

PROGRAMACIÓN PARA LA INTELIGENCIA ARTIFICIAL

Proyecto Final

David Martínez Galicia

David Hernández Enriquez

José Alberto López López

Arturo Márquez Flores

8 Enero 2019

Maestría en Inteligencia Artificial

Universidad Veracruzana

CIIA – Centro de Investigación en Inteligencia Artificial

Sebastián Camacho No 5, Xalapa, Ver., México 91000

<https://github.com/arturomf94/pia-mia/tree/master/proyecto-final>

- Validación Cruzada
- Clasificación por Votación
- Traducción a Prolog

VALIDACIÓN CRUZADA

VALIDACIÓN CRUZADA

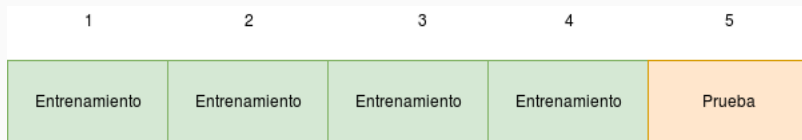


Figura: Validación cruzada con 5 iteraciones

MODIFICACIONES PARA LA VALIDACIÓN CRUZADA

```
1      ;;; Modificaciones
2      (defvar *classified-int* ())
3      (defvar *c-classified-int* ())
4
5      (defun gui-cross-validation (data interface)
6        (declare (ignore data))
7        (progn
8          ;; Toma K de la interfaz
9          (setf k (text-input-pane-text (k-pane interface)))
10         (setf *classified-int* '() *c-classified-int* '()
11              *k-validation-trees* '() *classify-on* t)
12         ;; Lanza la funcion con K
13         (cross-validation (parse-integer k))
14         ;; Escribe el mejor arbol
15         (traducir *best-tree*)
16         ;; Calcula y define la eficiencia
17         (setf (text-input-pane-text (e-pane interface))
18              (princ-to-string (/ (apply #'* *c-classified-int*)
19                                  (apply #'* *classified-int*))))
19         (setf (text-input-pane-text (e1-pane interface))
20              (princ-to-string (calculate-voting-accuracy *k-validation-trees*)))
21         (setf (text-input-pane-text (e2-pane interface))
22              (princ-to-string (calculate-best-tree-accuracy *best-tree*))))))
23
```

Cuadro: Modificaciones al Código en `cl-id3-gui.lisp`

Cross Validation	
K value	5
Average Accuracy	8/25

Figura: Validación cruzada con 5 iteraciones

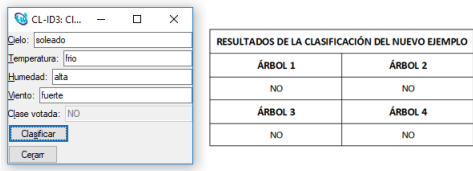
CLASIFICACIÓN POR VOTACIÓN

CLASIFICACIÓN DE UN NUEVO EJEMPLO

```
1
2 (defun classifyn-gui (data interface)
3   (declare (ignore data))
4   (progn
5     (setf nsi 0 nno 0)
6     (setf new-lst (list
7       (read-from-string (text-input-pane-text (ex1-pane interface)))
8       (read-from-string (text-input-pane-text (ex2-pane interface)))
9       (read-from-string (text-input-pane-text (ex3-pane interface)))
10      (read-from-string (text-input-pane-text (ex4-pane interface))))))
11     (setf new-l (loop for arbol in *k-validation-trees*
12                      collect (classify-new-instance new-lst arbol)))
13     (setf nsi (length (loop for class in new-l
14                            when (equal (string class) "SI")
15                            collect class)))
16     (setf nno (- (length new-l) nsi))
17     (if (> nsi nno)
18       (setf (text-input-pane-text (most-voted-class interface))
19         (princ-to-string 'si))
20       (setf (text-input-pane-text (most-voted-class interface))
21         (princ-to-string 'no))))
22
```

Cuadro: Modificaciones al Código en `cl-id3-gui.lisp`

CLASIFICACIÓN DE UN EJEMPLO NUEVO



The image shows a software window titled "CL-ID3: CL..." with a standard Windows-style title bar. Inside the window, there are five input fields with labels and values: "Cielo: soleado", "Temperatura: frio", "Humedad: alta", "Viento: fuerte", and "Clase votada: NO". Below these fields are two buttons: "Clasificar" (highlighted with a red dashed border) and "Cerrar".

To the right of the window is a table titled "RESULTADOS DE LA CLASIFICACIÓN DEL NUEVO EJEMPLO". The table has two columns and four rows. The first two rows are for "ÁRBOL 1" and "ÁRBOL 2", both showing "NO". The next two rows are for "ÁRBOL 3" and "ÁRBOL 4", both also showing "NO".

RESULTADOS DE LA CLASIFICACIÓN DEL NUEVO EJEMPLO	
ÁRBOL 1	ÁRBOL 2
NO	NO
ÁRBOL 3	ÁRBOL 4
NO	NO

Figura: Clasificación de un ejemplo nuevo

CLASIFICACIÓN POR VOTACIÓN

ÁRBOL 1	ÁRBOL 2	ÁRBOL 3	ÁRBOL 4
HUMEDAD - NORMAL CIELO - NUBLADO -> SI - LLUVIA TEMPERATURA - FRIO -> NO - TEMPLADO -> SI - SOLEADO -> SI - ALTA CIELO - LLUVIA -> NO - NUBLADO -> SI - SOLEADO -> NO	HUMEDAD - NORMAL -> SI - ALTA CIELO - SOLEADO -> NO - LLUVIA VIENTO - DEBIL -> SI - FUERTE -> NO - NUBLADO -> SI	CIELO - SOLEADO HUMEDAD - NORMAL -> SI - ALTA -> NO - LLUVIA VIENTO - FUERTE -> NO - DEBIL -> SI - NUBLADO -> SI MEJOR ÁRBOL	HUMEDAD - ALTA CIELO - SOLEADO -> NO - LLUVIA -> NO - NUBLADO -> SI - NORMAL CIELO - SOLEADO -> SI - LLUVIA VIENTO - DEBIL -> SI - FUERTE -> NO - NUBLADO -> SI
Clasificaciones correctas: 2 Clasificaciones incorrectas: 2	Clasificaciones correctas: 3 Clasificaciones incorrectas: 1	Clasificaciones correctas: 4 Clasificaciones incorrectas: 0	Clasificaciones correctas: 3 Clasificaciones incorrectas: 1

Figura: Árboles Generados

MODIFICACIONES PARA LA CLASIFICACIÓN POR VOTACIÓN

```
1
2 (defun calculate-voting-accuracy (trees)
3   (/ (count-voting-positives trees *examples*) (length *examples*)))
4
5 (defun calculate-best-tree-accuracy (best-tree)
6   (/ (count-positives best-tree *examples*) (length *examples*)))
7
8 (defun count-voting-positives (tree data)
9   (apply #'+
10    (mapcar #'(lambda (e)
11                (if (eql (classify-new-instance-votacion e tree)
12                        (get-value *target* e))
13                    1 0)) data)))
```

Cuadro: Modificaciones al Código en `cl-id3-cross-validation.lisp`

CLASIFICACIÓN DE UN EJEMPLO NUEVO

Voting Accuracy	13/14
Best-Tree Accuracy	1

Figura: Clasificación de un ejemplo nuevo

TRADUCCIÓN A PROLOG

La función principal para traducir el mejor árbol de **cross-validation** es **traducir**.

```
1 (defvar *classified-int* ())
2 (defvar *c-classified-int* ())
3
4 (defun gui-cross-validation (data interface)
5   (declare (ignore data))
6   (progn
7     ;; Toma K de la interfaz
8     (setf k (text-input-pane-text (k-pane interface)))
9     (setf *classified-int* '() *c-classified-int* '()
10          *k-validation-trees* '() *classify-on* t)
11     ;; Lanza la funcion con K
12     (cross-validation (parse-integer k))
13     ;; Traduce a Prolog el mejor arbol
14     (traducir *best-tree*)
15     ;; Calcula y define la eficiencia
16     (setf (text-input-pane-text (e-pane interface))
17          (princ-to-string (/ (apply #'* *c-classified-int*)
18                              (apply #'* *classified-int*))))
19     (setf (text-input-pane-text (e1-pane interface))
20          (princ-to-string (calculate-voting-accuracy *k-validation-trees*)))
21     (setf (text-input-pane-text (e2-pane interface))
22          (princ-to-string (calculate-best-tree-accuracy *best-tree*))))))
```

FUNCIÓN traducir

```
1 (defun traducir2 (arbol padre etiqueta)
2 (let ((*print-case* :downcase))
3 (if (listp arbol)
4     (loop for i in (cdr (atributo-valores arbol)) do
5         (progn
6             ;Escribir los nodos del árbol de Prolog.
7             (if (eq padre 0)
8                 (escribir (setf etiqueta (+ etiqueta 1))
9                     (car (atributo-valores arbol)) i 'raiz)
10                (escribir (setf etiqueta (+ etiqueta 1))
11                    (car (atributo-valores arbol)) i padre))
12             (loop for j in (cdr arbol) do
13                 (if (equal i (car j))
14                     (if (> (length j) 1)
15                         (setf etiqueta (traducir2 (cadr j) etiqueta etiqueta))
16                         (setf etiqueta (traducir2 (car j) etiqueta etiqueta))))))
17         finally (return etiqueta))
18 (progn
19     ;Si el argumento es de la forma SI/NO escribir un nodo hoja.
20     (escribir 'a arbol 'c etiqueta)
21     etiqueta))))
```

FUNCIÓN atributo-valores

La función **atributo-valores** recibe un árbol escrito en Lisp y obtiene una lista con el atributo de la raíz del árbol y sus correspondientes valores. Por ejemplo si recibe el siguiente árbol:

```
1 (CIELO
2  (SOLEADO (HUMEDAD (NORMAL SI) (ALTA NO)))
3  (NUBLADO SI)
4  (LLUVIA (VIENTO (FUERTE NO) (DEBIL SI))))
```

Retornaría la siguiente lista:

```
1 (CIELO SOLEADO NUBLADO LLUVIA)
```

Con la cual se generarían los siguientes nodos:

```
1 nodo(1,cielo=soleado,raiz).
2 nodo(4,cielo=nublado,raiz).
3 nodo(5,cielo=lluvia,raiz).
```


TRADUCCIÓN A PROLOG

```
1 (defun traducir2 (arbol padre etiqueta)
2 (let ((*print-case* :downcase))
3 (if (listp arbol)
4     (loop for i in (cdr (atributo-valores arbol)) do
5         (progn
6             ;Escribir los nodos del árbol de Prolog.
7             (if (eq padre 0)
8                 (escribir (setf etiqueta (+ etiqueta 1))
9                     (car (atributo-valores arbol)) i 'raiz)
10                (escribir (setf etiqueta (+ etiqueta 1))
11                    (car (atributo-valores arbol)) i padre))
12             (loop for j in (cdr arbol) do
13                 (if (equal i (car j))
14                     (if (> (length j) 1)
15                         (setf etiqueta (traducir2 (cadr j) etiqueta etiqueta))
16                         (setf etiqueta (traducir2 (car j) etiqueta etiqueta))))))
17         finally (return etiqueta))
18 (progn
19     ;Si el argumento es de la forma SI/NO escribir un nodo hoja.
20     (escribir 'a arbol 'c etiqueta)
21     etiqueta))))
```

FUNCIÓN escribir

La función **escribir** simplemente escribe en un documento **.pl** los nodos del árbol de Lisp traducidos a Prolog.

```
1 (defun escribir (a b c d)
2
3 (with-open-file (str "/home/j6/quicklisp/local-projects/cl-id3/cl-id3/arbol.pl"
4                     :direction :output
5                     :if-exists :append
6                     :if-does-not-exist :create)
7
8   (if (or (equal b 'si) (equal b 'no))
9       (format str "nodo(hoja,[~A/_],~A).~%" b d)
10      (format str "nodo(~A,~A=~A,~A).~%" a b c d))))
```

TRADUCCIÓN A PROLOG

```
1 (defun traducir2 (arbol padre etiqueta)
2 (let ((*print-case* :downcase))
3 (if (listp arbol)
4     (loop for i in (cdr (atributo-valores arbol)) do
5         (progn
6             ;Escribir los nodos del árbol de Prolog.
7             (if (eq padre 0)
8                 (escribir (setf etiqueta (+ etiqueta 1))
9                     (car (atributo-valores arbol)) i 'raiz)
10                 (escribir (setf etiqueta (+ etiqueta 1))
11                     (car (atributo-valores arbol)) i padre))
12             (loop for j in (cdr arbol) do
13                 (if (equal i (car j))
14                     (if (> (length j) 1)
15                         (setf etiqueta (traducir2 (cadr j) etiqueta etiqueta))
16                         (setf etiqueta (traducir2 (car j) etiqueta etiqueta))))))
17         finally (return etiqueta))
18 (progn
19     ;Si el argumento es de la forma SI/NO escribir un nodo hoja.
20     (escribir 'a arbol 'c etiqueta)
21     etiqueta))))
```

Finalmente la función que sirve como interfaz de todas las anteriores y del proceso de escritura del algoritmo en Prolog es la siguiente:

```
1 (defun traducir (arbol)
2
3 (progn
4 (traducir2 arbol 0 0)
5 (with-open-file (str "/home/j6/quicklisp/local-projects/cl-id3/cl-id3/arbol.pl"
6                     :direction :output
7                     :if-exists :append
8                     :if-does-not-exist :create)
9 (format str "~%~%
10 %Ejemplo:[cielo=soleado,temperatura=alta,humedad=alta,viento=debil].~%
11 jugarTenis(Ejemplo)␣:-␣member(X=Y,Ejemplo),
12 nodo(N,X=Y,raiz),␣jugarTenis(Ejemplo,N),!.~%
13 jugarTenis(Ejemplo,N)␣:-␣member(X=Y,Ejemplo),
14 nodo(N2,X=Y,N),␣jugarTenis(Ejemplo,N2).~%
15 jugarTenis(_,N)␣:-␣nodo(hoja,[X/_],N),␣write(X)."))))
```

Al aplicar la función **traducir** se generará un archivo como el siguiente:

```
1  nodo(1,cielo=soleado,raiz).
2  nodo(2,humedad=normal,1).
3  nodo(hoja,[si/_],2).
4  nodo(3,humedad=alta,1).
5  nodo(hoja,[no/_],3).
6  nodo(4,cielo=nublado,raiz).
7  nodo(hoja,[si/_],4).
8  nodo(5,cielo=lluvia,raiz).
9  nodo(6,viento=fuerte,5).
10 nodo(hoja,[no/_],6).
11 nodo(7,viento=debil,5).
12 nodo(hoja,[si/_],7).
13
14 jugarTennis(Ejemplo) :-
15 member(X=Y,Ejemplo), nodo(N,X=Y,raiz), jugarTennis(Ejemplo,N),!.
16
17 jugarTennis(Ejemplo,N) :-
18 member(X=Y,Ejemplo), nodo(N2,X=Y,N), jugarTennis(Ejemplo,N2).
19
20 jugarTennis(_,N) :-
21 nodo(hoja,[X/_],N), write(X).
```

Finalmente, al hacer una consulta con cualquier ejemplo, el árbol de Prolog determinará a qué clase pertenece.

```
1  ?- jugarTennis([cielo=nublado,temperatura=alta,humedad=alta,viento=debil]).  
2  si  
3  true.
```

