

Proyecto Final

Dr. Alejandro Guerra-Hernández, Dr. Héctor Gabriel Acosta Mesa,
Dr. Guillermo de Jesús Hoyos Rivera
Departamento de Inteligencia Artificial
Universidad Veracruzana
Facultad de Física e Inteligencia Artificial
Sebastián Camacho No 5, Xalapa, Ver., México 91000
aguerra@uv.mx, heacosta@uv.mx, ghoyos@uv.mx
www.uv.mx/aguerra, www.uv.mx/heacosta, www.uv.mx/ghoyos

23 de febrero de 2012

Consideren el Algoritmo 1 de inducción de árboles de decisión (ID3). Como hemos visto en los cursos de aprendizaje automático y programación funcional, éste algoritmo es egoísta en el sentido que elige la mejor partición, aquella que aporta una mayor ganancia de información, y coloca al atributo asociado como raíz del árbol que se está construyendo (líneas 10–12). Tal estrategia no es sensible al caso de empates al computar la ganancia de información. El resultado es que el algoritmo regresa un árbol de decisión que no necesariamente es el mejor dado un conjunto de entrenamiento y una clase.

1. Ejercicios

1. Modifique la implementación de este algoritmo que se revisó en el curso de programación funcional para que:
 - a) Considere mejoras al algoritmo documentadas en la literatura, por ejemplo: diferentes métricas de información, discretización de atributos continuos, diferentes criterios de paro, etc. Los criterios de este punto se ajustarán a los visto en el curso de aprendizaje automático.
 - b) La salida del algoritmo sea un bosque de árboles, donde se encuentran todos los árboles computables a partir de un conjunto de entrenamiento y una clase dada. La idea central es que cuando hay empates al computar el mejor atributo, se procede a construir tantos árboles como empates existan.
 - c) Utilice el mecanismo para seleccionar el árbol en el bosque que muestra un mejor desempeño, por ejemplo *cross-validation*.

Algorithm 1 El algoritmo ID3

```
1: function ID3(Ejs, Atbs, Clase)
2:   Arbol  $\leftarrow \emptyset$ ; Default  $\leftarrow$  claseMayoria(Ejs);
3:   if Ejs =  $\emptyset$  then
4:     return Default;
5:   else if mismoValor(Ejs, Clase) then
6:     return Arbol  $\leftarrow$  tomaValor(first(Ejs).Clase);
7:   else if Atbs =  $\emptyset$  then
8:     return Arbol  $\leftarrow$  valorMasComun(Ejs, Clase);
9:   else
10:    MejorParticion  $\leftarrow$  MejorParticion(Ejs, Atbs);
11:    MejorAtributo  $\leftarrow$  first(MejorParticion);
12:    Arbol  $\leftarrow$  MejorAtributo;
13:    for all ParticionEjs  $\in$  rest(MejorParticion) do
14:      ValoAtributo  $\leftarrow$  first(ParticionEjs);
15:      SubEjs  $\leftarrow$  rest(ParticionEjs);
16:      agregarRama(Arbol, valorAtributo, ID3(SubEjs, {Atbs
17:        MejorAtributo}, Clase));
18:    end for
19:    return Arbol
20: end function
```

2. Implementar una versión basada en procesos de Lisp (hilos). Consideren la librería nativa de Lispworks 6.0 para procesos concurrentes. La idea es contar con versiones basadas en paso de mensajes y memoria compartida, que puedan compararse contra la versión no concurrente del algoritmo.
3. Escriban un artículo siguiendo el formato de las *Lecture Notes in Computer Science* de Springer (máximo 11 páginas), reportando la experiencias implementadas y analizando formalmente los resultados obtenidos.
4. Al final se organizará una presentación del trabajo llevado a cabo y defenderá las bondades del enfoque que ha adoptado