PROGRAMACIÓN PARA LA INTELIGENCIA ARTIFICIAL

Proyecto Final

David Galicia
David Hernández
Alberto 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

CONTENIDO

- Validación Cruzada
- Clasificación por Votación
- Traducción a Prolog

VALIDACIÓN CRUZADA

VALIDACIÓN CRUZADA

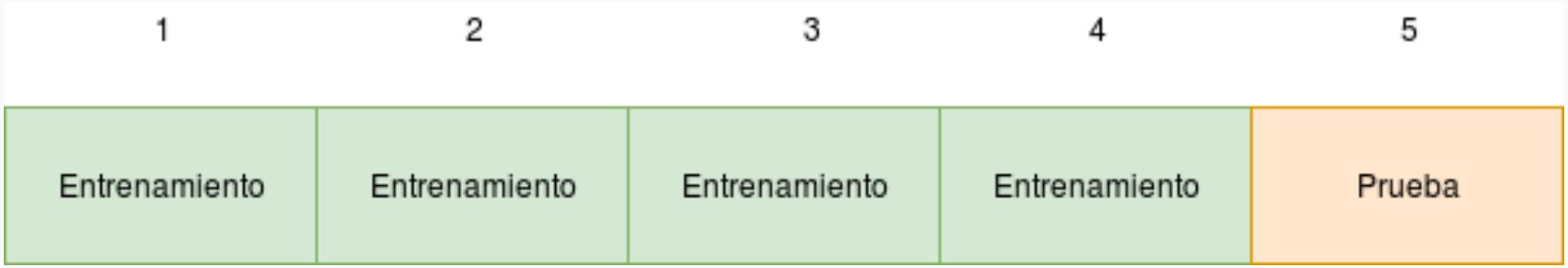

Figura: Validación cruzada con 5 iteraciones

MODIFICACIONES PARA LA VALIDACIÓN CRUZADA

```
;;; Modificaciones
(defvar *classified-int* ())
(defvar *c-classified-int* ())

(defun gui-cross-validation (data interface)
  (declare (ignore data))
  (progn
    ;; Toma K de la interfaz
    (setf k (text-input-pane-text (k-pane interface)))
    (setf *classified-int* '() *c-classified-int* '()
          *k-validation-trees* '() *classify-on* t)
    ;; Lanza la funcion con K
    (cross-validation (parse-integer k))
    ;; Escribe el mejor arbol
    (traducir *best-tree*)
    ;; Calcula y define la eficiencia
    (setf (text-input-pane-text (e-pane interface))
          (princ-to-string (/ (apply #'+ *c-classified-int*)
                              (apply #'+ *classified-int*))))
    (setf (text-input-pane-text (e1-pane interface))
          (princ-to-string (calculate-voting-accuracy *k-validation-trees*)))
    (setf (text-input-pane-text (e2-pane interface))
          (princ-to-string (calculate-best-tree-accuracy *best-tree*)))))
```

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

MODIFICACIONES PARA LA VALIDACIÓN CRUZADA

Figura: Validación cruzada con 5 iteraciones

Clasificación por votación

MODIFICACIONES PARA LA CLASIFICACIÓN POR VOTACIÓN

```
1       (defun classify-new-instance-votacion (ninstance arboles)
2         (car (repetidos
3           (loop for x in arboles
4             collect (classify ninstance x)
5           )
6         ))
7       )
```

Cuadro: Más modificaciones al Código en **cl-id3-classify.lisp**

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**

MODIFICACIONES PARA LA CLASIFICACIÓN POR VOTACIÓN

```
;;; Funcion para obtener los elementos mas repetidos de una lista
(defun repetidos (lst &optional (resultado '()))
 (if (null lst)
   (reverse resultado)
   (if (member (first lst) (rest lst))
     (repetidos (rest lst) (adjoin (first lst) resultado))
     (repetidos (rest lst) resultado))))

;;; Selecciona el mejor arbol
(defun select-bt (lst)
  (let ((cont 0) (index 0) (acc 0))
   (progn
     (loop for x in lst
           do (progn
                (when (> x acc) (and (setf acc x) (setf index cont)))
                (setf cont (+ cont 1))))
     index)))
```

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

MODIFICACIONES PARA LA CLASIFICACIÓN POR VOTACIÓN

```
1
2       (voting-accuracy-pane text-input-pane
3                             :text ""
4                             :accessor e1-pane
5                             :enabled nil)
6       (best-tree-accuracy-pane text-input-pane
7                             :text ""
8                             :accessor e2-pane
9                             :enabled nil)
```

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

MODIFICACIONES PARA LA CLASIFICACIÓN POR VOTACIÓN

```
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)))))
```

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

CLASIFICACIÓN POR VOTACIÓN

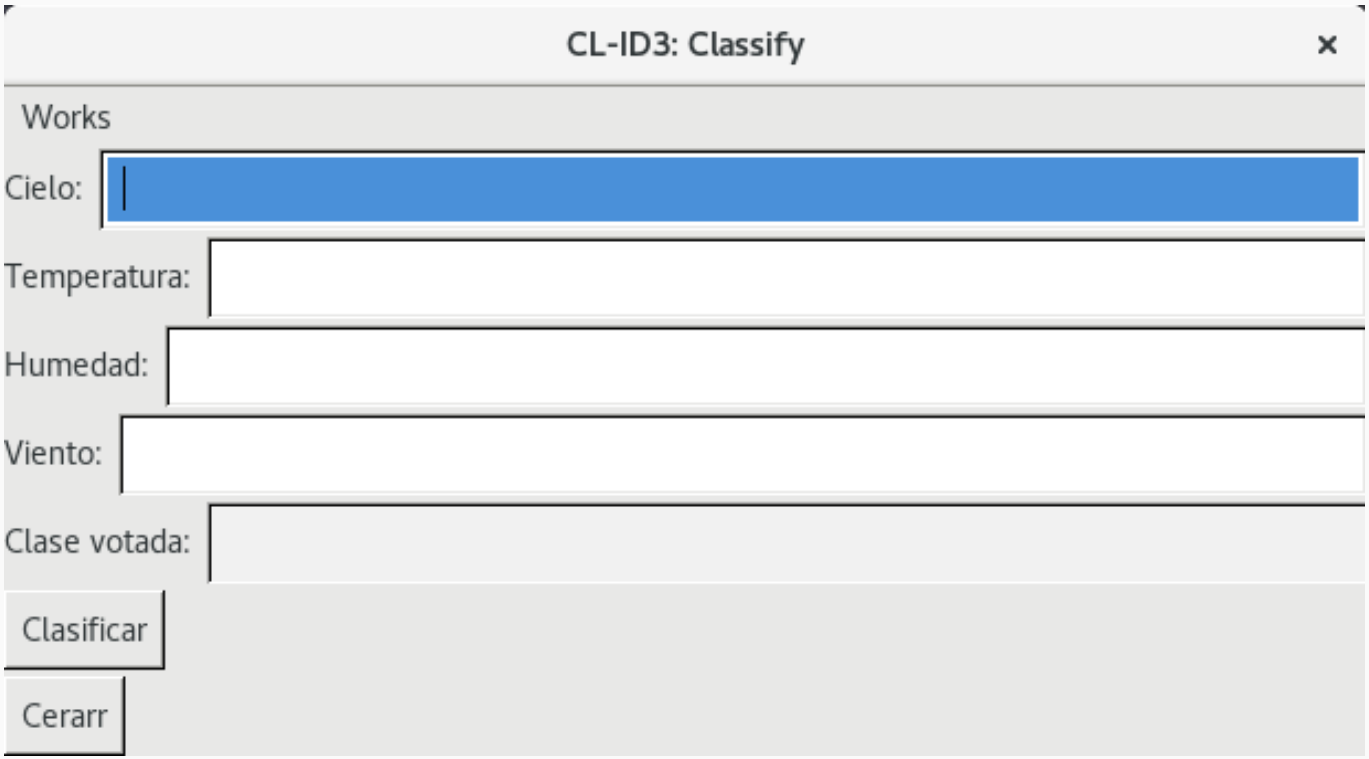

Figura: Clasificar otro Ejemplo

CLASIFICACIÓN POR VOTACIÓN

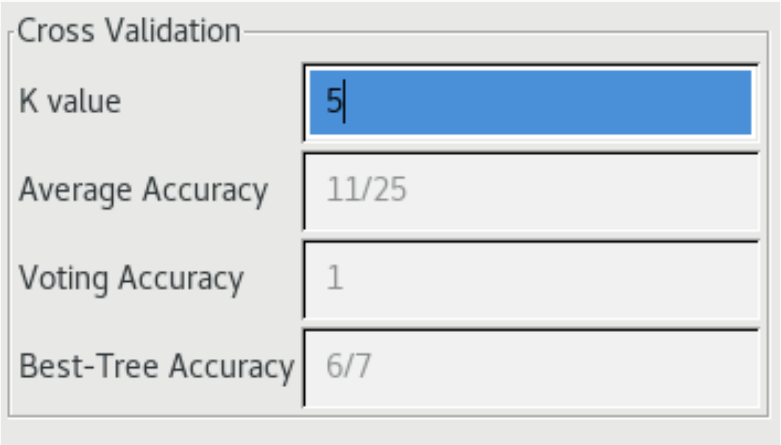

Figura: Clasificación por votación con 5 iteraciones

TRADUCCIÓN A PROLOG

TRADUCCIÓN A PROLOG

```
1
2       (defun traducir (arbol)
3
4           (progn
5           (traducir2 arbol 0 0)
6           (with-open-file (str "~/quicklisp/local-projects/cl-id3/cl-id3/arbol.pl"
7               :direction :output
8               :if-exists :append
9               :if-does-not-exist :create)
10          (format str
11          "~%~%%Ejemplo:[cielo=soleado,temperatura=alta,
12  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣humedad=alta,viento=debil].~%
13  ␣␣␣␣jugarTenis(Ejemplo)␣:-␣member(X=Y,Ejemplo),
14  ␣␣␣␣nodo(N,X=Y,raiz),␣jugarTenis(Ejemplo,N),!.~%
15  ␣␣␣␣jugarTenis(Ejemplo,N)␣:-␣member(X=Y,Ejemplo),
16  ␣␣␣␣nodo(N2,X=Y,N),␣jugarTenis(Ejemplo,N2).~%
17  ␣␣␣␣jugarTenis(_,N)␣:-␣nodo(hoja,[X/_],N),␣write(X).”))
18          )
19      )
```

Cuadro: Más modificaciones al Código en **cl-id3-cross-validation.lisp**

TRADUCCIÓN A PROLOG

```
1
2       nodo(1,cielo=lluvioso,raiz).
3       nodo(2,viento=fuerte,1).
4       nodo(hoja,[no/2],2).
5       nodo(3,viento=debil,1).
6       nodo(hoja,[si/3],3).
7       nodo(4,cielo=nublado,raiz).
8       nodo(hoja,[si/4],4).
9       nodo(5,cielo=soleado,raiz).
10      nodo(6,humedad=normal,5).
11      nodo(hoja,[si/2],6).
12      nodo(7,humedad=alta,5).
13      nodo(hoja,[no/3],7).
14
15      %Ejemplo:[cielo=soleado,temperatura=alta,humedad=alta,viento=debil].
16
17      jugarTenis(Ejemplo) :- member(X=Y,Ejemplo), nodo(N,X=Y,raiz),
18          jugarTenis(Ejemplo,N),!.
19
20      jugarTenis(Ejemplo,N) :- member(X=Y,Ejemplo), nodo(N2,X=Y,N),
21          jugarTenis(Ejemplo,N2).
22
23      jugarTenis(_,N) :- nodo(hoja,[X/_],N), write(X).
```

Cuadro: Mejor árbol en Prolog

TRADUCCIÓN A PROLOG

```
%%%?- jugarTenis([cielo=nublado,temperatura=alta,humedad=alta,viento=debil]).
%%%si
%%%true.
```

Cuadro: Ejecución de **arbol.pl**