Obligatorio 2 de Paradigmas de programación **Mayo 2015**

Programación Lógica

1 Contexto

El objetivo del obligatorio es implementar en Prolog una versión simplificada del algoritmo ID3[1] propuesto en 1986 por J. R. Quinlan para construir un *árbol de decisión* a partir de un conjunto de datos llamados *ejemplos* o también *conjunto de entrenamiento*. Un árbol de decisión es lo que se conoce también como un *clasificador*, es decir, permite determinar, de acuerdo a los datos disponibles, a qué clase pertenece un elemento determinado. Este problema se enmarca dentro del contexto general de la inteligencia artificial, más precisamente en aprendizaje automático.

ID3 es un algoritmo recursivo que comienza con el conjunto de *ejemplos*. En cada iteración, ID3 calcula la *entropía* para cada atributo no considerado previamente, sobre el conjunto de datos restantes. La entropía es una medida de la incertidumbre en los datos: cuanto menor la entropía, menor la incertidumbre. Por eso, ID3 elige el atributo que tiene menor entropía y particiona el conjunto de datos de acuerdo a los valores posibles para ese atributo. ID3 continúa recursivamente analizando cada uno de los subconjuntos así obtenidos, eliminando el atributo elegido del conjunto de atributos a considerar. La recursión termina si se da alguno de los siguientes casos:

1. Todos los elementos en el conjunto de datos pertenecen a la misma clase. En este caso, ID3 construye una hoja con la clase correspondiente.

2. El conjunto de atributos es vacío, pero en el conjunto de datos hay elementos pertenecientes a más de una clase. En este caso, ID3 construye una hoja con la clase más representativa, es decir, de la que hay más elementos en el conjunto de datos.

3. El conjunto de datos es vacío. Esto puede pasar cuando no se encontraron en el conjunto padre elementos correspondientes a un cierto valor del atributo. En este caso, se construye una hoja con la clase más representativa en el padre.

[1] `http://en.wikipedia.org/wiki/ID3_algorithm`

2 Construcción del árbol

Para este obligatorio se tomaron los datos del seminario *Machine Learning Algorithms for Classification*, Rob Schapire, Princeton University[2], en el cual se puede encontrar también información adicional sobre el tema.

El programa Prolog se deberá construir a partir de los siguientes predicados de base dados:

1. `nombre/1` define los *nombres* de los individuos. `nombres/1` define la lista de todos los nombres.
2. `atributo/1` define los *atributos*. `atributos/1` define la lista de todos los atributos.
3. `clase/1` define las *clases*. `clases/1` define la lista de todos las clases.
4. Para cada constante `atr` que satisface `atributo(atr)`, existe un predicado `atr/1` que define qué individuos tienen ese atributo.
5. Los ejemplos están definidos por el predicado `clasede/2`.
6. El predicado `ejemplos(?Ns)` es tal que `Ns` es la lista de todos los *nombres* `N` para los que existe una clases `C` que verifica `clasede(N, C)`.

Se pide definir los siguientes predicados Prolog:

1. `mismaclase(+Ns, ?C)`: todos los nombres en `Ns` son de clase `C`.
2. `particion(+Ns, +A, ?NT, ?NF)`: particiona la lista `Ns` en dos listas `NT` y `NF` tal que cada una contiene, respectivamente, los nombres que tienen y no tienen el atributo `A`.
3. `proporcion(+Ns, +C, +A, ?P)`: `P` es la proporción de elementos de clase `C` en la lista `Ns` para los cuales el atributo `A` es verdadero.
4. `entropia(+Ns, +As, +A, ?E)`: `E` es la `entropia` de la lista `Ns` para el atributo `A` si pertenece a la lista de atributos `As`, o `1.0` si no. La entropía para un atributo se calcula como: $(-1) \cdot \sum_{c \in clases} p(c) \cdot \log p(c)$, donde $p(c)$ es la proporción del punto anterior. Si $p(c)$ es 0, el sumando $p(c) \cdot \log p(c)$ se reemplaza por 0.
5. `minatr(+Ns, +As, ?M)`: `M` es el atributo de la lista `As` que tiene menor entropía en `Ns`. Si hay varios posibles, se devuelve el que aparece primero en la lista.

[2] `http://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf`

6. `maxcla(+Ns, ?C)`: `C` es la clase más representativa de `Ns`.

7. `id3(+Ns, +As, +C, ?T)`, `T` es el árbol de decisión para la lista de nombre `Ns` dada la lista de atributos `As` y la clase `C`. Cada hoja de `T` es un término de la forma `hoja(C)`, dónde `C` es una clase. Cada nodo es un término de la forma `nodo(A, T1, T2)` dónde `A` es un atributo y `T1` y `T2` son árboles.

8. `arbol(?T)`: `T` es el árbol construido usando `id3` para el conjunto de ejemplos. La lista de atributos inicial es la lista de todos los atributos. La clase inicial es la más representativa en los ejemplos.

9. `clasificar(+T, +N, ?C)`: `C` es la clase del nombre `N` según el árbol `T`.

10. `clasificacion(Ns, ?Xs)`: `Xs` es la lista de clasificaciones obtenida para la lista `Ns`. Cada elemento de `Xs` es un par `(N, C)` tal que `N` es un elemento de `Ns` y `C` es una clase.

3 Entregables

Se debe entregar el archivo id3.pl listo para ser interpretado en `swipl`. No se piden reportes extra, pero los programas deben ser legibles y estar comentados apropiadamente (descripción de los predicados auxiliares utilizados cuando sea necesario). Se debe entregar todo el código en forma impresa y en un CD, en un sobre transparente.

4 Criterios de corrección

El programa debe funcionar correctamente con los datos y tests dados. La cátedra se reserva el derecho de testear los entregables con tests adicionales. Los datos y tests dados serán usados para asignar un puntaje base de 12 puntos a partir del cual se restará hasta un 30% de puntos de acuerdo a los resultados sobre los tests adicionales y a criterios de buena codificación de la solución. Los obligatorios serán revisados de forma de detectar cualquier tipo de plagio mediante el software Moss de la Universidad de Stanford.