primary key (id)
) ENGINE=InnoDB;
```

Програмні класи проекту включено до пакета decision_tree.model. Сюди входять beans-класи Attribute, Attribute_for_list та CategorisedData для роботи з даними відповідних таблиць. SQL-запити щодо отримання відповідних даних, включаючи розрахунки інформаційних показників, реалізовано в класі AttributeListPeer.

Клас DecisionTree є нащадком класу DefaultTreeModel пакету javax.swing.tree. Він має два елементи класу: m_dataManager - менеджер даних та m_htAttribute_list - хеш-таблиця із списком атрибутів. Хеш-таблиця із списком атрибутів (у методах класу DecisionTree виступає під назвою htAttribute list) створюється для кожного вузла дерева рішень. Вона має два призначення - поряд із списком включених для даного вузла атрибутів зберігати умови поділу (splitting conditions), які перейшли до даного вузла від вузлів-батьків. Кожен вузол дерева рішень є об'єктом класу DefaultMutableTreeNode. В якості об'єкта кожен вузол зберігає об'єкт класу NodeObject, декларація якого наведена нижче:

```
class NodeObject {
   Attribute attribute;
   Hashtable htAttribute_list;
   String splitting_criterion;
   String sLabel;
   public String toString() {
     if (splitting_criterion.matches(“”)) { return sLabel;
}
      else return “if ““ + splitting_criterion + ““ then ““
+ sLabel + ““““; }
  }
```

Тут attribute - атрибут, який повертається методом Attribute selection method, splitting criterion - умова поділу, яка переходить від батьківського вузла, sLabel - надпис на вузлі. Хеш-таблиця htAttribute list використовується для побудови наборів навчальних даних D_j для кожного із вузлів і має таку структуру:

Тип ключа	int
Тип об'єкта	Attribute for list
Структура об'єкта	Attribute attribute; Hashtable htSplittingoutcomes; String splitting criterion; boolean included;

Тут included - булева змінна-прапорець належності атрибуту attribute до списку атрибутів даного вузла. Можна показати, що коли included=true, то вузол з назвою attribute є для даного вузла дочірнім (на певному нижчому рівні ієрархії). У випадку, коли атрибут attribute не входить до списку атрибутів для даного вузла (included=false), то вузол з назвою attribute є батьківським (на певному рівні ієрархії), а в змінній splitting criterion зберігається умова поділу, якій підлягає даний вузол відносно батьківського вузла attribute.

Хеш-таблиця htSplitting_outcomes містить усі можливі наслідки (умови поділу) щодо атрибуту attribute.

Метод Generate_decision_tree є безпосередньою реалізацією методу індукції дерева рішень. Заголовок методу має вигляд:

private DefaultMutableTreeNode Generate_decision_ tree (Hashtable htAttribute list, DefaultMutable TreeNode dmtnSubroot, String splitting_criterion).

В якості аргументів метод використовує кореневий вузол дерева, список пов'язаних з ним атрибутів htAttribute list та умову поділу splitting criterion. В якості значення метод повертає дочірній вузол типу DefaultMutableTreeNode. Шляхом рекурсивного виклику методу Generate decision tree будується дерево рішень.

З метою візуалізації представлення дерева використано клас javax. swing JTree. При цьому дерево рішень створюється виводиться за допомогою операторів:

dtDecision tree = new DecisionTree(dmtnRoot, dataManager, htAttributelist);
jTreel .setModel(dtDecision_tree);

SQL-реалізація розрахунку інформаційних показників. Ключовим в реалізації методу Attribute selection method є розрахунок інформаційних показників $Info(D_j)$ та $Info_{A_i}(D_j)$ на j-му кроці рекурсії для атрибута A_j. Так, показник розраховується методом

public double getInfo$D$ (DataManager dataManager, Hashtable htAttribute_list)

Зауважимо, що множина наборів D_j тут описується хеш-таблицею htAttribute list, з якої отримуємо перелік включених атрибутів sAttribute list та умов поділу sConditions.

Мова структурованих запитів SQL має досить гнучкі засоби для реалізації алгоритмів в галузі машинних знань. Так, використавши вкладені запити, псевдоніми та агрегативні функції можна розрахувати в результаті виконання такого SQL-запиту:

```
String sql = “select -SUM((Alias1.Ci/
Alias2.D)*(LOG(Alias1.Ci/Alias2.D)/LOG(2))) from “
+
  “(select SUM(1) as Ci from (select “ + sAttribute_list
+ “,class from categorised_data “ +
(sConditions.matches(“”)?””: “ where “ + sConditions)
+ “)Alias3 group by Alias3.class)Alias1, “ +
  “(select SUM(1) as D from (select “ + sAttribute_list
+ “ from categorised_data “ +
(sConditions.matches(“”)?””: “ where “ + sConditions)
+ “)Alias4)Alias2”;
```

Показник $Info_{A_i}(D_j)$ розраховується методом

public static double getInfo_AD(DataManager dataManager, int i, Hashtable htAttribute list)

Остаточно $Info_{A_i}(D_j)$ обчислюється в результаті виконання такого SQL-запиту:

String sql = "select SUM((Alias1.Dj/Alias2.D)*Alias3.InfoDj) from " + "(select SUM(1) as Dj from (select * from categorised_data " + (sConditions.matches("")?"": "where " + sConditions) + ")Alias6 group by Alias6." + ((Attribute_for_list)htAttribute_list.get(A)).attribute.getAttributeFieldName() + ")Alias1, " + "(select SUM(1) as D from (select " + sAttribute_list + " from categorised_data " + (sConditions.matches("")?"": "where " + sConditions) + ")Alias7)Alias2, " + "(select -SUM((Alias4.Ci/Alias5.D)*(LOG(Alias4.Ci/Alias5.D)/LOG(2))) as InfoDj from " + "(select SUM(1) as Ci from (select " + sAttribute_list + ",class from categorised_data " + (sConditions.matches("")?"": "where " + sConditions) + ")Alias8 group by Alias8.class)Alias4, " + "(select SUM(1) as D from (select " + sAttribute_list + " from categorised_data " + (sConditions.matches("")?"": "where " + sConditions) + ")Alias9)Alias5)Alias3";

Далі за допомогою розроблених програмних класів побудуємо дерево рішень щодо класифікації політравм на основі даних 6-ти класифікаційних груп, а саме:

- черепно-мозкова та скелетна травми мали місце 2 години тому;
- черепно-мозкова та скелетна травми з кровотечею мали місце 2 години тому;
- черепно-мозкова та скелетна травми мали місце 12 годин тому;
- черепно-мозкова та скелетна травми з кровотечею мали місце 12 годин тому;
- черепно-мозкова та скелетна травми мали місце 24 години тому;
- черепно-мозкова та скелетна травми з кровотечею мали місце 24 години тому.

Використано таку таблицю атрибутів:

INSERT INTO mysql.attribute (id, attributename, attribute field name) VALUES (1, 'Mass", " A 1 '). (2, 'Zag.bil.', "A2'). (3, "AsAT. 'A3'),(4, "A1AT. "A4'). (5, 'GP', 'A5'), (6, 'GR', 'A6'), (7, VG', 'A7'), (8, 'SOD", 'A8'), (9, 'Catalaza', 'A9'), (10, MDA', 'A10'), (11, DK', 'A11'), (12, 'TsP', 'A12'),(13, 'TsIK', 'A13'), (14, 'API', 'A14'), (15, 'IgA', 'A15'),(16, 'IgM', 'A16'), (17, 'Ig G', 'A17'), (18, '11-2', 'A18'), (19, '11-6', 'A19'), (20, '11-10', 'A20'), (21, 'TNF-a', 'A21');

Набори включають лише категоріальні дані (попередньо оброблені), наприклад:

INSERT INTO mysql.categorised_data (id, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15,A16,A17,A18,A19, A20, A21, class) VALUES

(1, 'low', 'low', 'high', 'high', 'normal', 'low', 'low', 'low', 'low', 'high', 'high', 'high', 'high', 'low', 'high', 'high', 'high', 'high', 'high', 'high', 'high', 'craniocerebral_injury+orthopedic_trauma_2_hours');

На рисунку 2 представлене побудоване дерево рішень. Час, затрачений на індукування дерева - 3104 мілісекунди.

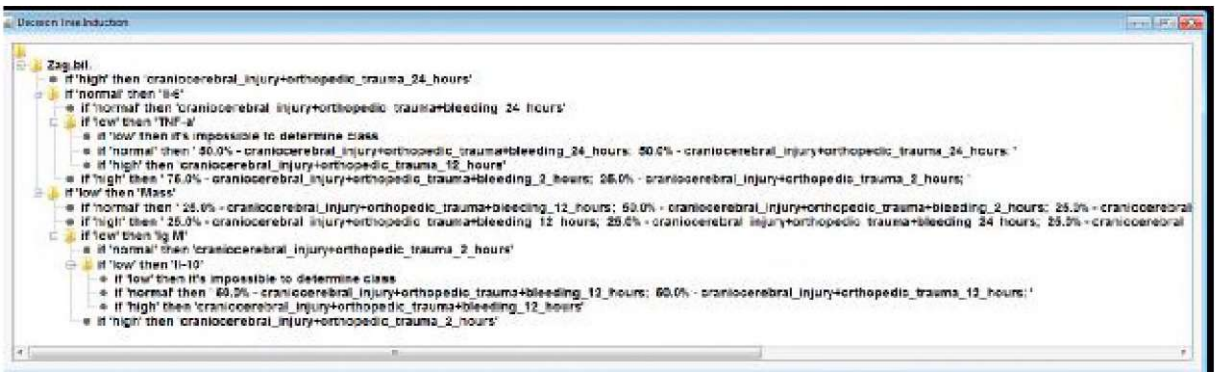

Рис. 2. Дерево рішень для класифікації політравм.

Висновки. У роботі розглянуто питання розробки і програмної реалізації методу індукції дерева рішень на основі інформаційних показників для побудови класифікаційного алгоритму політравм.

На даному експериментальному прикладі продемонстровано, що такий підхід дозволяє розробити систему підтримки клінічних рішень.

Показано, що мова SQL має достатні синтаксичні можливості, які дозволяють розрахувати інформаційні показники на основі таблиць баз даних.

За рахунок використання Java-класі в дана реалізації методу індукції дерева рішень є веб-інтегрованою.

Перспективи подальших досліджень - аналіз продуктивності програмного продукту залежно від кількості біохімічних показників та обсягу наборів навчальних даних.

Література

1. Соколов В. А. Множественные и сочетанные травмы / В. А. Соколов. - М. : "ГЭОТАР", 2006 г.
2. J.Han and M.Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, San Francisco, 1st edition 2001.
3. T.Hastie, R.Tibshirani and J.H.Friedman The Elements of Statistical Learning, Springer, New York, 1st edition 2001.
4. C.Ordonez, Comparing association rules and decision trees for disease prediction. In Proc. ACM HIKM Workshop, 2006, pp. 17-24.
5. C.Ordonez, Integrating K-means clustering with a relational DBMS using SQL, IEEE Transactions onKnowledge andData Engineering (IKDE) 18(2) (2006), 188-201.
6. J.R.Quinlan. Induction of decision trees. Machine Learning, 1:81-106,1986.
7. J.R.Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann 1993.
8. LBreiman, J.Friedman, R.Olshen and C.Stone. Classification and Regression Trees. Wadsworth International Group, 1984.
9. МарценюкВ. П. О программной среде проектирования интеллектуальных баз данных/ В. П. Марценюк, Н. О. Кравец // Клиническая информатика и телемедицина - 2004. - №1.-С. 47-53.
10. Математичні моделі в системі підтримки прийняття рішень страхового забезпечення лікування онкологічних захворювань: підхід на основі динаміки Гомперца / В. П. Марценюк, І. Є. Андрущак, І. С. Гвоздецька, Н. Я. Климук // Доповіді Національної академії наук України. -2012. -№10. -С. 34-39.
11. Марценюк В. П. Підхід на основі актуальних математичних моделей до задач страхової медицини / В. П. Марценюк, І. Є. Андрущак, Н. Я. Климук //Медична інформатика та інженерія.-2010.-№4.-С. 85-87.
12. Марценюк В. П. О модели онкологического заболевания со временем пребывания на стадии в соответствии с распределением Гомперца / В. П. Марценюк, Н. Я. Климук // Проблемы управления и информатики. Международный научно-технический журнал. - 2012. - № 6. - С. 137—143"
13. Марценюк В. П. Медична інформатика. Інструментальні та експертні системи/В. П. Марценюк, А. В. Семенець. - Тернопіль: Укрмедкнига. 2004. - 222 с.