Departamento de Informática
Universidad Carlos III de Madrid

Ingeniería en Informática

Aprendizaje Automático

Junio 2008

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 (4 puntos)

Se dispone de una versión del juego del tetris que tiene las siguientes características:

- un tablero de 10 filas y 6 columnas,
- dos tipos de fichas, una de tamaño 1×3, y otra de tamaño 2×2
- las fichas se pueden rotar, desplazar una posición a la izquierda, una posición a la derecha, dejar caer. Para la rotación, tomar como centro de rotación la casilla superior izquierda de la ficha.
- Una vez que cae una ficha, la siguiente aparece en el tablero, pero no se conoce con anterioridad qué ficha va a salir, ni en qué posición del tablero. Además, algunas veces y con una frecuencia desconocida, hay casillas del tablero que se ocupan por fichas de tamaño 1×1 que no caen.
- El tiempo en el juego es discreto. En cada instante de tiempo se puede ejecutar una única acción, y la ficha desciende una posición para todas las acciones (excepto la de caer).
- El tiempo de cada partida es ilimitado, y la partida acaba sólo cuando una una nueva ficha que sale no tiene hueco para colocarse.
- El resto de reglas son las del juego del tetris tradicional. Si alguna se desconoce, asumir lo que se considere más oportuno.

Cuestión 1

Describir el juego del tetris como un problema de aprendizaje por refuerzo. Para ello, definir:

1. El espacio de estados
   - Describir el espacio de estados.
   - ¿Cuál es el tamaño del espacio de estados?
   - ¿Es posible encontrar versiones del espacio de estados más reducidas? En caso afirmativo, pon un ejemplo. Describir si con esta nueva representación se asegura que las políticas que se encuentren sean óptimas o si, por el contrario, no se asegura optimalidad.
2. El espacio de acciones
   - Describir el espacio de acciones.
   - ¿Cuál es el tamaño del espacio de acciones?

3. La función de transición de estados
    - Definir, informalmente, la función de transición de estados
    - ¿Es la función de transición de estados determinista o estocástica?
4. La función de refuerzo
    - Describir formalmente la función de refuerzo
    - ¿Es la función de refuerzo determinista o estocástica?

Cuestión 2

Una vez descritos los elementos anteriores, justificar si es posible utilizar alguno de los siguientes métodos de aprendizaje por refuerzo para aprender un agente que juegue al tetris:

- Programación Dinámica
- Métodos basados en el modelo
- Métodos libres de modelo

Cuestión 3

Dada la respuesta anterior, elegir un algoritmo para aprender un jugador en el juego del tetris. Describir cómo sería el proceso de aprendizaje, definiendo:

- Cómo implementarías la función de valor acción o de valor acción-estado (dependiendo del algoritmo utilizado)
- Cómo se desarrollaría el proceso de aprendizaje.
- Cómo evaluarías lo bueno o malo que es el jugador aprendido.

Pregunta 2/3 (4 puntos)

Se dispone de cierta información sobre varios mercados empresariales, y sobre las empresas que compiten en dichos mercados. En concreto, la información de la que se dispone está contenida en una base de datos que contiene las siguientes tabla:

1. Sobre las empresas:
    - Identificador (ID-E): tipo entero
    - Capital (C): tipo entero, en millones de euros
    - Nivel tecnológico (NT): puede ser bajo, medio, o alto
    - Gastos publicidad (GP): puede ser bajo, medio, o alto
2. Sobre los productos:
    - Identificador (ID-P): tipo entero
    - Nombre de venta del producto (NP): una cadena de caracteres
    - Identificador de Mercado (ID-M): entero que identifica el mercado en el que se encuadra este producto
    - Identificador de Empresa (ID-E): entero que identifica la empresa que fabrica este producto
3. Sobre los mercados (todos los parámetros de los mercados se consideran constantes en el tiempo):
    - Identificador (ID-M): entero
    - Nombre del mercado (NM): sólo hay dos mercados: tomates y lechugas
    - IPC: tipo real, indicando el porcentaje medio anual (tanto por ciento) de incremento del precio del producto en los últimos 10 años
    - Nivel competitivo (NC): puede ser muy agresivo, agresivo, normal, suave

4. Valores: indica ciertos valores de los productos
    - Identificador del producto (ID-P)
    - Precio de venta al público (PVP): valor entero (en euros)
    - Coste del producto (C): valor entero (en euros)
    - Trimestre del año (T): primer, segundo, tercer, y cuarto trimestres. Indica a qué trimestre del año corresponden los valores del coste y del trimestre
    - Percepción Cliente (PC): caro, barato, normal. Especifica la percepción que el cliente tuvo sobre si el producto era caro, barato, o tenía un precio normal. Este valor es obtenido a través de encuestas.

En los cuadros 1,2,3 y 4 se muestran algunos de los registros contenidos en la base de datos:

Empresa			
Id-E	C	NT	GP
1	3	alto	bajo
2	2	medio	alto
3	3	alto	bajo
4	2	alto	alto

Cuadro 1: Datos de las empresas

Productos			
Id-P	NP	ID-M	ID-E
1	Tomates Caracol	1	1
2	Tomates de la Huerta	1	2
3	Lechugas Caracol	2	1
4	Lechugas Garrido	2	3

Cuadro 2: Datos de los productos

Mercados			
Id-M	NM	IPC	NC
1	tomates	4	alto
2	lechugas	2	medio

Cuadro 3: Datos de los mercados

A partir de dicha base de datos, se desea analizar si el producto "Lechugas Garrido" que se venderá en el cuarto trimestre a 2,4 euros, y que se fabrica a 1,5 euros, será percibido por los usuarios como un producto caro, generando para ello un modelo con un algoritmo de aprendizaje automático.

1. Representar el conjunto de entrenamiento siguiendo el método de representación más adecuado. Justificar la representación elegida.
2. Definir qué algoritmo de aprendizaje se puede utilizar para obtener una solución, y justificar la respuesta.
3. Describir las entradas del algoritmo elegido, y dar un ejemplo de salida (no es necesario ejecutar el algoritmo: sólo dar un ejemplo de una posible salida, que no tiene que ser óptima).
4. Dada la salida generada en el apartado anterior, ¿sería percibido el producto 4 como un producto caro?

Pregunta 3/3 (2 puntos)

ACME S.A. Tiene en total 500 empleados en una única sucursal. Para cada empleado, dispone de un total de 47 datos de diverso tipo, entre los que hay algunos personales, como el DNI, domicilio, etc; laborales, como su puesto, sueldo, etc; y su perfil profesional: estudios, experiencia, idiomas, etc.

Valores				
ID-P	PVP	C	T	PC
1	2,1	0,9	primer	barato
1	1,7	0,3	segundo	normal
1	1,1	0,2	tercer	caro
1	1,4	0,25	cuarto	caro
2	2,2	0,95	primer	barato
2	1,6	0,2	segundo	caro
2	1,0	0,15	tercer	normal
2	1,6	0,4	cuarto	barato
3	3,1	1,95	primer	caro
3	2,6	1,25	segundo	barato
3	2,4	1,19	tercer	caro
3	2,5	1,4	cuarto	normal
4	2,2	1,5	primer	barato
4	1,6	0,8	segundo	caro
4	1,4	0,7	tercer	barato

Cuadro 4: Valores de los productos

La organización ha decidido romper la estructura de la empresa en 5 sucursales, por lo que quiere distribuir a los empleados entre dichas sucursales. El objetivo es que los empleados que tengan relaciones entre sí, o que tengan características similares, terminen en la misma sucursal. Además, también se espera que las distintas sucursales tengan un número parecido de empleados.

1. Decidir, a grandes rasgos, cómo representar los datos
2. Definir qué algoritmo o algoritmos de aprendizaje se pueden utilizar para obtener una solución, y justificar la respuesta.
3. Una vez definidos el/los algoritmo/s a utilizar, describir de manera informal cómo sería el proceso completo de asignación de empleados a sucursales. Si lo consideras oportuno, pon un ejemplo de cómo se desarrollaría todo el proceso.

Pregunta 1/3 (4 puntos)

Cuestión 1

El tablero del tetris es de 10 filas por 6 columnas, haciendo un total de 60 casillas. Cada una de esas casillas podría estar libre (valor 0) u ocupada por una ficha que ha caido (valor 1). Además, la ficha que está cayendo en ese momento también debería estar representada (valor 2). Por tanto, los estados pueden representarse por un vector de 48 casillas, de forma que cada una de ellas puede tomar el valor 0, 1 o 2.

El tamaño del espacio de estados sería de 3^60. Sin embargo, es fácil ver que este número es mucho mayor que el de los posibles estados que se pueden llegar a dar. En primer lugar, sólo cae una ficha a la vez. En un tablero de 10×6, asumiendo que está vacío, sólo se puede colocar una ficha cuadrada de tamaño 2×2 de $9 \times 5 = 21$ formas distinas. La ficha de tamaño 1×3 sólo se puede poner de $4 \times 10 + 8 \times 6 = 88$ posiciones distintas. Eso hacen un total de 109 posibles formas de poner las dos fichas en el tablero. Por tanto, la cota superior del número posible de estados sería $2^48 \times 109$. Además, hay que tener en cuenta que las fichas caen por la gravedad, por lo que no todas las configuraciones representan estados posibles. Se puede pensar (pero es sólo una aproximación debido, sobre todo, a la aparición de bloques que no caen) que el número de estados reales es justo la mitad, simplemente notando que si una configuración es válida, la simétrica (resultante de cambiar unos por ceros y ceros por unos), no es válida. Por tanto, tendríamos una nueva cota superior del número de estados de $2^59 \times 109$, valor cercano a 10^20. Además, esta cota se podría reducir ligeramente, teniendo en cuenta que no se puede tener configuraciones en las que haya una fila totalmente ocupada. Por tanto, por cada fila, habría que eliminar $2^{9\times6}$) combinaciones, haciendo un total de $2^59 \times 109 - 10 * 2^54$, pero que sigue siendo de un valor de un orden cercano a 10^20.

Una forma de reducir el espacio de estados viene dado por la idea de que, para decidir qué ficha colocar en un determinado momento, sólo es necesario mantener información de las últimas filas que tienen bloques puestos, y de la posición relativa de la ficha que está cayendo sobre estas fichas. Por ejemplo, si sólo mantenemos información de las 2 últimas filas, entonces sólo tenemos 12 casillas, que pueden tomar el valor 0 o 1, haciendo un total de 2^12. Estas casilla pueden estar como mucho a 8 filas de la nueva ficha, y habría que indicar en qué columna está la ficha de las 6 posibles, y de qué ficha se trata de las dos posibles. Por tanto, hay $2^12 \times 8 \times 6 \times 2 = 393216$ estados. Si quisiéramos reducir todavía más el espacio de estados, podríamos usar sólamente la última fila, por lo que pasaríamos a $2^6 \times 9 \times 6 \times 2 = 6912$ estados.

Obviamente, estas soluciones pasar por asumir que la política a obtener podría ser subóptima, puesto que estamos perdiendo la propiedad de Markov. Esto es debido a que, si sólo tenemos información sobre la última fila donde hay fichas, la posición en la que debemos poner la siguiente ficha depende no sólo del estado de esa fila, sino también de las filas que hay por debajo y de las cuales no tenemos información, o lo que es lo mismo, de cómo y dónde pusimos las fichas en el pasado.

El espacio de acciones se compone de 4 acciones. No todas las acciones se pueden ejecutar en cualquier momento. Por ejemplo, si una ficha está pegada a la izquierda, no se podría ejecutar la acción de desplazar a la izquierda. También se podría permitir esta ejecución, pero entonces la ficha quedaría en la misma columna.

La función de transición de estados viene definida por el juego en sí. Sin embargo, hay que tener en cuenta que es estocástica. Dicha componente estocástica viene dada porque la ficha que sale nueva en cada momento es elegida al azar. Además, dado que de vez en cuando se ocupan casillas, esto aporta otra componente aleatoria.

La función de refuerzo puede venir dada de distintas formas. Dado que la partida sólo acaba cuando ya no hay donde poner fichas, se podría medir el tiempo que ha durado la partida como valor de refuerzo. También se podría dar como valor de refuerzo el número de veces que se ha completado una fila. Para acelerar el aprendizaje, también se podría pensar en dar un refuerzo positivo sólo cada vez que se hace una fila. en cualquiera de los casos anteriores, la función de refuerzo sería estocástica, puesto que la función de transición también lo es. Nótese, sin embargo, que la componente estocástica de la función de transición es tan débil que la función de refuerzo resulta determinista.

Cuestión 2

El algoritmo no se puede resolver directamente con programación dinámica, puesto que la función de transición de estados es desconocida. Por tanto, tendría que aplicarse un método libre de modelo (como Q-Learning) o basado en el modelo, como Dyna-Q.

Cuestión 3

La implementación de la función de refuerzo depende del tamaño del espacio de estados. Si el número de estados es pequeño, como en la versión en la que se tenían menos de 70000 estados, se haría con una tabla. En otro caso, habría que buscar otras aproximaciones, como métodos de discretización, aproximación de funciones, etc.

El proceso de aprendizaje vendría dado por un proceso de prueba y error definido por el algoritmo de aprendizaje elegido. La longitud de dicho proceso de prueba y error deberá ser suficiente como para que todos los pares estado-acción se visiten un número suficientemente grande de veces, y que así la función Q pueda acercarse lo más posible al óptimo. En ese momento, se podría fijar la política, y seguir una estrategia totalmente avariciosa para evaluar la calidad de dicha política.

Pregunta 2/3 (4 puntos)

En este caso, se debería utilizar una representación relacional, con los siguientes predicados:

- empresa(IdentificadorEmpresa, Capital, NivelTecnológico, GastosPublicidad)
- productos(IdentificadorProducto, IdentificadorMercado, IdentificadorEmpresa)
- mercado(IdentificadorMercado, IPC, NivelCompetitivo)
- valores(IdentificadorProducto, Coste, PVP, Trimestre, Percepción)

Primero, hay que destacar que tanto para el producto, como para la empresa, se ha eliminado la información de nombre, puesto que es redundante con el identificador. Una posibilidad sería proposicionalizar, para utilizar un algoritmo basado en una representación atributo-valor. Para ello, se podría tomar como base, los productos, y generar una instancia por producto. Sin embargo, dado que para cada producto, pueden existir muchos registros en la tabla de valores, esto no es directo. También se podría generar una instancia por cada registro en la tabla valores, pero estor haría que cada instancia tendría mucha información redundante (toda la relativa al producto, mercado y empresa). Además, recordar que cuando se parte de bases de datos relacionales, pude que las relaciones entre los datos pueden ser muy relevantes a la hora de clasificar. Por ejemplo, un la apreciación de si un producto es caro puede venir dada por la apreciación que tengan los clientes del resto de productos de esa empresa. Por tanto, una representación atributo-valor no se considera recomendada, aunque con ciertas transformaciones podría darse.

Además, tener en cuenta que la tabla valores se ha partido en dos. Habría que utilizar, por tanto, un algoritmo de aprendizaje relacional, como ALEPH, para resolver este problema. Sin embargo, Aleph necesita discriminar entre tuplas positivas y negativas. Para ello, la percepción del consumidor sobre el producto debería ser sacado de la tupla valores, y utilizar ese predicado como predicado a aprender. De esta forma, los predicados que quedarían son los siguientes:

- empresa(IdentificadorEmpresa, Capital, NivelTecnológico, GastosPublicidad)
- productos(IdentificadorProducto, IdentificadorMercado, IdentificadorEmpresa)
- mercado(IdentificadorMercado, IPC, NivelCompetitivo)
- valores(IdentificadorProducto, Coste, PVP, Trimestre)
- ProductoCaro(IdentificadorProducto, Trimestre)

Y el concepto a aprender es *ProductoCaro*. Si quisiéramos aprender con un algoritmo tipo FOIL o ALEPH, deberíamos proporcionar:

- Concepto a aprender: ProductoCaro(IdentificadorProducto, Trimestre)
- Conjunto de ejemplos positivos: vendría dado por todas las tuplas $< IdentificadorProducto, Trimestre >$, que cumplen que el producto es caro en ese trimestre, es decir, las tuplas $< 1, tercer >$, $< 1, cuarto >$, $< 2, segundo >$, $< tres, primer >$, $< 3, tercer >$, $< 4, segundo >$.
- Conjunto de ejemplos negativos: se compondría del resto de tuplas $< IdentificadorProducto, Trimestre >$ que no están entre los ejemplos positivos.
- Conocimiento del dominio, o resto de predicados de la base de datos.

Si utilizáramos ALEPH, tendríamos que proporcionar, además, un sesgo declarativo al lenguaje, a través de los modos. Por ejemplo habría que indicar qué predicados se pueden añadir a una clausula, y si sus variables deben estar instanciadas o no previamente. Por ejemplo, el modo *mode(1,valores(+IP, -number, -number))*, indicando

que el predicado valores sólo puede usarse una vez en la clausula, con un Identificador de producto usado como variable de entrada, y dos números que se usan como de salida, para el coste y el pvp.

Una posible salida del algoritmo vendría dada por la siguiente regla:

ProductoCaro(ID-P, Trimestre) :- Productos(ID-P, ID-M, ID-E), Mercado(ID-M, 2,medio).

Con esta regla, el producto "Lechugas Garrido" que se venderá en el cuarto trimestre a 2,4 euros, y que se fabrica a 1,5 euros, generará la siguiente tupla:

- valores(4,2, 4, 1, 5)

Por tanto, como es cierto Productos(4, 2,3) y Mercado(2,2,medio), la salida sería que se cumple ProductoCaro(4, cuarto).

Pregunta 3/3 (2 puntos)

Los empleados vienen caracterizados por 47 datos, que se pueden incluir en una única tabla de 500 instancias (una por empleado). En principio, y dada la escasa descripción del problema, se puede asumir que sus datos son numéricos y nominales.

Si se quieren dividir los 500 empleados en 5 grupos, en criterios basados en características similares, se puede pensar en utilizar un algoritmo de agrupación o *clustering*. De esa forma, se podría utilizar el algoritmo K-medias, mapas auto-organizativos de Kohonen, o una técnica aglomerativa.

En ambos casos, antes de nada, hay que definir una medida de similitud, como puede ser la distancia euclídea ponderada. Eso probablemente requeriría una normalización de los atributos. Además, dado que hay atributos nominales en los datos, habría que definir una medida de distancia para todos los datos nominales. Por último destacar que entre los datos hay algunos que son irrelevantes, como el DNI, y otros, como la dirección, que habría que revisar (puede no interesar saber la calle en la que viven, pero quizá sí la ciudad, o la zona).

Esta "limpieza" de los datos podría incluir alguna selección de características (que elimine atributos irrelevantes) y/o de instancias (eliminando aquellas instancias que pudieran incluir mucho ruido en el conjunto de entrenamiento).

Por último, habría que ejecutar el algoritmo de agrupación. Si se utiliza k-medias, habría que indicarle que el número de grupos a obtener es 5. con un mapa auto-organziativo de Kohonen, podríamos tener un mapa lineal de 5 neuronas. Si se utiliza una técnica aglomerativa que genera un dendograma, los grupos se pueden elegir a posteriori a partir del árbol generado.