Aprendizaje Relacional

Aprendizaje Automático

Ingeniería Informática

Fernando Fernández Rebollo y Daniel Borrajo Millán

Grupo de Planificación y Aprendizaje (PLG)
Departamento de Informática
Escuela Politécnica Superior
Universidad Carlos III de Madrid

27 de febrero de 2009

En Esta Sección:

12 Evaluación de Hipótesis
- Introducción
- Intervalos de Confianza
- Comparación de Hipótesis
- Validación Cruzada y t-test

13 Programación Lógica Inductiva
- Introducción
- Programación Lógica Inductiva (ILP)
- FOIL

14 Aprendizaje Relacional
- S-CART: Structural Classification and Regression Trees
- Aproximaciones Basadas en Distancias
- Aprendizaje por Refuerzo Relacional
- Otras Aplicaciones de ILP
- Conclusiones

Árboles de Decisión y Regresión: S-CART

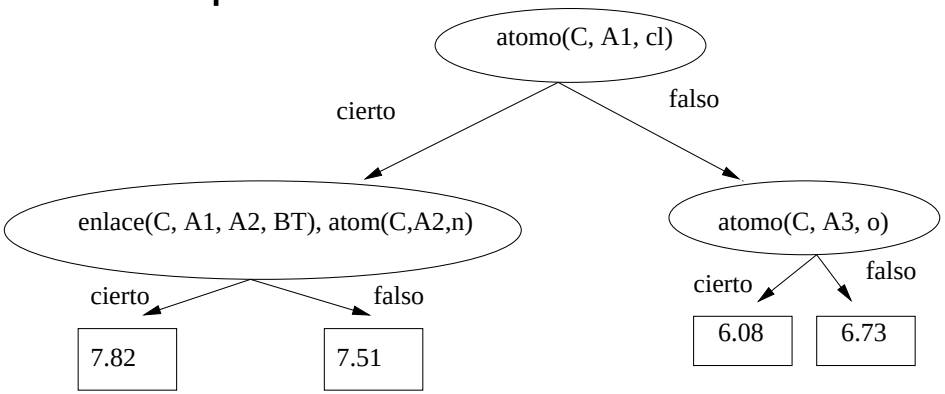
- S-CART: Structural Classification and Regression Trees
- Derivado de CART: árboles de decisión con representación atributo-valor
- Permite:
  - Clasificación: predicción de clases discretas
  - Regresión: predicción de valores continuos.
- Considera propiedades estructurales de los ejemplos en los nodos de decisión

Árboles de Decisión y Regresión con Representación Atributo-Valor

- Ventajas de los árboles de decisión:
  - Baja complejidad computacional
  - Buena aceptación por los usuarios
  - Manejo efectivo del ruido y la incertidumbre
  - Base teórica bien entendida
- Desventajas de los árboles con representación proposicional:
  - Vectores de atributos de tamaño fijo: ejemplos de entrenamiento como una matriz de valores, de la que no se puede obtener información estructural
  - Problema de la replicación y la fragmentación
  - Inestables ante el conjunto de entrenamiento

Árboles de Decisión y Regresión con Representación Relacional

- Árbol de decisión que predice la biodegradabilidad de un componente a partir de su estructura.
- La cantidad a predecir es el logaritmo de la mitad del tiempo de biodegradación del componente en agua.
- En la figura, C es un componente, A_i es un átomo, y BT es un tipo de enlace



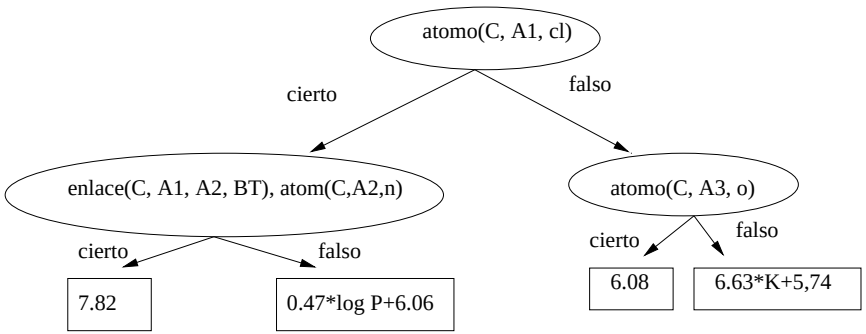
Árbol de Regresión

```
actividad(C, A) :- átomo(C, A1, cl),
                   enlace(C, A1,A2, BT),
                   átomo(C, A2, n),
                   A is 7.82, !.
actividad(C, A) :- átomo(C, A1, cl),
                   A is 7.51, !.
actividad(C,A) :- átomo(C, A3, o),
                   A is 6.08, !.
actividad(C,A) :- A is 6.73.
```

Representación Prolog

Árboles de Decisión y Regresión con Representación Relacional

- Un árbol de clasificación o decisión es como el anterior, pero en sus hojas, en vez de valores, tiene clases.
- Árbol con modelos de regresión:



Árbol de Regresión

actividad(C, A) :- átomo(C, A1, cl),
enlace(C, A1,A2, BT),
átomo(C, A2, n),
A is 7.82, !.
actividad(C, A) :- átomo(C, A1, cl),
A is $0{,}47 * \log P + 6{,}06$, !.
actividad(C,A) :- átomo(C, A3, o),
A is 6.08, !.
actividad(C,A) :- A is $0{,}63 * K + 7{,}74$.

Representación Prolog

S-CART: Structural Classification and Regression Trees

- S-CART: algoritmo que aprende una teoría para la predicción de clases discretas o valones numéricos a partir de un conjunto de ejemplos y conocimiento del dominio relacional.
- Pasos del algoritmo:
  - Construir el árbol
  - Podar el árbol
  - Añadir modelos de regresión lineal

Método Divide y Vencerás Para Construir Árboles en Lógica Proposicional

---

DivideYConquistarás(Ejemplo)

si CondiciónFin(Ejemplos) entonces
  nuevaHoja= CrearNuevaHoja(ejemplos)
  devolver nuevaHoja
si no hacer
  mejorAtributo=EncontrarMejorAtributo(ejemplos)
  particiones = PartirEjemplos(mejorAtributo, ejemplos)
  subárboles = []
  para cada partición $\in$ particiones hacer
    subárbol$_i$=DivideYConquistarás(partición$_i$)
    subárboles=[subárbol$_i$|subárboles]
  Devolver [mejorAtributo|subárboles]

Método Divide y Vencerás Para Construir Árboles en Lógica de Primer Orden

DivideYConquistarás(testsAnteriores,ejemplos)

si CondiciónFin(Ejemplos) entonces
  nuevaHoja= CrearNuevaHoja(ejemplos)
  devolver nuevaHoja
si no hacer
  mejorTest=EncontrarMejorTest(testsAnteriores,ejemplos)
  $(\text{partición}_1, \text{partición}_2)$ = PartirEjemplos(mejorTest, ejemplos, testsAnteriore
  subárbol_1=DivideYConquistarás(testsAnteriores $\wedge$ mejorTest, partición_i)
  subárbol_2=DivideYConquistarás(testsAnteriores, partición_2)
  Devolver [mejorTest, subárbol_1, subárbol_2]

Diferencias entre árboles Relacionales y Proposicionales

- Los tests no son simples comprobaciones de atributos, sino literales o conjunciones de literales que contienen variables.
- Dos tests en una rama deben contener variables en común, por lo que, en cada punto, el test que se puede hacer dependen de los tests anteriores
- Sólo se propagan los test positivos
- Sólo decisiones binarias

Cálculo de Literales o Conjunción de Literales en cada Nodo

- ejemplos: conjunto de ejemplos en un nodo
- testsAnteriores: conjunción de todos los tests anteriores positivos en el camino desde la raíz hasta el nodo actual.
- Asumimos que tenemos un conjunto de posibles literales o refinamiento para el nodo actual, calculados con la función refs(testsAnteriores)
- Para cada refinamiento $ref_i \in$ refs(testsAnteriores), se evalúa teniendo en cuenta como particiona el conjunto de ejemplos:
  - Clasificación: función de la frecuencia relativa de cada clase
  - Regresión: función del error cuadrático medio

Sesgo Declarativo del Lenguaje

- La función *refs*(*testsAnteriores*) se genera teniendo en cuenta sesgos declarativos del lenguaje
- El diseñador define cómo se pueden insertar literales mediante esquemas
- Ejemplo:

  esquema ((enlace(V,W,X,Y), átomo(V, X, Z))
  [V: químico:'+', W:id_átomo:'+', X:id_átomo:'+',
  Y:tipo_enlace:'-', Z:elemento:=]).

- En la primera lista aparecen los predicados que se pueden añadir (sólos o como conjunción de varios de ellos)
- Para cada variable se define el tipo y el modo
- Modos:
  - '+': La variable debe ser unificada con una variable ya existente
  - '-': La variable puede no estar unificada
  - '=': Se debe incluir una constante

Introducción

- Los métodos basados en una medida de distancia asumen que es posible calcular, para cada par de objetos en el dominio, su distancia mutua (o similitud)
- Posibilidades:
  - Aprendizaje predictivo
    - Almacenar todos los ejemplos disponibles
    - Dado un nuevo ejemplo no clasificado, predecir el valor asociado buscando el vecino más cercano al objeto consulta, es decir, el objeto ejemplo que tiene la menor distancia al objeto consulta.
  - Agrupación o clustering:
    - Agrupar un conjunto de instancias, $\mathcal{I}$, en diferentes subconjuntos, grupos o *clusters*, de forma que los objetos que pertenecen a un mismo grupo maximicen la función de similitud entre ellos a la vez que minimizan su similitud respecto a las instancias de los otros grupos
    - Estrategias heurísticas: k-medias y métodos aglomerativos
- Medida de Distancia para predicados en lógica de primer orden

Ejemplo

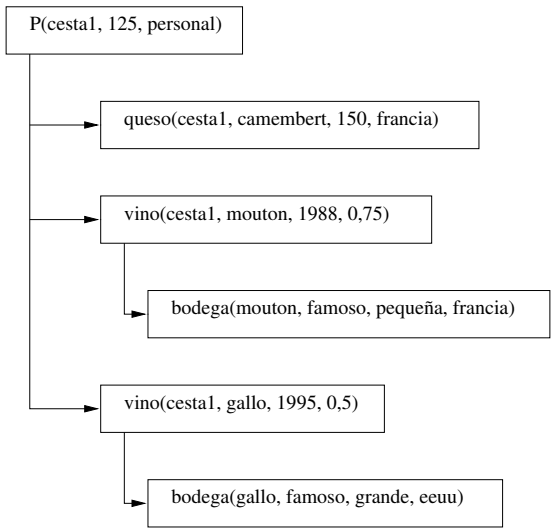
- Describir cestas de navidad de distintos vendedores en lenguaje relacional:
- Vocabulario:
  - P(a_1: nombre, a_2: número, a_3: discreto)
    %(identificador, precio, modo entrega)
  - queso(a_1: nombre, a_2: discreto, a_3: número, a_4: discreto)
    % (identificador, tipo de queso, peso, origen)
  - vino(a_1: nombre, a_2: discreto, a_3: número, a_4:número)
    % (identificador, tipo de vino, año, tamaño)
  - bodega(a_1: nombre, a_2: discreto, a_3: discreto, a_4: discreto)
    %(identificador, popularidad, tamaño, origen)
- Tipos: nombre, discreto, número, lista, término
- El tipo nombre se utiliza para identificadores (claves de una base de datos relacional)

Ejemplo

- Una instancia se compone de la instancia en sí con conocimiento de fondo (*background knowledge*):
  - I=P(cesta1,125,personal)
  - queso(cesta1,camembert,150,francia)
  - vino(cesta1,mouton, 1988, 0.75)
  - vino(cesta1,galla, 1995, 0.5)
  - bodega(gallo, famosa, grande, eeuu)
  - bodega(mouton, famosa, pequeña, francia)

Definición de Casos de Instancias

- El caso de una instancia $\mathcal{I}$, dado un conocimiento de fondo $\mathcal{B}$ y un límite de profundidad (entero $d \geq 0$) es el conjunto más grande $\{\mathcal{I}\} \cup \mathcal{B}'$, tal que $\mathcal{B}' \subseteq \mathcal{B}$, y para cada literal atómico de fondo $B \in \mathcal{B}'$ se cumple que hay un $d' < d$ y $B_0, B_1, \ldots, B_{d'}$ que satisfacen:
  - $B_0 = I$, $B_{d'} = B$, y $B_i \in \mathcal{B}'$ para $i = 1, \ldots, d' - 1$.
  - B_i y B_{i+1} contienen almenos un identificador común



Definición de la Distancia entre dos Instancias

- Pasos en el cálculo de la distancia entre dos instancias I_1 e I_2:
  - Calcular sus correspondientes casos de instancia
  - Calcular recursivamente la distancia entre dos instancias:

$$dist(\mathcal{I}_i, \mathcal{I}_j, \mathcal{B}, d) = \frac{1}{n} \sum_{k=1}^{n} dist(t_k(\mathcal{I}_i), t_k(\mathcal{I}_j))$$

- Donde:
  - Distancia entre números: distancia normalizada entre 0 y 1 (o distancia equivalente)
  - Distancia entre valores discretos: 1 si son iguales, 0 si son distintos (o distancia equivalente)
  - Distancia entre términos y listas: número de pasos que necesito para convertir el primer elemento en el segundo, o viceversa, dada una serie de operadores como insertar, borrar, etc.

Definición de la Distancia entre dos Instancias

Y donde:

- Distancia entre nombres:
  - Si la recursión ha alcanzado la profundidad máxima, compararlos como si fueran tipos discretos
  - Si no:
    - Coleccionar todos los hechos del caso que incluyan el nombre de los dos objetos, generando dos conjuntos de predicados
    - Cada conjunto anterior es dividido en predicados
    - Para cada predicado en los dos conjuntos, elegimos aquel conjunto con menos elementos. Calculamos recursivamente la distancia mínima entre esos predicados con los predicados del conjunto mayor, devolviendo ese valor.

Ejemplo de Cálculo de la Distancia

- Instancias:
  - I_1=P(cesta1,125,personal)
  - I_2 =P(cesta25,195,mail)
- Conocimiento de fondo:
  - queso(cesta1,camembert,150,francia)
  - queso(cesta25,roquefort,200,francia)
  - queso(cesta25,ricotta,100,italia)
  - vino(cesta1,mouton, 1988, 0.75)
  - vino(cesta1,galla, 1995, 0.5)
  - vino(cesta25, mouton, 1995, 0.75)
  - bodega(gallo, famosa, grande, eeuu)
  - bodega(mouton, famosa, pequeña, francia)
- $d = 2$

Ejemplo de Cálculo de la Distancia

$$dist(I_1, I_2, \mathcal{B}, 2) =$$
$$\frac{1}{3}(dist(cesta1, cesta25) + dist(125, 195) + dist(personal, correo))$$

$$dist(125, 195) = \frac{|195 - 125|}{500}$$

$$dist(personal, correo) = 1$$

$$dist(cesta1, cesta25) = ?$$

Ejemplo de Cálculo de la Distancia

- Generar los conjuntos de predicados que continen *cesta*1 y *cesta*2:
  - $L_{cesta1,queso} = \{queso(cesta1, camembert, 150, francia)\}$
  - $L_{cesta1,vino} = \{vino(cesta1, mouton, 1988, 0{,}75), vino(cesta1, galla, 1995, 0{,}5)\}$
  - $L_{cesta25,queso} = \{queso(cesta25, roquefort, 200, francia), queso(cesta25, ricotta, 100, ita$
  - $L_{cesta25,vino} = \{vino(cesta25, mouton, 1995, 0{,}75)\}$

$$D_{queso} = \frac{\min_{l \in L_{cesta25,queso}} dist(queso(cesta1,camembert,150,francia), l)}{|L_{cesta25,queso}|}$$

$$\frac{min((1+\frac{|150-200|}{300}+0)/3, 1+\frac{|150-100|}{300}+1/3)}{2} = 0{,}1944$$

$$dist(mouton, gallo) = \frac{min(\frac{dist(famoso,famoso)+dist(pequena,grande)+dist(francia,eeuu)}{3})}{1} =$$

$$0{,}666$$

$$D_{vino} = \ldots = 0{,}0117$$

$$dist(cesta1, cesta25) = \frac{D_{queso}+D_{vino}}{2} = 0{,}1031$$

Algoritmos

- KNN Relacional: equivalente a KNN proposicional, pero con la nueva medida de distancia
- K-medias relacional: también equivalente excepto en el cálculo del centroide: nos quedamos con la instancia que minimice su distancia media al resto de las instancias del clúster

Motivación

- ¿Por qué usar representaciones relacionales?
  - Los métodos de aproximación de funciones clásicos no generalizan bien cuando se aplican a dominios relacionales (requieren un mapeo de representación relacional a proposicional)
  - El aprendizaje por refuerzo clásico no permite generalizar entre objetos (la meta $on(a, b)$, podría ser genenarlizada a $on(A, B)$)
  - El aprendizaje por refuerzo clásico limita la generalización entre tareas (la tarea $on(a, b)$, puede ser útil para resolver la tarea $on(a, b) \wedge on(b, c)$)
  - La mayoría de métodos de aprendizaje por refuerzo clásico no permite planificación deliberativa ni razonamiento en tiempo real (nuevo conocimiento podría ser añadido al proceso de prueba y error para alcanzar una mejor aproximación de la política óptima)
  - El aprendizaje por refuerzo clásico resta importancia al papel del conocimiento previo durante el aprendizaje

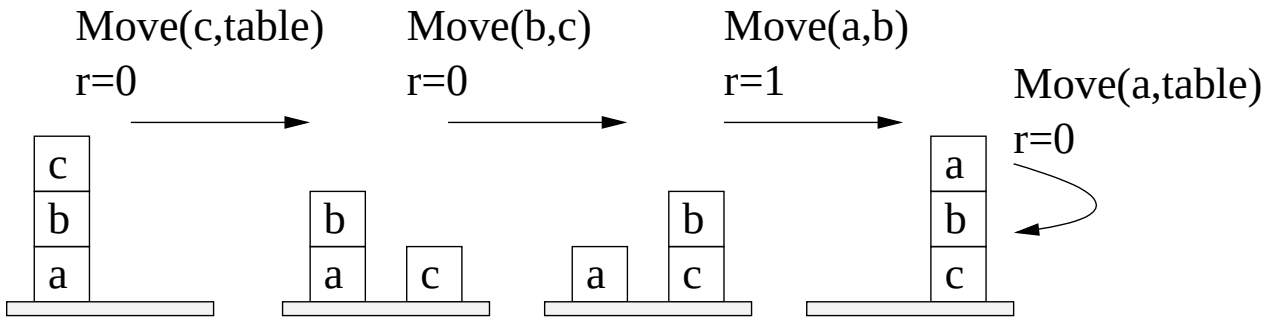
Componentes de la tarea de Aprendizaje por Refuerzo Relacional

- Un conjunto de posibles estados, descritos en lenguaje relacional
  - Un estado se representa con un conjunto de hechos que son ciertos en el estado actual (on(a,b), clear(a), on(b,table))
  - Mundo cerrado
  - Los posibles estados se descubren uno a uno, y no pueden ser enumerados a priori
  - El estado incluye la meta (goal_on(a,b))
- Un conjunto de posibles acciones:
  - Una acción también se representa con lenguaje relacional (move(a,b), move(a,table))
  - El algoritmo de RRL sólo ve las acciones aplicables en cada estado, según especifica la función $pre : S \times A \rightarrow \{true, false\}$
- Una función de transición de estados desconocida

Ideas básicas de RRL (Dzeroski, De Raedt and Driessens, 2001)

- Ejecutar Batch Q-Learning del mismo modo que en RL proposicional
- Utilizar:
  - Un método de regresión para aproximar la función Q (y por tanto, la política), capaz de aprender de la representación de estados y acciones relacional: Árboles de Decisión Lógicos (TILDE y TILDE-RT).
  - Se dispone de información del dominio (above(a,b)).
  - Un sesgo declarativo para el aprendizaje relacional de las políticas y la función Q. Por ejemplo, no permitir que la función Q aprendida se refiera a instancias concretas, sino que generalice: Aprender términos como on(A,B), y no on(a,b)

Traza de ejecución



- Ex. 1: (on(a,table), on(b,a), on(c,a), clear(c), goal_on(a,b), move(c,table),qvalue(0))
- Ex. 2: (on(a,table), on(b,a), on(c,table), clear(c), goal_on(a,b), move(b,c),qvalue(0))
- Ex. 3:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,b),qvalue(1))
- Ex 4:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,table),qvalue(0))

Propagando Refuerzos

- Iter. 1:
  - Ex. 1: (on(a,table), on(b,a), on(c,a), clear(c), goal_on(a,b), move(c,table),qvalue(0))
  - Ex. 2: (on(a,table), on(b,a), on(c,table), clear(c), goal_on(a,b), move(b,c),qvalue(0))
  - Ex. 3:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,b),qvalue(1))
  - Ex. 4:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,table),qvalue(0))

Propagando Refuerzos

- Iter. 2:
  - Ex. 1: (on(a,table), on(b,a), on(c,a), clear(c), goal_on(a,b), move(c,table),qvalue(0))
  - Ex. 2: (on(a,table), on(b,a), on(c,table), clear(c), goal_on(a,b), move(b,c),qvalue(0.9))
  - Ex. 3:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,b),qvalue(1))
  - Ex. 4:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,table),qvalue(0))

Propagando Refuerzos

- Iter. 3:
  - Ex. 1: (on(a,table), on(b,a), on(c,a), clear(c), goal_on(a,b), move(c,table),qvalue(0.81))
  - Ex. 2: (on(a,table), on(b,a), on(c,table), clear(c), goal_on(a,b), move(b,c),qvalue(0.9))
  - Ex. 3:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,b),qvalue(1))
  - Ex. 4:(on(a,table), on(b,c), on(c,table), clear(b), goal_on(a,b), move(a,table),qvalue(0))

Árbol de Decisión Lógico

root: goal_on(A,B), move(D,E)
on(A,B)?
+ − −yes: [0]
+ − −no: clear(A)?
+ − −yes: [1]
+ − −no: clear(E)?
+ − −yes: [0.9]
+ − −no: [0.81]

- Consulta:
  - (on(a,table), on(b,a), on(c,table), clear(c), goal_on(a,b), move(b,c))
- Unificación: A/a, B/b, D/b, E/c
- Resultado: 0.9

Obtención de Invariantes de bucle

- Método de ILP: Claudien (Raedt y Dehaspe, 1997)
- Invariantes de bucle: relaciones entre los valores de las variables que permanecen invariantes durante la ejecución del bucle
- Claudien aprende estas especificaciones a partir de trazas de ejecución.

Obtención de Invariantes de bucle

- Algoritmo de multiplicación:

function Product
    inputs: x, y: positive integers
    outputs: z: the product of x and y

$z := 0$; $u := x$; $v := y$;
while $\dagger(u \neq 0)$ do
    if $odd(u)$ then $z := z + v$;
    $u := u \div 2$;
    $v := 2 * v$;
endwhile
return z
endfunction

Obtención de Invariantes de bucle

- Observaciones:

Observación 1	Observación 2	Observación 3
$input(x(0), y(0))$	$input(x(7), y(6))$	$input(x(9), y(10))$
$trace(z(0), u(0), v(0))$	$trace(z(0), u(7), v(6))$	$trace(z(0), u(9), v(10))$
	$trace(z(6), u(3), v(12))$	$trace(z(10), u(4), v(20))$
	$trace(z(18), u(1), v(24))$	$trace(z(10), u(2), v(40))$
	$trace(z(42), u(0), v(48))$	$trace(z(10), u(1), v(80))$
		$trace(z(90), u(0), v(160))$

- Relaciones invariantes aprendidas en el punto †:

  U≥0 ← input (x(X), y(Y)), trace(z(Z), u(U), v(V))
  Term = XY ← input(x(X), y(Y)), trace(z(Z), u(U), v(V)),
  XY is X*Y, _Term is Z+U*V

- Equivale a: $((z + u * v) = x * y) \wedge (u \geq 0)$

Predicción de la Estructura de una Proteina y su Función

- Predecir la forma tridimensional de una proteina a partir de su secuencia de aminoácidos es uno de los grandes problemas de la biologíca molecular.
- Una proteina es básicamente una secuencia de aminoácidos
- Dicha secuencia es la estructura primaria. La forma tridimensional es su estructura secundaria
- Una versión limitada de este problema es predecir si la forma de la proteina es una α-hélice
- El sistema GOLEM utiliza información de la secuencia de aminoácidos para predecir si tiene forma de α-hélice.
- Las reglas inducidas obtenían un acierto el 81 % contra un 76 % obtenidos con otros métodos.

Otras Aplicaciones

- Aplicaciones médicas: diagnosis de reuma, glaucoma, modelado de efectos de medicamentos, predicción de cáncer, etc.
- Aplicaciones farmacéuticas: diseño de fármacos
- Aplicaciones medioambientales: predicción de tasas de biodegradabilidad, clasificación de rios según la calidad de sus aguas, etc.
- Aplicaciones en ingeniería mecánica
- Análisis de textos, de WEB y procesamiento de lenguaje natural
- Análisis de datos de negocio
- Etc.

Resumen

- Métodos inductivos basados en la representación atributo-valor pueden no ser útiles para todos los problemas
- Representación relacional de los datos
- Foil: ganancia de literales y lenguaje de cláusulas
- Sesgo en el aprendizaje
- Métodos de clasificación y agrupación relacional
- Medida de distancia relacional recursiva
- Medida de distancia definida por el diseñador
- Sesgo en el aprendizaje derivado de la medida de distancia

Bibliografía

- Machine Learning, Tom Mitchell. Capítulo 10.
- Relational Data Mining, Saso Dzeroski y Nada Lavrac (editores). Springer, 2001. Capítulos 6 y 9.
- Clausal Discovery. Luc de Raedt, Luc Dehaspe. Machine Learning 26, 99-146. 1997