Ingeniería del Conocimiento – Curso 05/06
Práctica 4: Adquisición Automática de Conocimiento en Reglas (Aprendizaje Automático)

Para optar al examen final de la asignatura **en la convocatoria de junio o en la de septiembre**, es obligatorio **realizar todas las prácticas de la asignatura durante el periodo de clases comprendido entre Febrero y Mayo de 2006**, siendo éste el único periodo realización y entrega de prácticas tanto para la convocatoria de junio como para la convocatoria de septiembre; Además, se deberá obtener una nota media de las cuatro prácticas no inferior a 5.0 puntos (sobre 10 puntos) para poder optar al examen final. Finalmente, como la calificación en final de prácticas (ver normativa de asignatura en la web) será individual y, por tanto, dado que el aprovechamiento individual durante la realización de las prácticas influirá de forma directa en la evaluación final de cada alumno mediante el examen final de prácticas, **es imprescindible la participación de cada alumno en todas las clases de prácticas** en los horarios y laboratorios asignados, donde podrá ser evaluado y corregido de forma continua por su profesor.

Calendario de entregas				
Grupo	Horario		Lab.	Entrega Electrónica
Mb	Lunes	16–18h	Lab 11	22–Mayo
Mc	Martes	16–18h	Lab 11	16–Mayo
Md	Miércoles	16–18h	Lab 11	17–Mayo
Me	Jueves	16–18h	Lab 11	11–Mayo
Ma	Viernes	15–17h	Lab 12	19–Mayo
Td	Lunes	10–12h	Lab 11	22–Mayo
Ta	Martes	10–12h	Lab 11	16–Mayo
Tb	Miércoles	9–11h	Lab 11	17–Mayo
Tc	Lunes	15–17h	Lab 12	22–Mayo

La práctica consta de tres partes. En la primera parte se utilizará el programa `http://www.ii.uam.es/~ic/practicas/pract4/random-trees.lsp` para, después de familiarizarse con él, añadirle funciones de extracción directa de reglas a partir de los árboles aleatorios que se generan con la función `random-tree`. En la segunda parte, se generaliza el conjunto de reglas extraído, y se mide además su rendimiento en predicción. En la tercera parte, se sustituye la función de elección aleatoria del atributo que se coloca en la raíz del árbol, utilizando en su lugar una función basada en los conceptos de entropía y ganancia de información que se estudian en teoría, y una vez hecho eso se comparan los resultados (tanto de ambas clases de árboles como de las reglas obtenidas de ambas clases de árboles).

Parte I. Descarga el programa `http://www.ii.uam.es/~ic/practicas/pract4/random-trees.lsp` y lee los comentarios para aprender a ejecutarlo con distintos conjuntos de datos. Los datos que se usan para la construcción del árbol se denominan *datos de entrenamiento*. Algunos ficheros con datos que se pueden usar como datos de entrenamiento para los problemas mencionados en clase están disponibles en `http://www.ii.uam.es/~ic/practicas/pract4/p4-datos/`. Los árboles que se generan son aleatorios, y por tanto no se parecerán semánticamente a los mostrados en clase de teoría. Sin embargo, sí que tendrán la misma estructura sintáctica, y ello te permite programar las funciones necesarias para extraer reglas del estilo:

"Si la tensión arterial es alta, y la función renal es deficiente, entonces hay riesgo con el tratamiento."

Puedes elegir la sintaxis que prefieras para estas reglas siempre que te permita llevar a cabo el resto de la práctica (sigue leyendo hasta el final del enunciado). A priori, una sintaxis parecida a la que usaban los programas `forward.lsp` y `backward.lsp` originales, sin necesidad de usar variables, parece razonable.

En esta primera parte, debes desarrollar funciones para **construir un conjunto de reglas** dado un árbol, y **evaluar un conjunto de reglas** dado un conjunto de ejemplos o casos.

La construcción de reglas puede basarse en recorrer todos los caminos desde la raíz del árbol hasta sus hojas, recordando los nodos recorridos (premisas de la regla) y la clasificación de la hoja alcanzada (consecuente de la regla). Para simplificar, podemos suponer que siempre trataremos con clasificación binaria, con lo que bastaría con sacar las reglas para una de las clases, y tratar la otra como clase a asignar cuando no se pueda demostrar la primera.

La evaluación de las reglas, requiere primero la creación de una función similar a la función `classify` (ver código) que usa un árbol para clasificar un ejemplo, pero ahora usando un conjunto de reglas para predicción de la clasificación del ejemplo. Comparando los resultados de dicha predicción, con la clase que viene prefijada para cada ejemplo, se puede obtener una tasa o porcentaje de aciertos en la predicción de un conjunto de reglas sobre un conjunto de ejemplos. Dicho porcentaje es el valor que usamos como evaluación (medida de calidad) de las reglas.

Parte II. La segunda parte de la práctica se centra en **simplificar el conjunto de reglas** obtenido en la primera parte, y probar que con la simplificación pueden llegar a ser mejores que el árbol del que se han extraído. Dada la estructura compacta de los árboles, las reglas que se extraen directamente de ellos, suelen tener condiciones innecesarias. En primer lugar se debe eliminar cualquier condición cuya eliminación no empeore la evaluación (tasa de aciertos) del conjunto de reglas utilizando los mismos datos de entrenamiento con los que se generó el árbol. En segundo lugar se deben eliminar reglas redundantes, iguales o más específicas que otras. Esta eliminación tampoco empeorará la evaluación del conjunto de reglas con los datos de entrenamiento.

Usando conjuntos de *datos de test* que son distintos a los datos de entrenamiento que se usaron para el aprendizaje del árbol, se puede mostrar que simplificando el conjunto de reglas se puede llegar a obtener mejores resultados de predicción sobre casos no vistos. Para probar esto, utiliza el problema de Tratamiento de Riesgo, para el que se facilitan tres conjuntos disjuntos de datos, que se pueden usar para entrenamiento o para test, según convenga (`http://www.ii.uam.es/~ic/practicas/pract4/p4-datos/`). Refleja los resultados en la memoria, incluyendo todos los detalles: conjunto de datos de entrenamiento, conjunto de datos de test, árbol aleatorio obtenido, reglas extraídas, y reglas simplificadas, con tasa de acierto de los tres, y justificación razonada de la mejora.

Parte III. La tercera parte de la práctica cambia un poco de dirección y se centra en **mejorar los árboles aleatorios**. Dado que el crecimiento de los árboles es recursivo (crear nodo con ramas, y llamada recursiva en los subárboles), la pieza clave es la elección del "mejor" atributo para ser colocado en la raíz de árbol (y recursivamente, en la raíz de cada uno de los subárboles). Se debe crear una función `id3-tree` similar la función `random-tree` pero que, al contrario que ésta, elija cuidadosamente el "mejor" atributo usando los criterios del algoritmo ID3 explicados en clase de teoría. Estos criterios persiguen la máxima *ganancia de información*, o lo que es lo mismo la mínima *entropía* media en el conjunto de la muestra de datos de entrenamiento.

Una vez conseguido este objetivo, se debe explicar en la memoria qué relación hay entre el tamaño de estos árboles y los aleatorios. Además, se aplicará a estos árboles el mismo proceso (sin modificación) de extracción y simplificación de reglas de las partes I y II. Aquí se debe comparar los tamaños de los conjuntos de reglas con los obtenidos a partir de árboles aleatorios. Finalmente, como se hizo en la parte II, compararemos entre sí la tasa de aciertos de los árboles con la de las reglas simplificadas a partir de ellos, pero ahora utilizando los árboles ID3. ¿Qué conclusiones o hipótesis surgen de este análisis, y qué modificación se podrían añadir a los programas y datos utilizados, para verificar o refutar dichas conclusiones o hipótesis? Detalla tu respuesta en la memoria.

Normas de entrega específicas para la práctica 4

Todo el código debe ir en único fichero, **GxxP4.lsp**, que al cargarse ejecute automáticamente e imprima los resultados de todas las pruebas realizadas por el alumno, pero especialmente las pruebas que se han utilizado para las explicaciones incluidas en la memoria. Todos los ficheros de datos para aprendizaje (extensión .dat) deben de tomarse del directorio local en el que se está trabajando, sin hacer referencia a directorios propios de cada pareja de prácticas que luego puedan no existir cuando el profesor ejecuta la práctica. Todos esos ficheros de datos utilizados en las pruebas deben incluirse también dentro del fichero comprimido **GxxP4.zip**, junto con el código **GxxP4.lsp**, la memoria **GxxMem.txt**, y pruebas adicionales **GxxPru.txt**.

Es importante que la memoria y las pruebas adicionales estén en formato TXT, PDF o HTML, pero nunca DOC (para evitar los virus).
