Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 5. Adquisición de Conceptos

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Aprendizaje Automático

1. Introducción

Aprendizaje Inductivo: consiste en inducir información de un concepto a partir de un conjunto de cosas en concreto. No requiere información previa del dominio. Se supone que todos los casos son iguales al primero hasta que se encuentre una contradicción que obligue a cambiar la información. Existen dos tipos de **aprendizaje inductivo**:

1. **Aprendizaje Inductivo Simbólico**: Utiliza una representación simbólica como redes semánticas, reglas, programación lógica... Existen varias formas de este aprendizaje:
   - **Adquisición de Conceptos**: Los ejemplos reflejan situaciones con múltiples objetos y relaciones.
   - **Clasificación Supervisada**: Los ejemplos se refieren a conjuntos atributo-valor.
   - **Programación Lógica Inductiva**: Se pretende adquirir un modelo lógico.
2. **Aprendizaje Inductivo Subsimbólico**: Utiliza una representación subsimbólica como conjuntos difusos. Utiliza algoritmos de ajuste paramétrico.

2. Análisis de diferencias o Adquisición de Conceptos

Surge en los 70s cuando se produce un declive de la aproximación neuronal. En esta época surgen los primeros lenguajes de manipulación simbólica y se intentan obtener mecanismos de aprendizaje simbólico generales con muy poca información de partida. La investigación se centra en problemas de juegos.
Se proponen varios algoritmos:

- **Winston (1970)**: Se considera el punto de partida del **aprendizaje basado en similitudes**. El autor lo denomina *learning by analyzing differences* e introduce el concepto de **quasi-ejemplo** (ejemplo **negativo** que se diferencia en una sola propiedad del modelo). Este algoritmo utiliza como representación las redes semánticas tanto como para los ejemplos como para los conceptos.
  El dominio de aplicación es el mundo del bloque.
  Los elementos del algoritmo son:
  - **Lenguaje de representación**: Redes semánticas.
    - Una red semántica es una forma de presentación del conocimiento lingüístico. En ella, los conceptos y sus interrelaciones se representan mediante un grafo (si es acíclico se puede representar como un árbol).
    - Los elementos semánticos se representan por nodos y las relaciones por aristas.
    - Se usan para representar mapas conceptuales y mentales.
  - **Mecanismo de cotejamiento**: Método para analizar las diferencias.
    - Permite comparar dos redes semánticas que corresponden a la definición actual del concepto y a la definición del nuevo ejemplo a tratar.
    - Si el ejemplo es positivo se dirige la búsqueda a generalizar la definición, mientras que si es negativo se dirige a especializarla.
    - Se pueden modificar las etiquetas de las relaciones entre nodos, pero no los nodos (en todo caso se pueden añadir nodos **no se pueden borrar**).
    - Se introducen las heurísticas **require-link** y **forbid-link**:
      - **Require-Link**: Se emplea cuando el modelo del concepto que está siendo aprendido tiene una etiqueta k en un lugar donde un quasi-ejemplo no. En la red semántica que representa el concepto, la etiqueta se convierte en **debe** (*must*).
      - **Forbidden-Link**: Se aplica cuando un quasi-ejemplo tiene una etiqueta i en un lugar donde el modelo no. Se coloca una etiqueta **no-debe** (*must-not*) en el modelo actual del concepto.

- **Proceso de generalización**: Para incorporar ejemplos positivos al modelo.
  - Coteja el ejemplo positivo con el modelo actual del concepto y procesa todas las diferencias.
  - Si falta una etiqueta, se elimina del concepto.
  - Si hay diferencia en el valor de una propiedad, se modifica el rango de dicha propiedad.
  - Si una etiqueta apunta a una clase diferente:
    - Si la clase pertenece a una jerarquía, se sube en la jerarquía.
    - Si la clase no pertenece a una jerarquía, se elimina.
- **Proceso de especialización**: Para rechazar quasi-ejemplos del modelo.
  - Coteja el ejemplo negativo con el modelo actual del concepto.
    - Si hay más de una diferencia, se ignora el ejemplo.
    - Si hay una única diferencia (quasi-ejemplo):
      1. Si el modelo tiene una etiqueta que el ejemplo no tiene, genera una etiqueta **required-link**.
      2. Si el ejemplo tiene una etiqueta que el concepto no tiene, genera una etiqueta **forbidden-link**.

2.1. Algoritmo Winston

1. Toma como modelo inicial la descripción de la primera instancia positiva del concepto. (**definición del concepto = modelo**).
2. Examinar la descripción de otras instancias positivas conocidas del concepto. **Generalizar** la definición del concepto para incluirlas.
3. Examinar las descripciones de los *quasi-ejemplos* del concepto. **Especializar** la definición del concepto para excluirlas.
4. Se intercalan los pasos 2 y 3 a medida que se tratan ejemplos positivos o negativos del concepto. Se repite hasta que no haya más cambios.

Propiedades:

- Es un algoritmo conservador, es decir, si existen dudas sobre lo que hay que aprender, no lo aprende.
- El aprendizaje se realiza en pasos pequeños (ley de Martin). Solo se puede aprender de los quasi-ejemplos.
- El algoritmo se ha descrito de manera no determinista. En cualquier punto puede haber varias posibles generalizaciones o especializaciones aplicables.
- La elección no lleva necesariamente a la hipótesis más sencilla. Puede ocurrir que lleguemos a una situación en la que ninguna modificación sencilla de la hipótesis la haga consistente con todos los ejemplos. Es necesario entonces **backtracking** al punto de elección.

Desventajas:

- Hay que comprobar que cada modificación es consistente con todos los ejemplos.
- Es muy sensible al orden del dataset.
- Es difícil encontrar una buena heurística para elegir la mejor modificación y el backtracking es muy costoso.

- Hayes-Roth (1977)
- Vere (1975)
- Michalski-Dietterich (1981)

3. Espacio de versiones

En 1982, Mitchell propone un marco unificado para la adquisición de conceptos, independiente de la representación a utilizar, denominado **espacio de versiones**.
Su objetivo es el mismo que con Winston: producir una descripción de un concepto a partir de un entrenamiento con ejemplos positivos y negativos. Sin embargo, este no se ve afectado por el orden en el que se presentan los ejemplos y, en lugar de describir un único concepto, mantiene un conjunto de descripciones posibles hasta llegar a la definición del mismo.

3.1. Nomenclatura

- **Hipótesis:** Representación de un concepto.
- **Espacio de hipótesis:** El conjunto de todos los conceptos que pueden ser descritos con la representación escogida.
- **Espacio de versiones:** Subconjunto del espacio de hipótesis que contiene todas las hipótesis consistentes con el conjunto de ejemplos positivos y negativos. (Reconocen a todos los ejemplos positivos y rechazan a todos los negativos).
  - **Orden Parcial**: h_1 es más general que h_2 ($h_1 >_g h_2$) si el conjunto de instancias cubierto por h_2 es subconjunto del conjunto de instancias cubierto por h_1
  - Para representar el espacio de versiones podemos hacer dos cosas:
    1. Enumerar todas las hipótesis: sería inviable por el tamaño.
    2. Conjunto más general (G) y el más específico (S).
  - $\geq_g$ no depende del concepto que se va a aprender. Este operador define un *orden parcial* sobre el conjunto de las hipótesis. Existe también la versión estricta ($>_g$) y la versión opuesta ($\leq_g$)

3.2. Algoritmo Find-S

1. Comienza con la hipótesis más específica: $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
2. Generaliza si el ejemplo **positivo** no está cubierto por la hipótesis.

Pseudocódigo:

```
1  Initialize the most specific hypothesis h
2
3  For each positive instance x:
4     For each attribute ai in h:
5        if x[ai] satisfy the constraint of h:
6           Nothing
7        else:
8           Replace ai in h with a generalization
9
10 return h
```

Figura 1: Pseudocódigo del algoritmo Find-S

Proceso de generalización:

- Si $h_i = \emptyset$ y lo coteja con un ejemplo e_i, entonces la hipótesis generalizada es $g_i = e_i$.
- Si $h_i =?$ (cualquier cosa) y lo coteja con un ejemplo e_i, entonces $g_i =?$.
- Si $h_i = e_i$ y lo coteja con un ejemplo e_i, entonces $g_i = e_i$.
- Si $h_i = e_i$ y lo coteja con un ejemplo $\neq e_i$ entonces $g_i =?$.

Proceso de especialización:

- Si $h_i =?$ y lo coteja con un ejemplo e_i, entonces genera una hipótesis por cada $g_i \neq e_i$.
- Si $h_i \neq e_i$ y $h_i \neq ?$ entonces la hipótesis no cubre el ejemplo negativo.
- Si $h_i = e_i$ entonces $g_i = \emptyset$
- Si $h_i = \emptyset$ y se coteja con un ejemplo e_i, no seguimos, ya que en el momento en el que se encuentre que una hipótesis donde un atributo es vacío los datos no serán adecuados.

Desventajas:

- Siempre genera una hipótesis consistente con los ejemplos (ignora los ejemplos negativos).
- No asegura que se haya aprendido el concepto correcto, porque coge una de las hipótesis posibles.
- No soporta el ruido en los ejemplos positivos. Si hay un falso positivo, no aprenderá nada.

3.3. Todas las Hipótesis

Este algoritmo construye el espacio de hipótesis completo Dado un ejemplo positivo se excluyen las hipótesis que no lo cubren y, dado un ejemplo negativo, se excluyen todas las hipótesis que sí lo cubren.

Pseudocódigo

```
1 for each example x in dataset:
2     for each hipotesis h in VS:
3         if x not satisfy h:
4             remove h
5
6 Return remaining VS
```

Figura 2: Pseudocódigo del algoritmo Todas las hipótesis.

3.4. Eliminación de candidatos

Pseudocódigo:

```
1. Sea G el conjunto de elementos de maxima generalidad de H.
2. Sea S el conjunto de elementos de maxima especificidad de H.

3. Para cada ejemplo d del conjunto de entrenamiento D:
  3.1 Si d es un ejemplo positivo, entonces:
    3.1.1 Eliminar de G cualquier hipotesis inconsistente con d.
    3.1.2 Para cada hipotesis s de S inconsistente con d:
      * Eliminar s de S.
      * Incluir en S todas las generalizaciones minimales h de s, tales que h
        es consistente con d y existe una hipotesis en G mas general que h.
      * Eliminar de S aquellas hipotesis tales que exista en S otra hipotesis
        mas general.

  3.2 Si d es un ejemplo negativo, entonces:
    3.2.1 Eliminar de S cualquier hipotesis inconsistente con d.
    3.2.2 Para cada hipotesis g de G inconsistente con d:
      * Eliminar g de G.
      * Incluir en G todas las especializaciones minimales h de g, tales que h
        es consistente con d y existe una hipotesis en S mas especifica que h.
      * Eliminar de G aquellas hipotesis tales que exista en G otra hipotesis
        mas especifica.
```

Figura 3: Pseudocódigo del algoritmo de eliminación de candidatos

Las opciones anteriores son muy costosas, por lo que debemos plantear una solución más tratable.

Este algoritmo mantiene las cotas de hipótesis superior e inferior de todas las hipótesis consistentes con los ejemplos. Cada tratamiento de ejemplos generaliza y especializa el grafo de las hipótesis para obtener un espacio final compatible.

Este algoritmo tiene como entrada un conjunto de datos y como salida un conjunto G que representa las hipótesis genéricas maximales y un conjunto S que representa las hipótesis específicas minimales.

Las hipótesis se representan en un retículo con orden parcial.

Propiedades Sean S y G obtenidos por eliminación de candidatos:

- Si S y G no están vacíos, resultan ser la cota específica y la cota general del espacio de versiones.
- Si $S = G = \{h\}$, entonces h es la única hipótesis de H consistente con todos los ejemplos.
- Si $S = G = \emptyset$ entonces no existe ninguna hipótesis en el espacio de hipótesis consistente con los ejemplo.
- El algoritmo converge hacia el objetivo siempre que:
  - El conjunto de entrenamiento sea suficientemente grande.
  - No haya ruido.
  - El concepto objetivo está en el espacio de hipótesis.