Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 6. Extracción de reglas

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Aprendizaje Automático

1. Introducción

La clasificación supervisada trata conjuntos de datos formados por tuplas atributos-clase. El problema puede ser bi-clase o multi-clase. Los atributos pueden ser numéricos o nominales y continuos o discretos. En este tipo de problemas puede existir ruido o datos incompletos (missing values).

Existen muchos tipos de clasificadores:

- **Árboles de Decisión**
- **Reglas de clasificación**
- Máquinas de Vector Soporte
- Redes Neuronales
- Redes bayesianas
- Sistemas difusos
- KNN

2. Árboles de Decisión

Es una representación de los procesos de decisión involucrados en la tarea de clasificación. Donde las **hojas** representan la etiqueta asociada a una clasificación, los **nodos** describen la pregunta acerca de un cierto atributo y las **ramas de un nodo** representan los diferentes valores que puede tomar dicho atributo. El **objetivo** es generar el árbol de decisión óptimo que permita describir la clasificación con el menor número de cuestiones posible.

2.1. Historia

En 1966, Hunt, Marin y Stone publican *Experiments in Induction* donde describen sistemas de adquisición de conceptos (CLS) utilizando atributos binarios y técnicas heurísticas para para construir árboles de decisión.

En 1984 se publica *Classification and regression trees* de Breiman, Friedman, Olshen y Stone donde se describen un método de inducción para construir árboles de decisión recursivamente (CART).

En 1986, J. Ross Quinlan desarrolla el ID3 que utiliza la entropía de información para crear los árboles. Posteriormente, se mejoraría creando el C4.5.

Otro algoritmo muy extendido para crear árboles de decisión es CHAID. Utiliza el contraste de Pearson (test χ^2) para seleccionar el atributo a estudiar en cada nodo.

Otros algoritmos son:

- ID4 e ID4R de Schlimmer y Fisher (1986): es una versión incremental del ID3.
- ID5 e ID5R de Utgoff (1990): versión incremental del ID3.
- J48: implementación de C4.5 (WEKA).
- C5.0: última versión mejorada de C4.5.

2.2. ID3

```
ID3 (Dataset)
    IF (all instances belong to class C) THEN
        return Sheet (C)
    ELSE IF (conj instances = empty) THEN
        return Sheet (Default_class)
    ELSE IF (set of instances does not contain any attributes) THEN
        return Sheet (Majority_Class)
    ELSE
        Choose attribute A with the highest information gain
        Create node with selected attribute
        For each V value of attribute A
            Create a branch with the value V
            Select the instances with the V value of attribute A
            Remove attribute A from this set of Cv instances
            Assign to the branch the tree returned by ID3 (Cv)
        Return node
```

Figura 1: Pseudocódigo ID3

Para obtener la mayor ganancia de información se utiliza la **entropía de Shanon** que viene dada por la siguiente expresión:

$$G(A_i) = I - I(A_i) \tag{1}$$

$$I = -\sum_{n_c \in n_{clases}} \frac{n_c}{n} \log_2(\frac{n_c}{n}) \tag{2}$$

$$I(A_i) = \sum_{j \in val_{A_i}} \frac{n_{ij}}{n} * I_{ij} \tag{3}$$

$$I_{ij} = -\sum_{c \in n_{val_j}} \frac{n_{ijc}}{n_{ij}} \log_2(\frac{n_{ijc}}{n_{ij}}) \tag{4}$$

2.3. Mejoras del ID3

- Evitar el **sobreajuste** de los datos: Si hay ruido en el entrenamiento, el algoritmo lo aprende. Si hay pocos ejemplos asociados a las hojas, no son representativos de la verdadera función.
  El ID3 produce árboles que sobreajustan los ejemplos de entrenamiento. No es capaz de generalizar.
- Determinar la profundidad del árbol.
- **Aplicando poda**: Para de aumentar el árbol antes de alcanzar el punto donde clasifica perfectamente los ejemplos de entrenamiento. Se aplica un test estadístico para estimar si expandiendo un nodo particular es probable producir una mejora más allá del conjunto de entrenamiento.
  También se puede aplicar **post-poda** que permite sobreajustar los datos, y después podarlo reemplazando subárboles por una hoja.

  Para evitar el sobreajuste se utiliza un dataset de validación que considera todos los nodos de decisión como candidatos para podar. Se eliminarán solamente los nodos si el árbol podado resultante no tiene peor resultado que el original. Convierte el nodo en hoja y elimina el resto del subárbol asignándole el valor más común de clasificación de los ejemplos asociados a ese nodo.
  Esta poda se realiza iterativamente y se realiza hasta que aumente el valor. Elimina patrones o coincidencias del entrenamiento que no aparecen en validación.

Metodología:

1. Examina los subárboles de los nodos no terminales (comenzando desde abajo).
2. Se elimina el subárbol correspondiente con raíz en ese nodo, convertir el nodo en hoja y asignarle la clasificación más común de los ejemplos de entrenamiento asociados a ese nodo.
3. Solo se poda si el árbol resultante mejora o iguala el rendimiento del árbol original sobre el conjunto de testeo.

Otra forma de realizar post-poda :

1. Crecer el árbol completamente sobre el training-set, aunque haya sobreajuste.
2. Convertir el árbol en un conjunto equivalente de reglas.
3. Podar cada regla eliminando cualquier condición que mejore la precisión estimada.
4. Ordenar las reglas podadas por su valor de precisión estimada, y considerarlas en este orden para la clasificación de nuevas instancias.

Ventajas de convertir el árbol en reglas:

- Mejor legibilidad
- Podando un nodo eliminas un atributo independiente del contexto
- Elimina la distinción entre atributos cerca de los nodos y cerca de las hojas.
- No necesita reorganizar el árbol.

- Condicionando la poda.
- Manejar atributos continuos.

  - El ID3 solo puede manejar atributos nominales. Los atributos objetivo y los atributos del árbol deben ser nominales.
  - Los atributos se pueden convertir dinámicamente en nominales creando un *booleano* (c) tal que divida los valores continuos en dos conjuntos. Para ello:
    - Ordenar los ejemplos acordes a sus valores continuos.
    - Identificar los valores adyacentes que difieran en su clasificación.
    - Generar los valores umbral, como la media de los valores adyacentes anteriores.
    - Evaluar los umbrales candidatos, con cada uno de sus dos rangos y calculando la ganancia de información.
    - Seleccionar el mejor candidato y fijar el valor de c.
- Manejo de datos de entrenamiento con valores faltantes.
  - Los valores perdidos se pueden estimar de diferentes formas:
    - Asignar el valor más común
    - Asignar el valor más común entre los ejemplos de la misma clase.
    - Asignar una probabilidad a cada uno de los siguientes valores del atributo y elegir un valor ponderado a esa probabilidad.
- Mejorar la eficiencia computacional.

2.4. Medidas Alternativas

El ID3 favorece a los atributos con muchos valores repetidos. El atributo tipo *Fecha*, por ejemplo, sería único en todos los ejemplos y produciría un árbol muy ancho y de profundidad 1. Este tipo de atributos separan el dataset en muchos subconjuntos de datos pequeños provocando una ganancia de información alta y un mal valor predictivo.

Para evitar este problema se utiliza otra medida de la información llamada **Ratio de Ganancia** (Quirlan, 1986) que penaliza a los atributos con muchos valores utilizando un término llamado *Split Information*

3. Reglas de Clasificación

3.1. Algoritmo AQ

Es una clase de algoritmo de generación de reglas de la familia *estrella*. Tiene como objetivo generar un conjunto de reglas que describan todos los ejemplos positivos y no reconozca ningún ejemplo negativo. Está formado por:

- **A**: Conjunto de Atributos.
- **V**: Conjunto de Valores de los atributos.
- **C**: Conjunto de Clases.
- **E**: Conjunto de Ejemplos de Entrenamiento.
- **LEF**: Lista de criterios de preferencias de reglas. (función de evaluación lexicográfica). Permite elegir la regla a añadir entre un conjunto de candidatas según diferentes criterios:
  - **Cobertura**: Número de ejemplos positivos cubiertos.
  - **Simplicidad**: Número de atributos que se estudian en el antecedente.
  - **Coste**: Coste de evaluación del antecedente.
  - **Generalidad**: Número de ejemplos observados entre el número de ejemplos posibles.

  El algoritmo es independiente de la función de evaluación, es decir, todas las soluciones serán equivalentes.

3.2. Selector

Permite realizar preguntas sobre un atributo. Se expresa de la forma (Atributo Operador Valor). Los operadores posibles son: $=, <, >, !=, ...$

3.3. Complejo

Es una conjunción de selectores. Permite describir el antecedente de una regla de clasificación.

3.4. Recubrimiento

Es una disyunción de complejos. Permite describir conjuntos de reglas.

```
Inicialmente el conjunto de reglas (recubrimiento) esta vacío
Se considera el conjunto P de ejemplos positivos y el conjunto N de ejemplos negativos
Mientras queden ejemplos positivos en P, repetir
    Elegir un ejemplo de P que será la semilla de la próxima regla
    Generar complejos que cubran la semilla y excluyan a los ejemplos de N (algoritmo star\foo
    Elegir de entre los complejos el que optimice el criterio de selección (LEF)
    Eliminar de P todos los ejemplos cubiertos por la nueva regla
    Añadir el complejo elegido al recubrimiento
```

Figura 2: Algoritmo para generar las reglas

- Sea **E**: conjunto de ejemplos a devolver.
- Sea **L**: lista de complejos a estudiar.
- Sea **S**: conjunto de selectores de la semilla.

```
Inicialmente: E = $\emptyset$ (conjunto vacío)
Inicialmente: L = \{ () \}, es decir contiene un complejo que acepta todo
Mientras L no esté vacía repetir
    Crear un conjunto E' con complejos creados por conjunción de un elemento de L
            y un selector de S
    Eliminar de E' los elementos que ya estén incluidos en E
    Para cada complejo de E', si no cubre ningún ejemplo negativo, entonces
        Añadir el complejo a E
        Eliminar el complejo de E'
    Actualizar la lista L a los elementos de E'
Devolver el conjunto E
```

Figura 3: Algoritmo Estrella