TEMA 4 APRENDIZAJE POR ANALOGÍA

Francisco José Ribadas Pena, Santiago Fernández Lanza

Modelos de Razonamiento y Aprendizaje
5º Informática
ribadas@uvigo.es, sflanza@uvigo.es

4 de marzo de 2013

4.1 Introducción
4.2 Medidas de similaridad
4.3 Método de los k vecinos más próximos
- Funcionamiento
- Interpretación
- Mejoras y variantes
4.4 Razonamiento basado en casos

4.1 Introducción

También: aprendizaje basado en instancias (*Instance Based Learning (IBL)*, aprendizaje *perezoso* (*"lazzy"*)

Idea Base: Almacenar los ejemplos de entrenamiento.

- No se construye representación explícita de la hipótesis aprendida
  - Simplemente se almacenan los ejemplos de entrenamiento
  - La generalización tiene lugar cuando haya que clasificar un ejemplo nuevo
- La clasificación de nuevos ejemplos se obtiene midiendo su "*similaridad*" con los ejemplos memorizados.

  Fase de entrenamiento Almacenamiento ejemplos en estructuras de datos adecuadas

  Fase de clasificación Comparación del nuevo ejemplo con todos o con parte de los ejemplos almacenados
  - la salida se construye en base a los ejemplos más parecidos

    $\rightarrow$ normalmente simple copia de la salida del ejemplo/s más cercano/s
  - normalmente se omiten los ejemplos no parecidos, pero pueden tenerse en cuenta
  - es necesario definir medida de similaridad entre ejemplos

- Se dice que es un tipo de aprendizaje *"perezoso"* (*lazzy learning*) o retardado
  - La generalización se postpone hasta el momento de realizar la clasificación de cada ejemplo.
  - No se construye una representación de la hipótesis objetivo en base a todos los ejemplos del conjunto de entrenamiento.
  - Implícitamente se estima una nueva "*función objetivo*" cada vez que se realiza la clasificación de un ejemplo nuevo.
    - Se realiza una generalización específica para cada ejemplo concreto.
- En el peor de los casos, requeriría comparar cada nuevo ejemplo con todos y cada uno de los ejemplos de entrenamiento almacenados.
  - alto coste en la clasificación si nº de ejemplos de entrenamiento es muy grande.
  - necesidad de mecanismos de acceso y comparación con ejemplos almacenados eficiente.

4.2 Medidas de Similaridad

- **Aspecto crucial en métodos IBL:** definición de funciones de similaridad entre ejemplos adecuadas.
  - depende del problema y del tipo de atributos que se manejen
  - en general, ejemplos podrán representarse en un espacio $n-$dimensional ($n =$ nº atributos)

Ejemplo X con n atributos: $[a_1(x), a_2(x), \ldots, a_n(x)]$

- **Atributos continuos:** uso de la *distancia euclídea*
  - Similaridad entre ejemplos x e y = distancia entre sus vectores de atributos

$$d(x,y) = \sqrt{\sum_{i=1}^{n} \left(a_i(x) - a_i(y)\right)^2}$$

  - En general, valores de atributos normalizados en el rango $[0, 1]$
- **Atributos no continuos:** $\begin{cases} \text{convertirlos en números (si posible)} \\ \text{uso de métricas discretas (dist. } \textit{Hamming}) \end{cases}$
  - Distancia *Hamming*: x e y ejemplos con n atributos discretos

$$d_{Hamming}(x,y) = \textstyle\sum_{i=1}^{n} \delta\left(a_i(x), a_i(y)\right)$$

$$\text{con } \delta(a,b) = \begin{cases} 0 & \text{si } a = b \\ 1 & \text{si } a \neq b \end{cases}$$

- Otras opciones:
  - Medidas de similaridad no numéricas basadas únicamente en valores simbólicos/discretos
  - Si los valores simbólicos están estructuradas (ontologías, redes semánticas) se podrán definir similaridades numéricas (no $\{0,1\}$) entre los valores de los atributos

- Existe la posibilidad de definir otras métricas específicas dependientes del dominio.
  - No necesitan cumplir las propiedades matemáticas de las distancias.
  - Ejemplo: Modelo vectorial de recuperación de información
    - Coseno del ángulo entre los vectores que representan documento y consulta
    - Máxima similaridad (idénticos): 1 Mínima similaridad: 0

- Similaridad entre conjuntos.
  - Coeficiente Jaccard:

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  - Coeficiente del Dado:

$$sim(A, B) = \frac{2 \cdot |A \cap B|}{|A| + |B|}$$

  - Coeficiente del Coseno:

$$sim(A, B) = \frac{|A \cap B|}{\sqrt{|A|} \cdot \sqrt{|B|}}$$

  - Coeficiente de Similaridad Mutua:

$$sim(A, B) = \frac{|A \cap B|}{\frac{2}{\frac{1}{|A|} + \frac{1}{|B|}}}$$

  - Coeficiente de Solapamiento:

$$sim(A, B) = \frac{|A \cap B|}{min(|A|, |B|)}$$

4.3 Método de los k vecinos más próximos

k-NN: k *nearest neighbours*

(a) FUNCIONAMIENTO

- Cada ejemplo de entrenamiento almacenado es un par $x, f(x)$

  $x = [a_1(x), \ldots a_n(x)]$: descripción del ejemplo (vector n dimensiones)
  $f(x)$: clasificación del ejemplo

- Procedimiento de clasificación:
  Dado un ejemplo nuevo: $y = [a_1(y), \ldots, a_n(y)]$
  1. Calcular la distancia entre el nuevo ejemplo, y, y cada ejemplo de entrenamiento memorizado, x_j.
  2. La clasificación del nuevo ejemplo se obtiene de las clasificaciones de los k ejemplos memorizados más similares (k-vecinos más próximos)
  3. El valor estimado $\tilde{f}(y)$ será la clase **más común** (*moda*) de entre esos k ejemplos más similares.
     - Algoritmo de votación simple: gana la clase que aparece más veces.
       Siendo $C = \{c_1, c_2, ..., c_l\}$ el conj. de posibles clases y siendo $x_1, x_2, ..., x_k$ los k ejemplos almacenados más similares a y

$$\tilde{f}(y) = argmax_{c \in C} \sum_{i=1}^{k} \beta(f(x_i), c)$$

       con $\beta(f(x_j), c) = \begin{cases} 1 \text{ si } f(x_j) = c \\ 0 \text{ si } f(x_j) \neq c \end{cases}$
     - Con $k = 1$, $\tilde{f}(y) = f(x_j)$, con x_j el ejemplo que minimiza la distancia $d(y, x_j)$
     - Valores $k > 1$ aumentan la tolerancia a ruido (errores en conj. de entrenamiento)

       $\rightarrow$ normalmente k será impar

- Funciones objetivo continuas
  - El método de los $k-$vecinos puede aplicarse en tareas de aproximación de funciones (predicción numérica, regresión)
  - La salida deseada, $f(y)$, es un número real.
  - Su estimación, $\tilde{f}(y)$, se obtiene como la **media** de las salidas asociadas a los $k-$ejemplos más similares.
    Siendo $x_1, x_2, ..., x_k$ los k ejemplos almacenados más similares a y.

$$\tilde{f}(y) = \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

- **Extensión:** k-vecinos ponderado
  - Dar mayor relevancia a los vecinos más cercanos al nuevo ejemplo
  - Se asocia un peso al "voto" de cada ejemplo
  - Normalmente se ponderá la contribución de cada vecino en base a su distancia al ejemplo
    - ◇ mayor peso a los ejemplos más cercanos
    - ◇ uso de la inversa del cuadrado de la distancia ($w_i = \frac{1}{d(y,x_i)^2}$)
  - Clasificación:
    - con salida discreta:

$$\tilde{f}(y) = argmax_{c \in C} \sum_{i=1}^{k} w_i \beta(f(x_i), c)$$

    - con salida continua:

$$\tilde{f}(y) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

- Usando la ponderación según la distancia inversa no es necesario restringir la votación a los k ejemplos más cercanos, puede usarse la totalidad de los ejemplos almacenados.
  $\rightarrow$ ejemplos lejanos tendrán "poco efecto"
  - Métodos globales: usan *todos* los ejemplos de entrenamiento para construir la salida
    $\rightarrow$ aumenta el coste temporal de la clasificación
  - Métodos locales: usan sólo los k más cercanos
  - En cualquier caso será conveniente estructurar los ejemplos memorizados para agilizar la clasificación (comparación lineal demasiado costosa)

(b) Interpretación método $k - NN$

- En $k - NN$ no se construyen de forma explícita hipótesis sobre el concepto a aprender
  $\Rightarrow$ Si existen *"hipótesis implícitas"* específicas para cada ejemplo.
- La casificación "aprendida" como resultado de la selección de los k ejemplos más similares de entre todos los almacenados realiza una *división del espacio de ejemplos* (de dimensión n)
- La función de similaridad, los k ejemplos seleccionados y el método de construcción de la salida (votación, media, etc,...) determinan el *espacio de decisión*
  - Se asignará una clase en función de la cercanía de cada "punto" del espacio de ejemplos a los ejemplos de entrenamiento memorizados más próximos.
- En general: $k - NN$ ofrece mayor tolerancia a ruido (ejemplos erróneos) y mejor tratamiento de las excepciones que los métodos anteriores (árboles de decisión y reglas).

- **Ventajas:**
  - Entrenamiento rápido (sólo memorizar ejemplos (todos o parte))
  - Capacidad para aprender hipótesis muy complejas (admite superficies de decisión complejas)
  - No presenta problemas de pérdida de información
  - Buena tolerancia al ruido en los ejemplos de entrenamiento
- **Inconvenientes:**
  - Lentitud en clasificación
  - Muy sensible a la presencia de atributos irrelevantes
- En general, $k - NN$ útil si:
  - ejemplos son "mapeables" como puntos en $\mathbb{R}^n$
  - num. no demasiado elevado de atributos (num. dimesiones, n, reducido)
  - grandes cantidades de ejemplos de entrenamiento disponibles

(c) Mejoras

Principales aspectos susceptibles de mejora:

- selección de atributos relevantes
- selección de ejemplos a almacenar

1. **Atributos relevantes**
   - Problema del método $k - NN$: las métricas estándar asignan la misma importancia a cada atributo de los ejemplos (*problema de la dimensionalidad*)
   - Si hay pocos atributos realmente relevantes $\Rightarrow$ atributos irrelevantes pueden desviar la la clasificación
     $\rightarrow$ necesidad de identificar/ponderar atributos útiles
   - Métodos de selección de atributos relevantes

     a) Ejecuciones previas utilizando distintos conj. de atributos
     - Selección de pequeños conjuntos de entrenamiento y prueba
     - Comenzar con un conj. de atributos candidatos vacío
     - Ir añadiendo atributos y evaluando el rendimiento obtenido sobre ese conj. de prueba reducido
     - Omitir atributos que degraden rendimiento
     - Problemas: alto coste en entrenamiento + dependencia del orden de inserción de atributos

     b) Ponderación de atributos
     - Cada atributo tendrá un peso calculado, p. ej., en base a su ganancia de información
       $\rightarrow$ equivale a asociar un *factor de escala* a los "ejes" del espacio de ejemplos
     - Otra opción: construir primero un árbol de decisión
       - Usar/ponderar atributos de acuerdo a su "importancia" (profundidad) en el árbol
       - Adicionalmente, el árbol es útil para estructurar los ejemplos (mejorar tiempos clasificación)

2. **Selección de ejemplos**
   - Idea: Almacenar sólo un subconjunto de los ejemplos de entrenamiento disponibles
     - Reducción del coste computacional durante la fase de clasificación
     - Posiblilidad de aumentar la capacidad de abstracción
   - *Modelo clásico* (modelos de proximidad)
     - Aprendizaje = almacenamiento de TODAS las instancias
     - Clasificación $\begin{cases} \text{cálculo similaridad con TODAS las instancias} \\ \text{clasificar usando ejemplos más próximos} \end{cases}$
     - Grado de abstracción bajo (depende del num. de ejemplos disponibles)
   - *Modelo de los mejores ejemplos*
     - Asumen que existen "*prototipos*" para cada clase
     - Un protitipo de clase se representa mediante un conjunto de ejemplo de esa clase
     - Idea base: almacenar **sólo** los ejemplos prototipo de cada clase
       - Entrenamiento incorpora una fase previa de construcción de esos prototipos
       - Necesidad de utilizar algoritmos de *clustering* (vectores medios, SOM, etc ...)
     - Mayor capacidad de abstracción

- *Modelo de selección de ejemplos*
  - No asume la existencia de prototipos
  - Almacena sólo parte de los ejemplos (los realmente útiles)
  - El aprendizaje incorpora un proceso incremental de selección de ejemplos

    Métodos de crecimiento (IB2, Aha, 1991)
    - Comenzar con un conjunto de ejemplos memorizados vacío
    - Para cada ejemplo de entrenamiento:

    a) Si es clasificado correctamente con sus k vecinos más proximos $\Rightarrow$ desecharlo

    b) Sino $\Rightarrow$ añadirlo al conj. de ejemplos almacenados

    Métodos de decrecimiento (IB3, Aha, 1991)
    - Idea: Eliminar "excepciones improductivas"
    - Comenzar con todos los ejemplos de entrenamiento y recorrerlos uno a uno
    - Mantener los que se clasifiquen mal en base a sus vecinos
    - Ir eliminando a los que sean **buenos predictores** para las clases de sus ejemplos vecinos
    - Objetivo: Quedarse sólo con los ejemplos situados en los límites de las regiones de decisión
  - Importante: el orden en que se presentan los ejemplos es relevante
  - Menores requisitos de espacio y coste de clasificación
  - Mejora abstracción (elimina ruido)
  - Mayor complejidad en el aprendizaje

4.4 Razonamiento Basado en Casos

Case Based Reasoning (CBR)

- Aproximación alternativa para el desarrollo de sistemas inteligentes
  - Dificultades de los sistemas inteligentes "convencionales" (basados en el conocimiento)
    - Difultades en la extracción del conocimiento
      - ◇ disponibilidad de expertos
      - ◇ problemas de comunicación
    - Dificultades para representar (reglas, lógica, redes semánticas,...) el conocimiento obtenido
- Se aplican ideas similares a las del aprendizaje por analogía (IBL) utilizando ejemplos (casos) basados en descripciones simbólicas complejas.
- **Idea básica:** *"Un sistema CBR resuelve problemas nuevos mediante la adaptación de soluciones previas utilizadas en la resolución de problemas similares"*
- **Ventajas**
  - Para el experto, más sencillo "contar" sus experiencias concretas que proporcionar reglas generales
  - Permiten proponer soluciones en dominios que nos se comprenden completamente
    - no requiere un modelo explícito del dominio
    - útil en dominios poco formalizados
  - Facilidad para incorporar nuevo conocimiento
    - basta añadir nuevos casos, no exige descubrir y generalizar nuevas reglas

- Posibilidad de ir "creciendo", reflejando la experiencia acumulada
  - la posibilidad de aprendizaje está integrada en la propia arquitectura del sistema
  - aprenden de su propio funcionamiento
  - si funcionamiento propuesto es correcto/útil, incorporará esa solución propuesta como un nuevo caso
- En general, más rápido y sencillo reutilizar una solución previa que "crearla" desde cero.
- Posibilidad de aportar explicaciones: apoya solución propuesta en los casos que la aconsejan

- **Funcionamiento**
- Casos: *"Piezas" de conocimiento reutilizable aplicable en determinados contextos*
- Representación de casos:
  1. Descripción de la situación/problema: objetivos, restriciones, características
  2. Descripción de la solución: características solución, modo de obtenerla, justificación
  3. Resultado: Éxito o fracaso en el problema real

  Técnicas de representación: { pares atributo-valor (arrays); registros de BD; lógica/Prolog; Frames, redes semánticas y/o objetos }
- Ciclo básico de los sistemas CBR:

- Pasos:
  1. Determinar las caraterísticas relevantes de la situación/problema actual
  2. Recuperar casos con características similares que resuelvan situaciones/problemas parecidos
  3. Comparar situación actual con situaciones/problemas recuperados y adaptar sus soluciones al problema actual
  4. Almacenar descripción de situación/problema actual junto con la solución propuesta y (opcionalmente) su resultado como un nuevo caso
- Componentes:

  Recuperación

  - Pasos típicos { valorar situación: determinar caract. relevantes probl. actual; búsqueda en memoria de casos similares; ordenación/ranking de casos seleccionados; selección de casos más prometedores }
  - Necesidad de mecanismo de indexación de casos (acceso eficiente a casos relevantes)
  - Compromiso entre capacidad de:
    { abstracción: representación en dominios diversos (flexibilidad)
    { concrección: facilidad en la identificación (exactitud)
  - Métricas de similaridad entre casos complejos (atributos discretos y/o estructurados)
  - Técnicas de recuperación/organización de la base de casos
    - lineal
    - medidas de proximidad ($k - NN$)
    - árboles de decisión

Reutilización

- Utilización/adaptación del conocimiento presente en los casos recuperados para resolver el problema actual
- Análisis de diferencias entre caso actual y casos recuperados
- Aproximaciones:
  - Adaptación estructural: copia y/o modificación de la salida de los casos recuperados
  - Adaptación derivacional: reutiliza (reaplica) los métodos o reglas usados sobre los casos recuperados trasladándolos al caso actual

Revisión

- Comprobación y (opcionalmente) corrección de la solución propuesta
- Normalmente externa al sistema y generalmente con participación de los usuarios
- Elemento básico de la adaptación y "autoaprendizaje" de los sistemas CBR

Recuerdo

- Permite el aprendizaje y la adquisición de nuevas experiencias
- Mejora la eficiencia del sistema al hacer disponibles un mayor número de casos