Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2613 – Inteligencia Artificial (II/2017)

Solución Examen

Pregunta 1

Verdadero o Falso: Argumente su respuesta en cualquiera de los dos casos.

a) Sea $\Sigma = \{\forall x\,(P(x) \rightarrow x = a)\}$. Existe una fórmula atómica ψ, que sólo menciona al predicado P, tal que $\Sigma \cup \{\psi\}$ es inconsistente. (Ayuda: las formulas atómicas no contienen conectivos lógicos ni cuantificadores.) **(0.75 ptos.)**
Solución: Falso. Primero observemos que cualquier interpretación para Σ debe asignar a lo más un elemento para la extensión del predicado P. Para provocar una inconsistencia, en ψ debemos usar una fórmula que fuerce a la extensión de P a tener más de un elemento. Sin embargo, ψ puede ser solo $P(c)$ o $P(a)$. Ninguna de estas formulas forzaría a la extensión de P a tener dos elementos puesto que c siempre se puede interpretar igual que a.

b) DFS(h) es una versión del algoritmo de búsqueda en profundidad DFS que recibe una función heurística h como parámetro. Al expandir un nodo, ordena sus sucesores por h y los agrega a la frontera de búsqueda de tal forma que un sucesor de menor h queda al tope del stack. Si h^* es la heurística perfecta (es decir, predice el costo exacto para todo nodo), entonces DFS(h^*) es óptimo y ejecuta en tiempo lineal en el largo de la solución cuando el problema es uno de costos uniformes (es decir, todas las acciones tienen el mismo costo asociado). **(0.75 ptos.)**

Solución: Verdadero. Supongamos que DFS(h^*) retorna el camino $s_1 s_2 \ldots s_n$. Ahora observamos que su largo es igual a $h^*(s_1)$ y por lo tanto es óptimo. Eso es porque, como la heurística es perfecta, $h^*(s_{i+1}) - h^*(s_i) = 1$ (porque 1 es el costo de la acción) y $h^*(s_n) = 0$.

c) Considere el programa:

```
poss(soplar(V),S) :- vela(V), holds(encendida(V),S).
poss(encender(V),S) :- vela(V), \+ holds(encendida(V),S).
efecto_negativo(soplar(V),encendida(V)).
efecto_positivo(encender(V),encendida(V)).
holds(F,do(A,S)) :- holds(F,S),\+ efecto_negativo(A,F).
holds(F,do(A,S)) :- efecto_positivo(A,F).
legal(s0).
legal(do(A,S)) :- legal(S),poss(A,S).

vela(V) :- member(V,[v1,v2]).
holds(encendida(v1),s0).
```

La segunda respuesta dada por Prolog a la consulta `legal(S),holds(encendida(v2),S)` es `S = do(soplar(v1), do(encender(v2), s0))`. **(0.75 ptos.)**

Solución: Falso. Observando que el predicado legal simplemente enumera las situaciones en profundidad, y que por el orden en que está escrito el programa, primero siempre se genera `soplar` y luego `encender` (de ser posibles) y que primero se considera a v1 y luego a v2, podemos dibujar el árbol de situaciones legales y encontrar que `S = do(encender(v2), do(soplar(v1), s0))` es la segunda respuesta.

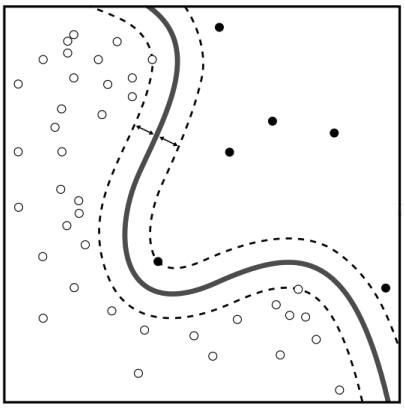
d) Suponga un problema de búsqueda con adversarios en donde el adversario siempre puede ejecutar una única acción. Sobre este problema, Minimax se comporta esencialmente igual que DFS (es decir, expanden los mismos nodos, en el mismo orden y ocupan recursos similares), cuando a este último se le pasa la condición objetivo "ganar el juego". **(0.75 ptos.) Solución:** Falso. Minimax siempre revisa el árbol completo, porque evalúa siempre a todos los hijos. DFS, en cambio, una vez que encuentra una rama se detiene.

e) El algoritmo de descenso de gradiente estocástico permite entrenar clasificadores utilizando una gran cantidad de datos, pero su convergencia puede ser lenta, debido al gradiente parcial que es utilizado para el descenso. Una manera de aliviar este problema es usar la técnica de momentum, que permite evitar las oscilaciones dadas por el ruido inherente de un minibatch. **(0.75 ptos.)**
Solución: Verdadero, al combinar la dirección de descenso previa con el gradiente parcial, y por eso ruidoso, se consigue una dirección de descenso más estable y por lo tanto, una convergencia más rápida.

f) Una manera de mejorar los árboles de decisión, es modificando el procedimiento de testeo realizado en cada nodo. En particular, con el fin de reducir el sobreentrenamiento, en vez de usar tests basados en umbrales para variables numéricas, es posible usar un clasificador de K-vecinos cercanos. **(0.75 ptos.)**
Solución: Falso, el algoritmo de K-vecinos cercanos tiene serios problemas de sobreentrenamiento cuando hay pocos ejemplos. Esto ocurre regularmente en los árboles de decisión, a medida que aumenta la profundidad.

g) Siempre es posible transformar un problema de clasificación multiclase, en una serie de problemas de clasificación binarios. En particular, dado un problema con K clases, es necesario entrenar al menos $\frac{K\times(K-1)}{2}$ clasificadores binarios (uno por cada par de clases), para luego tomar la decisión en base a qué clasificador obtuvo más *votos*. **(0.75 ptos.)**
Solución: Falso, bastan K clasificadores binarios, donde cada uno se entrena usando como ejemplos de la clases positiva a los de una clase en particular, y todo el resto de los ejemplos como de la clase negativa. Este esquema se conoce como *one-vs-all*.

h) Un *Soft-margin SVM* no puede clasificar perfectamente el problema presentado en la siguiente figura **(0.75 ptos.)**:



Solución: Falso, si puede clasificarlo si se utilizan kernels que transformen el espacio de características del conjunto de datos de entrada.

Pregunta 2

Algunos problemas de búsqueda parecen ser una combinación de otros dos problemas de búsqueda. Como ejemplo, considere el problema de descargar y llevar a destino containers en un puerto. En esta situación los containers se pueden ubicar unos encima de otros (similar a un problema de bloques), pero, a su vez, los containers deben moverse a diferentes destinos (similar a un problema logístico). No es sorprendente, entonces, que para estos problemas sea posible construir dos heurísticas admisibles h_1 y h_2, cada una de las cuales se "enfoca" en una parte del problema, prácticamente ignorando a la otra.

a) Demuestre que cuando h_1 y h_2 son admisibles, entonces $\max\{h_1, h_2\}$ también lo es. **(1 pto.) Solución:** Como ambas heurísticas son admisibles $h_1(s) < h^*(s)$ y $h_2(s) < h^*(s)$, para todo estado s. De estas dos desigualdades se sigue inmediatamente que $\max\{h_1, h_2\}(s) < h^*(s)$ y por lo tanto $\max\{h_1, h_2\}$ es admisible.

b) Existen situaciones en que $\max\{h_1, h_2\}$ es una "mala heurística". Encuentre una de estas situaciones, definiendo un problema de búsqueda en donde se aprecie que A* no es guiado bien por tal heurística. Inspírese en el ejemplo de arriba y ejerza la claridad y la elegancia al explicar. **(3 ptos.)**

Posible solución: Consideremos una extensión al mundo de bloques en donde hay dos mesas: mesa1 y mesa2. En solo una de ellas (mesa2) es posible pintar a los bloques blancos. En el estado inicial, donde hay $N > 2$ bloques, todos los bloques están sobre la mesa1. En el estado final, todos los bloques deben estar sobre la mesa1 y se pide una configuración específica de bloques, pero uno de los bloques debe estar pintado blanco. Consideramos dos heurísticas para este problema. La primera, h_1, cuenta el número de subobjetivos "on" que no se han satisfecho. La segunda, h_2, cuenta cuantos bloques aún no están pintados del color correcto. Claramente ambas heurísticas son admisibles. Si tomamos el máximo de las dos heurísticas A* está guiado hacia armar la configuración de bloques porque en la práctica se ignora h_2 porque h_1 es casi siempre mayor. Asi, si el bloque a pintar tiene 10 bloques sobre él A* avanzará hacia el estado en donde esos 10 bloques están sobre el bloque a pintar, antes de considerar llevarlo a la otra mesa y pintarlo.

c) Explique con precisión cómo podría usar A* con h_1 como heurística principal y a h_2 como un "quebrador de empates". Hágalo de tal forma el algoritmo siga siendo óptimo. (No debe demostrar que lo sigue siendo.) **(2 ptos.)**

Solución: En la cola de prioridades es posible usar $f = K(g + h_1) + h_2$ donde $K = 2^p$ tal que p es un número tal que $h_2 < 2^p$ (p es, entonces, una constante dependiente del problema). Esencialmente, lo que estamos haciendo es mantener el mismo ranking de estados que usaría A* cuando es usado con h_1 pero variamos la posición relativa de los estados que tienen igual valor $g + h_1$.

Pregunta 3

Para representar el conocimiento en un árbol genealógico se utilizan dos predicados:

- $Pr(x, y)$: expresa que x es progenitor de y.
- $An(x, y)$: expresa que x es ancestro de y.

Para definir el predicado An en términos de Pr es posible utilizar la siguiente fórmula:

$$\psi = \forall x \forall y \forall z \, (Pr(x, y) \wedge An(y, z) \rightarrow An(x, z)) \wedge \forall x \forall y \, (Pr(x, y) \rightarrow An(x, y))$$

a) Use resolución para demostrar que "si a no es ancestro de b entonces no hay un camino de largo 2 entre a y b a través de la relación Pr". **(4 ptos.) Solución:** Para la resolución demostramos que $\{\neg An(a, b), \psi\} \models \neg \exists z Pr(a, z) \wedge Pr(z, b)$. Las cláusulas que se obtienen son:

(a) $\neg An(a, b)$
(b) $\neg Pr(x, y) \vee \neg An(y, z) \vee An(x, z)$
(c) $\neg Pr(x, y) \vee An(x, y)$
(d) $Pr(a, c)$
(e) $Pr(c, b)$

donde c es una constante de Skolem.

Al resolver (b) y (c) obtenemos $\neg Pr(x, y) \vee \neg Pr(y, z) \vee An(x, z)$ y luego encontramos la cláusula vacía resolviendo esta cláusula con (a), (d), y (e).

b) Generalice la resolución anterior. Específicamente, diga qué pasos de resolución son necesarios para demostrar que no hay un camino de largo n entre a y b si a no es ancestro de b, para cualquier n. **(2 ptos.)**

Solución: Observemos que las cláusulas (a)-(c) se mantienen iguales, mientras que (d) y (e) quedan reemplazadas por el conjunto $C = \{Pr(a, c_1), Pr(c_1, c_2), \ldots Pr(c_{n-2}, c_{n-1}), Pr(c_{n-1}, b)\}$.

Si resolvemos a (b) consigo misma $n-2$ veces obtendremos:

$$\neg Pr(x, x_1) \vee \neg Pr(x_1, x_2) \vee \neg Pr(x_2, x_3) \vee \ldots \vee \neg Pr(x_{n-2}, x_{n-1}) \vee \neg An(x_{n-1}, z) \vee An(x, z)$$

Ahora la resolvemos con (c) y (a) para obtener:

$$\neg Pr(a, x_1) \vee \neg Pr(x_1, x_2) \vee \neg Pr(x_2, x_3) \vee \ldots \vee \neg Pr(x_{n-2}, x_{n-1}) \vee \neg Pr(x_{n-1}, b)$$

Resolvemos esta última cláusula con todas las cláusulas en C para obtener la cláusula vacía.

Pregunta 4

Un *Double Soft-margin SVM* es un algoritmo de clasificación similar a un *Soft-margin SVM*, pero que impone dos restricciones sobre el margen funcional: una sobre su valor mínimo (*Soft-margin SVM*) y una sobre el máximo. Tomando esto en consideración, responda las siguientes preguntas:

a) Defina el problema de aprendizaje para un *Double Soft-margin SVM*, como un problema de minimización cuadrático con restricciones lineales. **(3 ptos.)**
Solución: Por cada ejemplo, se agregan dos nuevas restricciones, que imponen un límite superior constante al margen (τ), que si es violado se penaliza en una cantidad μ_i.

$$\begin{aligned} \underset{w,b,\{\xi_i\},\{\mu_i\}}{\operatorname{argmin}} \quad & \frac{\lambda}{2}\|w\|^2 + \sum_i^N \xi_i + \sum_i^N \mu_i \\ \text{s.a} \quad & y_i(w^\top x_i + b) \geq 1 - \xi_i, \forall i \in (1, N) \\ & \xi_i \geq 0, \forall i \in (1, N) \\ & y_i(w^\top x_i + b) \leq \tau + \mu_i, \forall i \in (1, N) \\ & \mu_i \geq 0, \forall i \in (1, N) \end{aligned}$$

b) Modifique el problema de aprendizaje definido en el ítem anterior, planteándolo esta vez como un problema de minimización cuadrático sin restricciones. **(3 ptos.)**
Solución:

$$\underset{w,b}{\operatorname{argmin}} \quad \frac{\lambda}{2}\|w\|^2 + \sum_i^N \max\{0, 1 - y_i(w^\top x_i + b)\} + \max\{0, y_i(w^\top x_i + b) - \tau\}$$

Pregunta 5

Considere una red convolucional profunda de 8 capas, que es utilizada para resolver un problema binario de clasificación de imágenes, consistente en decidir la aparición o no de personas en ellas. La red tiene 5 capas convolucionales (*stride* de 1, sin *padding*) con 128 filtros de 5x5 cada una, seguida por 3 capas *fully-connected* con 256, 256 y 1 neurona, respectivamente. En base a esta información, responda las siguientes preguntas:

a) ¿Cuál es la función de las capas convolucionales en esta red? ¿Por qué no basta con usar sólo capas *fully-connected*?. **(2 ptos.)**
Solución: Las capas convolucionales permiten detectar patrones visuales altamente especializados, con una cantidad drásticamente menor de parámetros que las capas *fully-connected*. El usar capas *fully-connected* aumenta el riesgo de sobreentrenamiento, además de perder la capacidad de detectar patrones visuales altamente localizados.

b) Asumiendo que las imágenes tienen una resolución de 224x224, indique la cantidad total de parámetros que utiliza esta red. **(2 ptos.)**
Solución: $\underbrace{128 \times 5 \times 5 \times 3}_{\text{filtros capa 1}} + \underbrace{4 \times 128 \times 5 \times 5 \times 128}_{\text{filtros capas 2-5}} + \underbrace{204 \times 204 \times 128 \times 256}_{\text{neuronas capa 6}} + \underbrace{256 \times 256}_{\text{neuronas capa 7}} + \underbrace{256}_{\text{neurona capa 8}}$.

c) Después de realizar un análisis exploratorio, se verificó que en las imágenes utilizadas para entrenar la red, las cabezas de las personas tienen un diámetro aproximado de 15 pixeles. ¿En que capa esperaría observar entonces filtros (neuronas) que se activen fuertemente con cabezas humanas? **(2 ptos.)**
Solución: Como las capas convolucionales utilizan filtros de 5×5 y no existe *max-pooling* entre capas, los mapas de activaciones reducirán sus dimensiones en 2 por lado al pasar entre capas. Esto implica que los filtros capturarán 4 unidades de información más por cada capa. Tomando esto en consideración, si partimos con filtros que cubren 5×5 pixeles, en la capa 4 se tendrán filtros que cubren 17×17 pixeles, que es una dimensionalidad suficiente para contener una cabeza completa.