Sztuczna Inteligencja – Projekt

Temat: Algorytm LEM2
Liczba osób realizujących projekt: 2

1. Zaimplementować algorytm $LEM2$.
2. Zaimplementować klasyfikator $Classifier$.
3. Za pomocą algorytmu $LEM2$ wygenerować dla każdej klasy decyzyjnej reguły decyzyjne z 2/3 losowo wybranych obiektów zbioru *iris.tab*.
4. Za pomocą klasyfikatora $Classifier$ dokonać klasyfikacji pozostałych obiektów zbioru *iris.tab*.

Ad 1.

1. Zaimplementować funkcję $Satisfied$ wyznaczającą $[t] \cap G$ dla dowolnego warunku t i dowolnego zbioru G obiektów.
2. Zaimplementować funkcję $SatisfiedSet$ wyznaczającą zbiór $T_G = \{t : [t] \cap G \neq \emptyset\}$.
3. Zaimplementować funkcję $ChooseCondition$, która wybierze najlepszy warunek do dodania do reguły.
4. Zaimplementować funkcję $DropCondition$ usuwającą zbędny warunek z reguły.
5. Zaimplementować funkcję $DropRule$ usuwającą zbędną regułą ze zbioru reguł.

Ad 2.
Klasyfikator $Classifier$ ma przypisać każdy obiekt do klasy tej reguły, którą obiekt spełnia. Jeżeli występuje konflikt klas, tzn. obiekt spełnia reguły dotyczące różnych klas, to jest on przypisywany do klasy tych reguł, których spełnia więcej. W przypadku, gdy obiekt spełnia tyle samo reguł każdej ze spornych klas, to nie jest on przypisywany do żadnej klasy.

Charakterystyka algorytmu LEM2

Oznaczenia i definicje:

- X – niepusta dolna lub górna aproksymacja klasy decyzyjnej;
- $t = (a, v)$ – warunek, gdzie a – atrybut warunkowy, v – wartość przyjmowana przez a;
- $[t]$ – blok, tj. zbiór obiektów spełniających warunek $t = (a, v)$;
- T - zbiór warunków t;
- $[T] = \bigcap_{t \in T}[t]$ - zbiór obiektów spełniających każdy z warunków $t \in T$;
- Zbiór X *zależy* od zbioru T wtedy i tylko wtedy, gdy $\emptyset \neq [T] \subseteq X$;
- T jest *minimalnym kompleksem* zbioru X wtedy i tylko wtedy, gdy X zależy od T i nie istnieje $T' \subset T$ takie, że X zależy od T';
- $\mathcal{T}$ - rodzina zbiorów T;
- $\mathcal{T}$ jest *lokalnym pokryciem* zbioru X wtedy i tylko wtedy, gdy:
  1. Każdy zbiór $T \in \mathcal{T}$ jest minimalnym kompleksem zbioru X;
  2. $\bigcup_{T \in \mathcal{T}}[T] = X$;
  3. $\mathcal{T}$ jest minimalny, tj. składa się z najmniejszej możliwej liczby zbiorów T.

Opis działania algorytmu

Algorytm LEM2 na wejściu otrzymuje aproksymację rozpatrywanej klasy decyzyjnej. Jeżeli jest to dolna aproksymacja, to są generowane reguły pewne, jeżeli górna, to możliwe (Zamiast aproksymacji można rozpatrywać obszar brzegowy, wtedy generowane są reguły przybliżone). Algorytm przy generowaniu każdej reguły bierze pod uwagę tylko te warunki, które są spełnione przynajmniej przez jeden obiekt z rozpatrywanego zbioru (tj. $T_G := \{t : [t] \cap G \neq \emptyset\}$). W każdym kroku generowania reguły (zbiór T reprezentuje regułę) wybierany jest taki warunek, który jest spełniany przez największą liczbę obiektów (tj. $|[t] \cap G|$ jest maksymalne). Jeżeli jest więcej takich warunków spełniających podane kryterium, to spośród nich wybierany jest ten, który jest spełniony przez najmniejszą liczbę wszystkich obiektów z U (tj. $|[t]|$ jest minimalne). W tym przypadku jest to równoważne temu, że taki warunek jest spełniony przez najmniejszą liczbę obiektów spoza zbioru G. Reguła jest tworzona (tj. $T := T \cup \{t\}$), dopóki nie jest dodany ani jeden warunek do reguły (tj. $T = \emptyset$) lub te warunki które już są dodane do reguły są spełniane nie tylko przez obiekty z rozpatrywanej aproksymacji (tj. $[T] \not\subseteq X$). Po wygenerowaniu każdej reguły, następuje jej przycinanie poprzez usunięcie

Algorytm 1: LEM2

Data: zbiór X;
Result: pojedyncze lokalne pokrycie $\mathcal{T}$ zbioru X;
begin
  $G := X; \mathcal{T} := \emptyset$;
  while $G \neq \emptyset$ **do**
    $T := \emptyset$; $T_G := \{t : [t] \cap G \neq \emptyset\}$;
    while $T = \emptyset$ **or** $[T] \not\subseteq X$ **do**
      wybierz $t \in T_G$ takie, że $|[t] \cap G|$ jest maksymalne; jeżeli jest więcej niż jedno t spełniające warunek, to wybierz spośród nich pierwsze znalezione t takie, że $|[t]|$ jest minimalne;
      $T := T \cup \{t\}$; $G := [t] \cap G$; $T_G = \{t : [t] \cap G \neq \emptyset\} \setminus T$;
    foreach $t \in T$ **do**
      if $[T \setminus \{t\}] \subseteq X$ **then** $T := T \setminus \{t\}$;
    $\mathcal{T} := \mathcal{T} \cup \{T\}$; $G := X \setminus \bigcup_{T \in \mathcal{T}} [T]$;
  foreach $T \in \mathcal{T}$ **do**
    if $\bigcup_{T' \in \mathcal{T} \setminus \{T\}} [T'] = X$ **then** $\mathcal{T} := \mathcal{T} \setminus \{T\}$;
end

zbędnych warunków (tj. **if** $[T \setminus \{t\}] \subseteq X$ **then** $T := T \setminus \{t\}$). Przy generowaniu kolejnej reguły, rozpatrywane są tylko te obiekty z danej aproksymacji, które nie spełniają wygenerowanych do tej pory reguł (tj. $G := X \setminus \bigcup_{T \in \mathcal{T}} [T]$), czyli nie spełniają reguł znajdujących się w zbiorze $\mathcal{T}$. Po wygenerowaniu wszystkich reguł, tzn. w sytuacji gdy nie ma już obiektów z rozpatrywanej aproksymacji niespełniających którejkolwiek z reguł (tj. $G \neq \emptyset$ nie jest spełnione), następuje przycinanie zbioru reguł poprzez usuwanie zbędnych reguł (tj. **if** $\bigcup_{T' \in \mathcal{T} \setminus \{T\}} [T'] = X$ **then** $\mathcal{T} := \mathcal{T} \setminus \{T\}$).

Przykład
Dana jest tablica decyzyjna $DT = (U, A \cup \{d\})$:

obiekt	*temperatura*	*bol_glowy*	*oslabienie*	*nudnosci*	*grypa*
1	b._wysoka	tak	tak	nie	tak
2	wysoka	tak	nie	tak	tak
3	normalna	nie	nie	nie	nie
4	normalna	tak	tak	tak	tak
5	wysoka	nie	tak	nie	tak
6	wysoka	nie	nie	nie	nie
7	normalna	nie	tak	nie	nie

gdzie $U = \{1, \ldots, 7\}$,
$A = \{temperatura, bol\_glowy, oslabienie, nudnosci\}, d = grypa$.

Wyznaczyć reguły decyzyjne stosując algorytm LEM2.

Niech $X = \{1, 2, 4, 5\}$.
$G := X; \mathcal{T} := \emptyset;$
Pętla **while** $G \neq \emptyset$ (krok 1):
$T := \emptyset; T_G := \{(temperatura, b.\_wysoka), (temperatura, wysoka),$
$(temperatura, normalna), (bol\_glowy, tak), (bol\_glowy, nie), (oslabienie, tak),$
$(oslabienie, nie), (nudnosci, tak), (nudnosci, nie)\};$
Pętla **while** $T = \emptyset$ **or** $[T] \not\subseteq X$ (krok 1):
Dla warunków (bol_glowy, tak) i $(oslabienie, tak)$ wyrażenie $|[t] \cap G|$ przyjmuje maksymalną wartość, tj. 3. Wybierany jest warunek (bol_glowy, tak), gdyż $|(bol\_glowy, tak)| = 3 < |(oslabienie, tak)| = 5$.
$T := \{(bol\_glowy, tak)\}; G := \{1, 2, 4\}; T_G = \{(temperatura, b.\_wysoka),$
$(temperatura, wysoka), (temperatura, normalna, (bol\_glowy, nie),$
$(oslabienie, tak), (oslabienie, nie), (nudnosci, tak), (nudnosci, nie)\}.$
Otrzymujemy $T :\neq \emptyset$ **and** $[T] \subseteq X$, więc wychodzimy z pętli wewnętrznej.
$\mathcal{T} := \{\{(bol\_glowy, tak)\}\}; G := \{1, 2, 4, 5\} \setminus \{1, 2, 4\} = \{5\}.$
Pętla **while** $G \neq \emptyset$ (krok 2):
$T := \emptyset; T_G := \{(temperatura, wysoka), (bol\_glowy, nie), (oslabienie, tak),$
$(nudnosci, nie)\};$
Pętla **while** $T = \emptyset$ **or** $[T] \not\subseteq X$ (krok 1):
Wybierany jest warunek $(temperatura, wysoka)$, otrzymujemy
$T := \{(temperatura, wysoka)\}; G := \{5\}; T_G = \{(bol\_glowy, nie),$
$(oslabienie, tak), (nudnosci, nie)\}.$
Ponieważ $[T] = \{2, 5, 6\} \not\subseteq X$, więc
Pętla **while** $T = \emptyset$ **or** $[T] \not\subseteq X$ (krok 2):
Wybierany jest warunek (bol_glowy, nie), otrzymujemy
$T := \{(temperatura, wysoka), (bol\_glowy, nie)\}; G := \{5\};$
$T_G = \{(oslabienie, tak), (nudnosci, nie)\}$
Ponieważ $[T] = \{5, 6\} \not\subseteq X$, więc
Pętla **while** $T = \emptyset$ **or** $[T] \not\subseteq X$ (krok 3):
Wybierany jest warunek $(oslabienie, tak)$, otrzymujemy
$T := \{(temperatura, wysoka), (bol\_glowy, nie), (oslabienie, tak)\}; G := \{5\};$
$T_G = \{(nudnosci, nie)\}$
Ponieważ $[T] = \{5\} \subseteq X$, więc wychodzimy z pętli wewnętrznej.
Przycinając regułę reprezentowaną przez zbiór T (pętla **foreach** $t \in T$) otrzymujemy
$T = \{(temperatura, wysoka), (oslabienie, tak)\}; \mathcal{T} := \{\{(bol\_glowy, tak)\},$
$\{(temperatura, wysoka), (oslabienie, tak)\}\}; G := \{5\} \setminus \{5\} = \emptyset,$ więc wychodzimy z pętli zewnętrznej.

Na podstawie otrzymanego lokalnego pokrycia konstruowane są reguły: $(bol\_glowy, tak) \rightarrow (grypa, tak), (temperatura, wysoka) \wedge (oslabienie, tak) \rightarrow (grypa, tak)$. Zbiór reguł wygenerowanych przez algorytm LEM2 składa się z następujących reguł (po dwukropku podane pokrycie reguły):

$(bol\_glowy, tak) \rightarrow (grypa, tak) : 1, 2, 4,$
$(temperatura, wysoka) \wedge (oslabienie, tak) \rightarrow (grypa, tak) : 5,$
$(temperatura, normalna) \wedge (bol\_glowy, nie) \rightarrow (grypa, nie) : 3, 7,$
$(bol\_glowy, nie) \wedge (oslabienie, nie) \rightarrow (grypa, nie) : 3, 6.$