Departamento de Informática
Universidad Carlos III de Madrid

Ingeniería en Informática

Aprendizaje Automático

Septiembre 2006

Normas generales del examen

- El tiempo para realizar el examen es de **3 horas**
- No se responderá a ninguna pregunta sobre el examen
- Si se sale del aula, no se podrá volver a entrar durante el examen
- No se puede presentar el examen escrito a lápiz

Pregunta 1/3 (3 puntos)

Se quiere implementar un agente que sea capaz de jugar al juego de "Conecta-4". Este juego se muestra en la figura 1. El objetivo es conectar 4 fichas del mismo color (azul o roja) en vertical, horizontal o diagonal, antes de que lo haga el oponente.

Figura 1: Imagen de una situación en el juego de Conecta 4

En este ejercicio, se pide:

1. Definir qué técnica de Aprendizaje Automático es la más adecuada para resolver este problema, y razonarlo.
2. Realizar el diseño en detalle de todos los elementos requeridos para implementar el sistema mediante el sistema de aprendizaje escogido
3. Definir cómo se realizaría el proceso de aprendizaje
4. Definir cómo se realizaría el proceso de test
5. Definir aquellos elementos del proceso de aprendizaje que pudieran ser problemáticos, y plantear distintas alternativas.

Pregunta 2/3 (4 puntos)

Se dispone de datos sobre vivienda en una ciudad, tal y como se muestran en la Tabla 1:

A partir de dicha base de datos, se desea construir un árbol de decisión que modele cuándo un piso pertenece a una categoría A o a una categoría B. Para ello, se pide:

ID	Dormitorios	Baños	Tipo	Chimenea	Zona	Categoría
1	1	2	piso	no	centro	A
2	1	1	adosado	si	periferia	B
3	1	1	piso	no	centro	B
4	2	2	piso	no	centro	A
5	2	2	adosado	si	periferia	B
6	2	1	adosado	si	periferia	B
7	3	1	piso	no	centro	B
8	3	2	adosado	si	centro	A
9	2	1	adosado	si	periferia	B
10	1	2	piso	no	centro	A
11	3	1	piso	no	centro	B
12	3	2	piso	no	periferia	B

Cuadro 1: Conjunto de Datos

1. Representar el conjunto de entrenamiento, definiendo si hay atributos innecesarios.
2. Definir qué algoritmo de aprendizaje se puede utilizar para generar el árbol de decisión, y seguirlo para generar el árbol.
3. Generar la matriz de confusión para el conjunto de datos y el árbol de decisión obtenido.
4. Dado el árbol de decisión generado, aplicamos un método para obtención de reglas de decisión. ¿Cuántas reglas obtenemos? ¿Cuáles son?

Pregunta 3/3 (3 puntos)

Muchos métodos de análisis de datos se basan en una medida de distancia entre las distintas instancias. Dado el conjunto de datos de la tabla 2:

Instancia	A1	A2
1	1,5	87
2	3,1	88,5
3	6	67
4	1,7	88
5	9	33
6	9,1	34,7
7	5	71
8	8,1	41,2
9	9,2	35,4
10	2,5	90
11	5	59
12	5	61
13	3	98
14	8,3	37,2
15	4	62

Cuadro 2: Conjunto de Datos

1. Definir una medida de distancia adecuada para realizar agrupación o *clustering* sobre el conjunto de datos. ¿Sería útil aplicar algún tipo de preprocesado sobre los datos?
2. Definir un número de grupos o *clusters* adecuado para este conjunto de datos, y razonar la respuesta.
3. ¿Qué algoritmo se puede utilizar para realizar agrupación sobre este conjunto de datos? Describir cuál sería la salida de dicho algoritmo sobre este conjunto de datos, teniendo en cuenta el número de grupos definido anteriormente.

Pregunta 1/3 (3 puntos)

Este juego consiste en la ejecución de una secuencia de acciones. Por tanto, para poder jugar, es necesario obtener una *Política de Acción*, que para cada estado del juego, indique cuál es la mejor acción a ejecutar. Es, además, un problema de prueba y error, puesto que no se sabe a priori qué política es la más adecuada, sino que se decide en función de un refuerzo (si se gana o no la partida)Por tanto, la técnica más adecuada es el Aprendizaje por Refuerzo. Además, debe usarse un algoritmo de aprendizaje por refuerzo directo (como Q-Learning), ya que la función de transición de estados es desconocida. Por último, cabe destacar que el problema cumple la propiedad de Markov, necesaria para modelar procesos de aprendizaje por refuerzo.

Los elementos de cualquier problema de Aprendizaje por Refuerzo son los siguientes:

- Espacio de Estados: conjunto de todas las posibles combinaciones de fichas en el tablero. Esto daría un tamaño del espacio de estados de 3^{48}. Sin embargo, no todos estos estados son válidos, por lo que el espacio es bastante más reducido. Por ejemplo, siempre hay tantas fichas de un color como del otro más/menos 1, una casilla sólo puede estar desocupada si las que hay por encima también lo están, etc.
- Espacio de acciones: se limita a 8 acciones, una por cada posible columna por donde se puede echar una ficha.
- Función de transición de estados: viene dada por las reglas del juego y por los movimientos del jugador contrario. Dado que estos movimientos dependen del jugador contrario, la función de transición es estocástica.
- Función de refuerzo: se puede dar un valor positivo cuando se gana una partida (+1), un valor negativo cuando se pierde (-1), y un valor nulo si la partida continua o acaba en tablas. Es por tanto, un sistema de refuerzo retardado en el tiempo.

Dado este diseño, habría que comenzar el aprendizaje de la función Q. Para ello, se ejecutaría un número alto de episodios de aprendizaje, consistentes en jugar muchas partidas. Siguiendo ese proceso, se generan tuplas de experiencia que permiten actualizar la función Q siguiendo el algoritmo Q-Learning para dominios estocásticos. El proceso se repite hasta que se converja a una política. Para ese proceso se debe elegir una estrategia de exploración adecuada.

Posteriormente, se debe validar la política obtenida siguiendo un proceso equivalente, en el que se sigue la política de forma totalmente avariciosa para obtener las acciones. Repitiendo un conjunto de test suficientemente grande, se puede evaluar la calidad de la política obtenida, teniendo en cuenta el número de veces que se ha conseguido ganar o empatar la partida.

El principal inconveniente de la aplicación del algoritmo Q-Learning a este problema es cómo representar la función Q. Una aproximación tabular podría ser intratable, puesto que el espacio de estados es demasiado grande, tal y como se introdujo anteriormente. Las tres posibles soluciones pasan por una discretización de los espacios, por utilizar un aproximador de funciones, o por utilizar conocimiento del dominio para redefinir el espacio de estados.

Pregunta 2/3 (4 puntos)

El conjunto de entrenamiento, en formato ARFF, se muestra a continuación:

```
@RELATION viviendas

@ATTRIBUTE dormitorios    {1,2,3}
@ATTRIBUTE banios  {1,2}
@ATTRIBUTE tipo         {piso,adosado}
@ATTRIBUTE zona         {centro,periferia}
@ATTRIBUTE categoria {A,B}


@DATA
1,2,piso,centro,A
1,1,adosado,periferia,B
1,1,piso,centro, B
2,2,piso,centro,A
2,2,adosado,periferia,B
2,1,adosado,periferia,B
3,1,piso,centro,B
3,2,adosado,centro,A
2,1,adosado,periferia,B
1,2,piso,centro,A
3,1,piso,centro,B
3,2,piso,periferia,B
```

En esta descripción, se ha eliminado ya el atributo de identificador, que es irrelevante ya que es distinto para cada ejemplo. También se puede eliminar el atributo *Chimenea*, ya que en el conjunto de datos se demuestra que ese atributo puede derivarse del atributo *Tipo* fácilmente.

El algoritmo ID3 es suficiente para generar un árbol de decisión, si se consideran los valores numéricos de los atributos *Baños* y *Dormitorios* como etiquetas, y no como valores numéricos. El árbol de decisión que genera ID3 es el siguiente:

```
baños = 1: B
baños = 2
|  zona = centro: A
|  zona = periferia: B
```

Para derivar este árbol de conocimiento, se puede seguir el siguiente razonamiento. Primero, se comprueba cómo divide cada atributo el conjunto de entrenamiento. Eso se muestra en la figura 2. En ella se comprueba que los atributos *Baños* y *Zona* son los que mejor discriminan, ya en una de sus ramas dejan sólo instancias de una clase, mientras que los otros atributos dejan instancias de ambas clases en todas sus ramas. Eso nos da una idea de que las ganancias de esos dos atributos serán las mayores. Calculamos dichas ganancias, a modo de ejemplo:

Para el atributo *Baños*:

$I_{Banos} = \frac{6}{12} I_{Banos,1} + \frac{6}{12} I_{Banos,2} = \frac{6}{12} 0{,}91 = 0{,}46$
$I_{Banos,1} = 0$;Dado que todas las instancias en este grupo pertenecen a la misma clase
$I_{Banos,2} = -(\frac{4}{6} \log_2 \frac{4}{6}) - \frac{2}{6} \log_2 \frac{2}{6} = 0{,}91$

Para el atributo *Zona*:

$I_{Zona} = \frac{7}{12} I_{Zona,centro} + \frac{5}{12} I_{Zona,periferia} = \frac{7}{12} 0{,}98 = 0{,}57$
$I_{Zona,periferia} = 0$; Dado que todas las instancias en este grupo pertenecen a la misma clase
$I_{Zona,centro} = -(\frac{4}{7} \log_2 \frac{4}{7}) - \frac{3}{7} \log_2 \frac{3}{7} = 0{,}98$

Se observa, por tanto, que el atributo *Baños* tiene menor entropía, por lo que su ganancia es mayor, y es elegido para partir el árbol. Dado que una de sus ramas (*Baños=1*) sólo contiene instancias de la misma clase (B), en esa rama se termina. En la otra rama, se genera el conjunto de datos mostrado en la tabla 3 (del que eliminamos el atributo *Baños*).

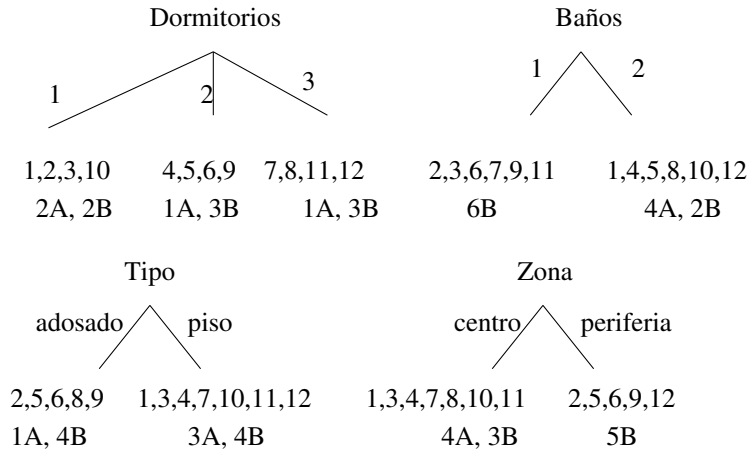

Figura 2: Particiones del conjunto de entrenamiento según el atributo elegido

ID	Dormitorios	Tipo	Zona	Categoría
1	1	piso	centro	A
4	2	piso	centro	A
5	2	adosado	periferia	B
8	3	adosado	centro	A
10	1	piso	centro	A
12	3	piso	periferia	B

Cuadro 3: Conjunto de Datos

En el nuevo conjunto de datos, se observa que el altributo *Zona* divide perfectamente el conjunto de datos. Además, es el único que cumple esta propiedad, por lo que es de máxima ganancia, y sería elegido, terminando la ejecución del algoritmo.

La matriz de confusión que genera el anterior árbol de decisión es:

```
 a b   <-- classified as
 4 0 | a = A
 0 8 | b = B
```

Por último, dado que en el árbol de decisión hay tres hojas, se generarían tres reglas (en formato WEKA):

```
banios = 1: B (6.0)
zona = centro: A (4.0)
: B (2.0)
```

Solución Pregunta 3/3 (3 puntos)

Para realizar clustering, sería necesario utilizar una medida de distancia que tenga en cuenta la distancia individual entre cada par de atributos, como la distancia euclídea:

$$d(\vec{x}_i, \vec{x}_j) = \sqrt{\sum_{r=1}^{n} (\vec{x}_i[r] - \vec{x}_j[r])^2} \tag{1}$$

Sin embargo hay que tener en cuenta que el rango de los distintos atributos es muy diferente. En este caso, convendría utilizar una medida de distancia ponderada.

$$d(\vec{x}_i, \vec{x}_j, \vec{w}) = \sqrt{\sum_{r=1}^{n} \vec{w}[r](\vec{x}_i[r] - \vec{x}_j[r])^2} \tag{2}$$

Un posible vector de pesos podría ser $\vec{w} = (1, 1/10)$. Otra opción más cómoda en este caso sería normalizar los atributos para que todos sus valores estén en el rango $[0, 1]$. A simple vista, se podría divir el atributo 1 entre 10, y el atributo 2 entre 100.

Para definir el número de grupos, se puede hacer una inspección visual de los datos. La figura 3 muestra los datos normalizados en el espacio de dos dimensiones.

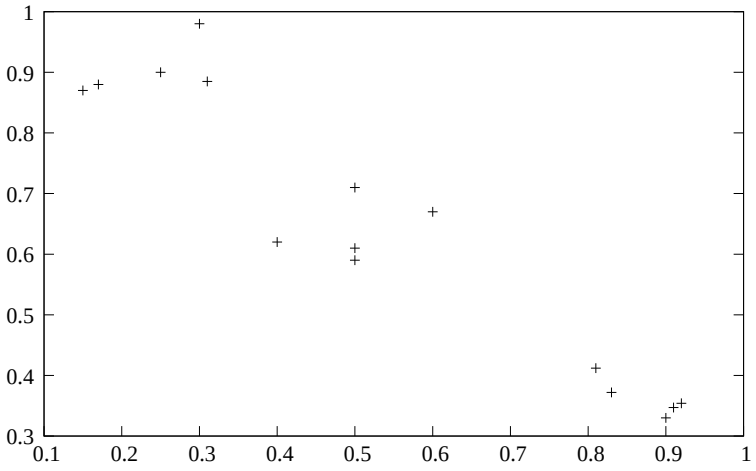

Figura 3: Conjunto de datos en el espacio euclídeo.

Según se ve, tres grupos sería un número adecuado, ya que haría pequeña la distancia intra-grupo, y muy alta la inter-grupo. Para conseguir estos grupos, podría utilizarse el algoritmo *k-medias*. Este algoritmo devolvería tres prototipos, que serían los centroides de cada uno de los tres grupos. Para calcular estos centroides, habría que definir qué instancia pertenecen a cada grupo. Una vez calculados los tres grupos de instancias, se calcularía la media entre las instancias de cada grupo, obteniendo el centroide, obteniendo los resultados que se muestran a continuación.

Centroides que devolvería el algoritmo *k-medias*:

- Centroide del grupo 1: $< 0'182, 0'903 >$
- Centroide del grupo 2: $< 0'5, 0'64 >$
- Centroide del grupo 3: $< 0'874, 0'363 >$

Instancia	A1	A2
1	0,15	0,87
2	0,31	0,885
4	0,17	0,88
10	0,25	0,90
13	0,03	0,98

Cuadro 4: Conjunto de Datos del Grupo 1

Instancia	A1	A2
3	0,6	0,67
7	0,5	0,71
11	0,5	0,59
12	0,5	0,61
15	0,4	0,62

Cuadro 5: Conjunto de Datos del grupo 2

Instancia	A1	A2
5	0,9	0,33
6	0,91	0,347
8	0,81	0,412
9	0,92	0,354
14	0,83	0,372

Cuadro 6: Conjunto de Datos del grupo 3