Programación para la Inteligencia Artificial
Tarea 2

Arturo Márquez Flores
Maestría en Inteligencia Artificial
Universidad Veracruzana
CIIA – Centro de Investigación en Inteligencia Artificial
Sebastián Camacho No 5, Xalapa, Ver., México 91000
arturomf94@gmail.com
https://github.com/arturomf94/pia-mia

16 de Octubre del 2018

1. Pregunta 1

El código para responder la primera pregunta es el siguiente, en los Cuadros 1 y 2.

```
%%% Las dos primeras clausulas se encargan de
%%% definir una relacion entre los animales
%%% que represente la imposibilidad de dejarlos
%%% a ambos en el mismo lugar.

odia(perro, gato).
odia(gato, hamster).
```

Cuadro 1: Primera Parte de la Implementación en Prolog para la Pregunta 1

```
1
2       %%% La clausula s define, por otro lado,
3       %%% las conexiones entre diferentes estados.
4       %%% Estos estados son de la siguiente forma:
5       %%% [Casa1, Transporte, Casa2].
6
7       s([[], [X], Y],[[],[],[X|Y]]) :- !.
8
9       s([Casa1, [], Casa2], [Casa1_Nueva, [X], Casa2]) :-
10          member(X, Casa1),
11          mover(X, Casa1, Casa1_Nueva),
12          not(estado_invalido(Casa1_Nueva, Casa2)).
13
14      s([Casa1, [X], Casa2], [Casa1, [], [X|Casa2]]) :-
15          not(estado_invalido(Casa1,[X|Casa2])).
16
17      s([Casa1, [], Casa2], [Casa1, [X], Casa2_Nueva]) :-
18          member(X, Casa2),
19          mover(X, Casa2, Casa2_Nueva),
20          not(estado_invalido(Casa1, Casa2_Nueva)).
21
22      s([Casa1, [X], Casa2], [[X|Casa1], [], Casa2]) :-
23          not(estado_invalido([X|Casa1],Casa2)).
24
25      s([Casa1, [X], Casa2],[Casa1, [Y], [X|Casa2_Nueva]]) :-
26          member(Y, Casa2),
27          mover(Y, Casa2, Casa2_Nueva),
28          not(estado_invalido(Casa1,[X|Casa2_Nueva])).
29
30      s([Casa1, [X], Casa2],[[X|Casa1_Nueva], [Y], Casa2]) :-
31          member(Y, Casa1),
32          mover(Y, Casa1, Casa1_Nueva),
33          not(estado_invalido([X|Casa1_Nueva],Casa2)).
34
35      %%% Definimos la meta como cualquier estado
36      %%% [[], [], P], donde P es cualquier permutacion
37      %%% se los tres animales.
38
39      meta([[],[],P]) :-
40          permutation([perro,gato,hamster],P).
41
42      %%% mover es una clausula adicional que
43      %%% elimina un elemento de una lista.
44
45      mover(X, [X|Tail], Tail).
46      mover(X, [Y|Tail], [Y|NewTail]) :-
47          mover(X, Tail, NewTail).
48
49      %%% La clausula estado_invalido
50      %%% verifica que no existan conflictos
51      %%% de la clausla odia en el mismo lugar.
52
53      estado_invalido(C1, C2) :-
54          (estado_invalido_aux(C1);
55          estado_invalido_aux(C2)).
56
57      estado_invalido_aux([Top|Tail]) :-
58          (odia(Top, Other); odia(Other, Top)),
59          member(Other, Tail),
60          !.
```

Cuadro 2: Segunda Parte de la Implementación en Prolog para la Pregunta 1

Utilizando el método de búsqueda en profundidad visto en clase, [1] podemos ver en el Cuadro 3 la solución de la búsqueda.

```
1
2       %%% ?- solucion2([[perro, gato, hamster], [], []], Sol).
3       %%% Sol = [[[], [], [gato, hamster, perro]], [[], [gato],
4       %%% [hamster, perro]], [[gato], [], [hamster, perro]], [[gato],
5       %%% [hamster], [perro]], [[hamster], [gato], [perro]], [[hamster],
6       %%% [perro], [...]], [[perro|...], []|...], [[...|...]|...],
7       %%% [...|...]] [write]
8       %%% Sol = [[[], [], [gato, hamster, perro]], [[], [gato],
9       %%% [hamster, perro]], [[gato], [], [hamster, perro]], [[gato],
10      %%% [hamster], [perro]], [[hamster], [gato], [perro]], [[hamster],
11      %%% [perro], [gato]], [[perro, hamster], [], [gato]], [[perro,
12      %%% hamster], [gato], []], [[perro, gato, hamster], [], []] .
```

Cuadro 3: Ejecución de Código para la Pregunta 1

[1] Particularmente, utilizando la cláusula solucion2 que busca evitando ciclos en el archivo *busquedaProfundidad.pl*

2. Pregunta 2

Para responder la pregunta 2, se calculó el costo de viajar de una ciudad a otra tomando el precio de la gasolina actual, los kilometros de ruta en [1] y los costos de caseta en [2]. Los Cuadros 4 y 5 muestran la implementación en Prolog del mapa construido.

```
1
2       %%% Las primeras clausulas incluyen
3       %%% el costo de viaje de una ciudad
4       %%% a otra, conformado por costo en
5       %%% casetas y gasolina.
6
7       costo(tuxpan, poza_rica, 146).
8       costo(poza_rica, xalapa, 413).
9       costo(xalapa, veracruz, 314).
10      costo(xalapa, cordoba, 563).
11      costo(cordoba, orizaba, 77).
12      costo(cordoba, veracruz, 411).
13      costo(cordoba, minatitlan, 1050).
14      costo(veracruz, boca_del_rio, 19).
15      costo(boca_del_rio, minatitlan, 1048).
16      costo(minatitlan, coatzacoalcos, 40).
17
18      %%% La clausula s se define para
19      %%% tener la relacion bidireccional
20      %%% entre ciudades.
21
22      s(X, Y, C) :- costo(X,Y, C).
23      s(X, Y, C) :- costo(Y,X, C).
```

Cuadro 4: Implementación en Prolog para la Pregunta 3

```
1
2       %%% La clausula de distancia_aux
3       %%% define la distancia entre ciudades
4       %%% conectadas.
5
6       distancia_aux(tuxpan, poza_rica, 56).
7       distancia_aux(poza_rica, xalapa, 221).
8       distancia_aux(xalapa, veracruz, 107).
9       distancia_aux(xalapa, cordoba, 140).
10      distancia_aux(cordoba, orizaba, 25).
11      distancia_aux(cordoba, veracruz, 110).
12      distancia_aux(cordoba, minatitlan, 286).
13      distancia_aux(veracruz, boca_del_rio, 10).
14      distancia_aux(boca_del_rio, minatitlan, 288).
15      distancia_aux(minatitlan, coatzacoalcos, 21).
16
17      %%% distancia generaliza distancia_aux
18      %%% para que sea una relacion bidireccional.
19
20      distancia(X, Y, D) :- distancia_aux(X, Y, D).
21      distancia(X, Y, D) :- distancia_aux(Y, X, D).
22
23      %%% La clausula h define la heuristica
24      %%% utilizada en la busqueda.
25      %%% h calcula la suma de distancias entre
26      %%% cualquier par de ciudades.
27
28      h(X, Y, D) :-
29          h_aux(X, Y, [X], D), !.
30
31      h_aux(X, Y, _, D) :- distancia(X, Y, D), !.
32      h_aux(X, Y, _, D) :- distancia(Y, X, D), !.
33      h_aux(X, Z, V, D) :-
34          distancia(X, Y, D1),
35          \+ member(Y, V),
36          h_aux(Y, Z, [Y|V], D2),
37          D is D1 + D2.
```

Cuadro 5: Implementación en Prolog para la Pregunta 2

Utilizando el método de búsqueda primero el mejor visto en clase, implementado en el archivo *busquedaPrimeroMejor.pl*, obtenemos los resultados que se muestran en el Cuadro 6.

```
1
2       %%% ?- primeroMejor(poza_rica, minatitlan, Sol).
3       %%% Sol = [minatitlan, boca_del_rio, veracruz, xalapa, poza_rica] ;
4       %%% Sol = [minatitlan, cordoba, xalapa, poza_rica] ;
5       %%% Sol = [minatitlan, cordoba, veracruz, xalapa, poza_rica] ;
6       %%% Sol = [minatitlan, boca_del_rio, veracruz, cordoba, xalapa, poza_rica] ;
7       %%% false.
```

Cuadro 6: Resultado de una Ejecución para la Pregunta 2

Para poder hacer esta ejecución posible, se hicieron algunos cambios en el archivo *busquedaPrimeroMejor.pl*. Las clausulas modificadas pueden verse en 7.

```
1
2       %%% La primera clausula modificada es primeroMejor
3       %%% que ahora tiene 3 argumentos. Esta clausula
4       %%% define la meta y hace una llamada a primeroMejorAux
5       %%% que es la version anterior de primeroMejor.
6
7       primeroMejor(Inicio, Fin, Sol) :-
8           retractall(meta(_)),
9           assertz(meta(Fin)),
10          primeroMejorAux(Inicio, Sol).
11
12      %%% El segundo cambio que se hizo fue en la
13      %%% clausula listaSucs, donde aniadimos un
14      %%% argumento a la clausula h para que
15      %%% funcionara con las definiciones del mapa.
16
17      listaSucs(G0, [N/C|NCs], As)  :-
18        meta(X),
19        G is G0 + C,
20        h(N, X, H),                              % Heuristica h(N)
21        F is G + H,
22        listaSucs(G0, NCs, As1),
23        insertar(l(N,F/G), As1, As).
```

Cuadro 7: Clausulas modificadas en *busquedaPrimeroMejor.pl*

3. Pregunta 3

Como se menciona en [3], en la sección 5, una de las limitaciones del modelo del ID3 presentado es el hecho de que el algoritmo no puede procesar datos que sean conflictivos; es decir, que contengan ruido.

Una de las soluciones planteadas por el autor es la posibilidad de generalizar la noción de clase como un número entre 0 y 1. Con esto, podríamos interpretar el resultado como la probabilidad de que una observación con los atributos correspondientes pertenezca a dicha clase.

En esta sección se muestran las modificaciones pertinentes al archivo *id3.pl* para poder lograr este cambio en la noción de clase. Para tener un punto de referencia fijo, utilizamos los mismos ejemplos vistos en clase y tratados en [3]. En la Figura 1 observamos estos ejemplos.

1	cielo	temperatura	humedad	viento	jugarTenis
2	soleado	alta	alta	debil	no
3	soleado	alta	alta	fuerte	no
4	nublado	alta	alta	debil	si
5	lluvioso	templada	alta	debil	si
6	lluvioso	fresca	normal	debil	si
7	lluvioso	fresca	normal	fuerte	no
8	nublado	fresca	normal	fuerte	si
9	soleado	templada	alta	debil	no
10	soleado	fresca	normal	debil	si
11	lluvioso	templada	normal	debil	si
12	soleado	templada	normal	fuerte	si
13	nublado	templada	alta	fuerte	si
14	nublado	alta	normal	debil	si
15	lluvioso	templada	alta	fuerte	no

Figura 1: Ejemplos Originales

Si utilizamos el algoritmo original ID3 con estos datos, obtenemos un arbol decisión sin errores, ya que estos ejemplos no tienen ruido. Esto podemos observarlo en 8.

```
1
2       %%% Con que archivo CSV desea trabajar: 'tenis.csv'.
3       %%% 14 ejemplos de entrenamiento cargados.
4       %%% Tiempo de induccion:
5       %%% % 5,260 inferences, 0.001 CPU in 0.001 seconds (100% CPU, 5532271 Lips)
6       %%%
7       %%%   cielo=lluvioso
8       %%%     viento=fuerte => [no/2]
9       %%%     viento=debil => [si/3]
10      %%%   cielo=nublado => [si/4]
11      %%%   cielo=soleado
12      %%%     humedad=normal => [si/2]
13      %%%     humedad=alta => [no/3]
14      %%% true.
```

Cuadro 8: ID3 con Ejemplos Originales

Sin embargo, si utilizamos este mismo algoritmo para intentar procesar los ejemplos en 2, que, siguiendo el ejemplo en [3], modifican la observación 1 en la columna *cielo*, entonces, debido al conflicto entre la observación 1 y la observación 3, se produce un error. Esto lo podemos observar en la ejecución en el Cuadro 9.

	cielo	temperatura	humedad	viento	jugarTenis
1					
2	nublado	alta	alta	debil	no
3	soleado	alta	alta	fuerte	no
4	nublado	alta	alta	debil	si
5	lluvioso	templada	alta	debil	si
6	lluvioso	fresca	normal	debil	si
7	lluvioso	fresca	normal	fuerte	no
8	nublado	fresca	normal	fuerte	si
9	soleado	templada	alta	debil	no
10	soleado	fresca	normal	debil	si
11	lluvioso	templada	normal	debil	si
12	soleado	templada	normal	fuerte	si
13	nublado	templada	alta	fuerte	si
14	nublado	alta	normal	debil	si
15	lluvioso	templada	alta	fuerte	no

Figura 2: Ejemplos Modificados

```
1
2       %%% Con que archivo CSV desea trabajar: 'altered_tenis.csv'.
3       %%% 14 ejemplos de entrenamiento cargados.
4       %%% Tiempo de induccion:Datos inconsistentes: no es posible construir
5       %%% particion de [1,3] en el nodo temperatura=alta
6       %%% % 7,877 inferences, 0.002 CPU in 0.002 seconds (100% CPU, 3195259 Lips)
7       %%%
8       %%%   humedad=normal
9       %%%     viento=fuerte
10      %%%       cielo=soleado => [si/1]
11      %%%       cielo=nublado => [si/1]
12      %%%       cielo=lluvioso => [no/1]
13      %%%     viento=debil => [si/4]
14      %%%   humedad=alta
15      %%%     cielo=lluvioso
16      %%%       viento=fuerte => [no/1]
17      %%%       viento=debil => [si/1]
18      %%%     cielo=soleado => [no/2]
19      %%%     cielo=nublado
20      %%%       viento=fuerte => [si/1]
21      %%%       viento=debil
22      %%%         temperatura=alta
23      %%% true.
```

Cuadro 9: ID3 con Ejemplos Modificados

Para poder lidear con esta limitación se hicieron las modificaciones en las clausulas mostradas en 10.

```
%%% La primera modificacion realizada fue
%%% En la clausula inducir que trata el
%%% caso 4, en el cual no era posible
%%% procesar datos inconsistentes.
%%% En lugar de mandar un error, la
%%% clausula ahora cuenta las observaciones
%%% con la clase 'si' y divide ese numero
%%% entre el numero de ejemplos del conjunto.

inducir(Ejs,Padre,_,_) :- !,
    nodo(Padre,_Test,_),
    findall(Ej,(member(Ej,Ejs),ejemplo(Ej,si,_)),EjsEnClase),
    length(EjsEnClase,NumEjsEnClase), !,
    length(Ejs,NumEjs), !,
    Prop is NumEjsEnClase / NumEjs,
    assertz(nodo(hoja,[si/Prop],Padre)).

%%% Se hizo una segunda modificacion para
%%% expresar todas las clases como probabilidades
%%% En este caso, queremos que si un conjunto
%%% tiene solo ejemplos de una misma clase
%%% entonces la probabilidad sea 1.

cuentaClases([Clase|Clases],Ejs,[Clase/Test|RestoCuentas]) :-
    % Extrae los ejemplos con clase Clase en la lista Cuentas
    findall(Ej,(member(Ej,Ejs),ejemplo(Ej,Clase,_)),EjsEnClase),
    % Computa cuantos ejemplos hay en la Clase
    length(EjsEnClase,NumEjsEnClase), !,
    Test is NumEjsEnClase / NumEjsEnClase,
    % Cuentas para el resto de los valores de la clase
    cuentaClases(Clases,Ejs,RestoCuentas).
```

Cuadro 10: Modificaciones el ID3

Utilizando este algoritmo modificado ahora podemos procesar los ejemplos de 2. En el Cuadro 11 podemos ver el resultado.

```
%%% Con que archivo CSV desea trabajar: 'altered_tenis.csv'.
%%% 14 ejemplos de entrenamiento cargados.
%%% Tiempo de induccion:
%%% % 7,976 inferences, 0.001 CPU in 0.001 seconds (100% CPU, 7190788 Lips)
%%%
%%%   humedad=normal
%%%     viento=fuerte
%%%       cielo=soleado => [si/1]
%%%       cielo=nublado => [si/1]
%%%       cielo=lluvioso => [no/1]
%%%     viento=debil => [si/1]
%%%   humedad=alta
%%%     cielo=lluvioso
%%%       viento=fuerte => [no/1]
%%%       viento=debil => [si/1]
%%%     cielo=soleado => [no/1]
%%%     cielo=nublado
%%%       viento=fuerte => [si/1]
%%%       viento=debil
%%%         temperatura=alta => [si/0.5]
%%% true.
```

Cuadro 11: Modificaciones el ID3

Podemos observar que ahora el árbol de decisión expresa todas las clases con un número del 0 al 1. En particular, el caso conflictivo ahora se expresa como un *sí* con probabilidad .05.

Referencias

[1] Google Maps (n.d.), consultado el 26 de octubre, 2018.

[2] Secretaría de Comunicación y Transportes (n.d.), consultado el 26 de octubre, 2018.

[3] Quinlan, *Induction of Decision Trees*, Machine Learning 1: 81-106, 1986;