Міністерство освіти і науки України

Тернопільський національний економічний університет

Факультет комп’ютерних інформаційних технологій

Кафедра КІ

Лабораторна робота №4 з дисципліни «Дослідження комп’ютерних систем штучного інтелекту»

Виконав:

Студент групи КСМм-51

Яцик С.М.

Перевірив :

Вербовий С.О.

Тернопіль-2013

Тема: Машинне навчання на основі прикладів

Мета: Вивчити можливості засобів мови програмування Пролог для створення систем машинного навчання.

# 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Із всіх форм навчання освоєння понять на прикладах є найпоширенішим і наочним. Вибір алгоритмів навчання залежить від того, на якій мові представлені досліджувані поняття. У даній роботі розглядаються програми, здатні до вивчення понять, представлених у вигляді **правил виводу.**

При навчанні на прикладах системі надаються приклади (зразки) для вивчення на основі яких вона робить певні висновки, тобто створює правила за якими вона буде розпізнавати (класифікувати) інші об’єкти.

Навчання на прикладах називається також **індуктивним навчанням***.* На прикладах може бути організоване навчання виконанню завдань декількох типів: з їхньою допомогою можна навчити програму діагностувати захворювання людей; прогнозувати погоду; приймати кращі фінансові рішення; управляти динамічною системою. Одна із прикладних областей відноситься до отримання знань для експертних систем. Ще одним способом використання методів машинного навчання є виявлення прихованих закономірностей у базах даних (Knowledge Discovery in Databases — KDD), що називається також інтелектуальним аналізом даних.

**Варіант 17**

example( nut, [ size = small, shape = compact, holes = 1]).

example( key, [ size = large, shape = other, holes =1]).

example( pen, [ size = large, shape = long, holes = none]).

example( scissors, [ size = large, shape = long, holes =2]).

example( key, [ size = large, shape = long, holes =3]).

example( pen, [ size = large, shape = long, holes = none]).

example( scissors, [ size = large, shape = other, holes = 2]).

example( key, [ size = small, shape = other, holes = 2]).

Код програми на мові Пролог:

attribute( size, [ small, large]).

attribute( shape, [ long, compact, other]).

attribute( holes, [ none, 1, 2, 3, many]).

% БД прикладів для навчання

example( nut, [ size = small, shape = compact, holes = 1]).

example( key, [ size = large, shape = other, holes =1]).

example( pen, [ size = large, shape = long, holes = none]).

example( scissors, [ size = large, shape = long, holes =2]).

example( key, [ size = large, shape = long, holes =3]).

example( pen, [ size = large, shape = long, holes = none]).

example( scissors, [ size = large, shape = other, holes = 2]).

example( key, [ size = small, shape = other, holes = 2]).

%навчання на основі простих правил виводу

:- op( 300, xfx, <==).

%learn( Class): зібрати навчальні приклади в список, сформувати й вивести

% опис для класу Class, потім внести в базу даних відповідне

% правило, що стосується класу Class

learn( Class) :-

%Зібрати приклади

bagof(example(ClassX, Obj), example(ClassX,Obj), Examples),

learn( Examples, Class, Description), % Сформувати правило

nl, write( Class), write(' <== '), nl, % Вивести правило

writelist( Description),

assert( Class <== Description). % Внести правило в базу даних

% learn( Examples, Class, Description):

% опис Description охоплює точно всі приклади кл. Class у списку Examples

learn( Examples, Class, []) :-

% Немає прикладів, які потрібно було б охопити описом

not( (member( example(Class,\_), Examples))).

learn( Examples, Class, [Conj | Conjs]) :-

learn\_conj( Examples, Class, Conj),

% Видалити приклади, які відповідають умові Conj

remove( Examples, Conj, RestExamples),

learn( RestExamples, Class, Conjs). % Охопити описом інші приклади

%learn\_conj( Examples, Class, Conj):

% Conj - це список значень атрибутів, якому відповідають деякі

% приклади класу Class і не відповідає жоден приклад якогось іншого класу

learn\_conj( Examples, Class, []) :-

not( (member( example( ClassX ,\_), Examples), % Немає прикладів якогось

ClassX \== Class )), !. % іншого класу

learn\_conj( Examples, Class, [Cond | Conds]):-

choose\_cond( Examples, Class, Cond), % Вибрати значення атрибута

filter( Examples, [ Cond], Examples1),

learn\_conj( Examples1, Class, Conds).

choose\_cond( Examples, Class, AttVal):-

findall( AV/Score, score( Examples, Class, AV, Score), AVs),

best( AVs, AttVal). % Атрибут з найкращою оцінкою

best( [ AttVal/\_], AttVal).

best( [ AV0/S0, AV1/S1 | AVSlist], AttVal):-

S1 > S0, !, % Атрибут AV1 має кращу оцінку, ніж AV0

best( [AV1/S1 | AVSlist], AttVal)

;

best( [AV0/S0 | AVSlist], AttVal).

%filter( Examples, Condition, Examples1) :

% список Examplesl містить елементи списку Examples, які відповідають

% умові Condition

filter( Examples, Cond, Examples1):-

findall( example( Class, Obj),

( member( example( Class, Obj), Examples), satisfy( Obj, Cond)),

Examples1).

% remove( Examples, Conj, Examples1):

% видалення зі списку Examples тих прикладів, які охоплені умовою Conj,

% й одержання списку Examplesl

remove( [], \_, []).

remove( [example(Class, Obj) | Es], Conj, Es1):-

satisfy(Obj, Conj), !, % Перший приклад відповідає умові Conj

remove( Es, Conj, Es1). % Видалити його

remove( [E | Es], Conj, [E | Es1]):- % Залишити перший приклад у списку

remove( Es, Conj, Es1).

satisfy(Object, Conj) :-

not( (member( Att = Val, Conj) ,

member( Att = ValX, Object) ,

ValX \== Val) ).

score( Examples, Class, AttVal, Score) :-

candidate( Examples, Class, AttVal), % Підходяще значення атрибута

filter( Examples, [ AttVal], Examples1), % Приклади в списку Examplesl

%відповідають умові Att = Val

length(Examples1, N1), % Довжина списку

count\_pos( Examples1, Class, NPos1), % Кількість позитивних прикладів

NPos1 > 0, % Щонайменше один позитивний

% приклад відповідає значенню AttVal

Score is 2 \* NPos1 - N1.

candidate(Examples, Class, Att = Val) :-

attribute( Att, Values), % Атрибут

member( Val, Values), % Значення

suitable( Att = Val, Examples, Class).

suitable( AttVal, Examples, Class):-

% Щонайменше один негативний приклад не повинен відповідати знач. AttVal

member( example( ClassX, ObjX), Examples),

ClassX \== Class, % Негативний приклад, що

not( satisfy( ObjX, [ AttVal])), !. % не відповідає значенню AttVal

% count\_pos( Examples, Class, N):

% N - кількість позитивних прикладів класу Class

count\_pos( [], \_, 0).

count\_pos( [example( ClassX,\_ ) | Examples], Class, N):-

count\_pos( Examples, Class, N1),

( ClassX = Class, !, N is N1 + 1; N = N1 ).

writelist( []).

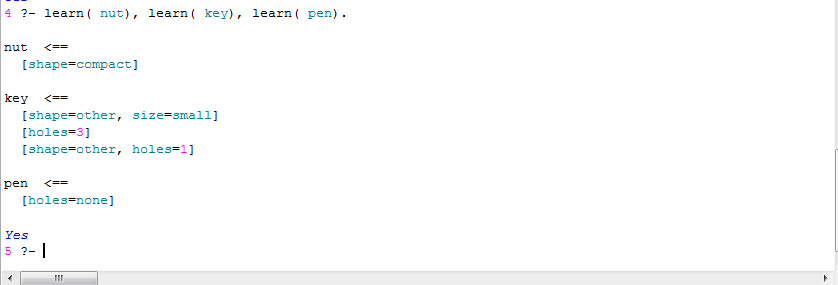
writelist( [X | L]):-

tab( 2), write( X), nl,

writelist( L).

Результат роботи програми





Висновок: на даній лаболаторні роботі було вивчено можливості засобів мови програмування Пролог для створення систем машинного навчання. Також було створено програму логічного виводу правил виводу БД прикладів для навчання були взяті згідно варіанту.