PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Pauta Interrogación 2

IIC2613 - Inteligencia Artificial

Segundo Semestre, 2019

Responda cada una de las tres preguntas en una hoja por separado.
Tiempo: 2 hrs.

1. *a*) (2 puntos) Al "relajar" un problema de búsqueda P, se obtiene un problema de búsqueda P' cuyo espacio de estados y grafo de búsqueda están íntimamente relacionados al de P. Explique en detalle cuál es esta relación, ilustre con un ejemplo de un problema de búsqueda, y demuestre que si c es la solución óptima a P y c' es la solución óptima para P', entonces $c' \leq c$.
   Respuesta: el grafo de P es subconjunto del de P' (al relajar pueden haber estados nuevos y acciones nuevas) - Al relajar un problema P, obteniendo P', se hacen disponibles nuevas acciones que en el problema P no eran posibles debido a las restricciones.
   - En P', para pasar de S_0 a S_2 existe una acción e_1 de costo unitario.
   - En P, para pasar de S_0 a S_2 no existe una acción directa (ya que hay una restricción que impide pasar directamente entre estos dos estados), por lo tanto es necesario ejecutar la acción e_2 para ir de S_0 a S_1, y luego la acción e_3 para ir de S_1 a S_2.
   - Si los costos de las acciones son uniformes, $e_1 < e_2 + e_3$ lo que es lo mismo que $c*(P') < c*(P)$.

   b) - Si los costos de las acciones no son uniformes, la solución al problema relajado P' contendrá e_1 y no contendrá e_2 ni e_3, por lo tanto $c*(P') \leq c*(P)$.

   c) El algoritmo A*, retorna un nodo objetivo inmediatamente después de extraerlo desde la OPEN. Considere una versión de A* que, en vez de hacer aquello, retorna un nodo objetivo inmediatamente después de que este es generado (y antes de ser agregado a la lista OPEN).

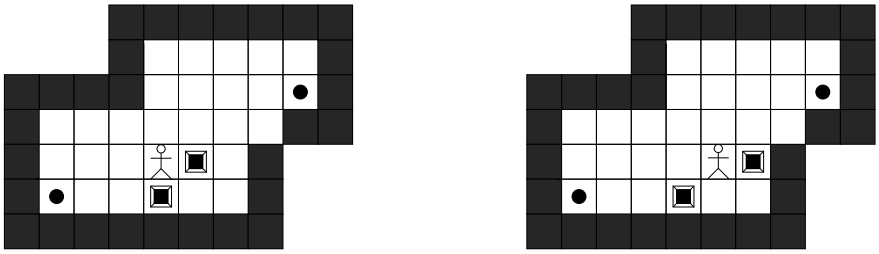
   1) (2 puntos) Muestre que esta variante de A* es subóptima. Use como contraejemplo un espacio de búsqueda con tres estados.
      Respuesta: Un estado S puede generar 2 estados objetivos: S^1 y S^*, siendo ambos $\in Succ(S)$.S^1 es sub-optimo y $S*$ es óptimo, sin embargo $S*$ no alcanzó a ser generado ya que al generar S^1 retornó como goal

   2) (2 puntos) Sea c_{sol} el costo de la solución encontrada por esta variante de A* y suponga se utiliza una heurística admisible. Demuestre, que $c^* \leq c_{\text{sol}} \leq c^* + c_{\text{ult}}$, donde c^* es el costo de la solución óptima al problema de búsqueda en cuestión y c_{ult} es el costo del último arco del camino que conecta el estado inicial con el estado retornado. ***Ayuda****: para resolver esta pregunta, puede convenir que identifique al padre del nodo retornado con una letra (por ejemplo s_{padre}) y que luego use relaciones conocidas para llegar al resultado.*
      Respuesta: Sea S_{padre} el padre de S_{sol} ($S_{sol} \in Succ(S_{padre})$
      Por propiedades de A*, se sabe que (ya que hasta S_{padre} el algoritmo funciona igual a A*):
      $g(padre) + h(padre) \leq c*$
      como h es no negativa, se puede decir que:
      $g(padre) \leq c*$ (1)
      Por otro lado:
      $c(S_{sol}) = g(S_{padre}) + c_{ult}$ (2)
      reemplazando $g(S_{padre})$ en (2) según (1), podemos demostrar que:
      $c_{sol} \leq c* + c_{ult}$

2. El juego de *Sokoban* es un puzle cuyo objetivo es empujar cajas hasta posiciones de almacenamiento. En este problema, un agente puede moverse sobre una grilla hacia arriba, abajo, a la izquierda o a la derecha. Una posición de la grilla puede estar bloqueada (*obstáculo*), puede contener una caja, o puede contener un punto de destino. En una misma celda, no pueden haber dos cajas simultáneamente, ni tampoco estar el agente y una caja.

   El agente puede navegar sobre cualquier posición de la grilla sin obstáculos. Al avanzar el agente sobre una posición que contiene una caja, ésta se desplaza una celda en la dirección del movimiento del agente, a menos que haya un obstáculo en tal celda, en cuyo caso el movimiento del agente no es posible.

   La figura muestra un estado particular y el resultado de ejecutar un movimiento a la *derecha*.



   a) (2 puntos) Sea $d_m(c_1, c_2)$ la distancia Manhattan entre las celdas c_1 y c_2. Defina una heurística admisible en base a sumas de distancias manhattan que retorne valor no nulo para todo estado que no es un objetivo. Demuestre que la heurística es admisible.

   b) (2 puntos) Suponemos que eliminamos las restricción de que no pueden haber dos cajas en una misma posición y que debe ser el agente quien empuje las cajas. En ese caso, el costo de resolver el problema es:

$$\sum_{c \in C} \min_{c_d \in Obj} d_m(c, c_d).$$

   Tal costo es una heurística admisible pues es el costo de una solución a un problema más relajado que el original.

   Sea h_m la heurística anterior. ¿Es $2 \cdot h_m$ admisible? Demuestre su respuesta.

   Respuesta: No es admisible. Suponga que que solo hay una caja y que el agente puede resolver el problema haciendo un movimiento (es decir, la caja está inmediatamente al lado de una posición objetivo). En este caso, el problema tiene costo 1, pero el doble de la heurística anterior es 2, que es mayor al costo óptimo. La heurística es inadmisible.

   c) (2 puntos) Dé una heurística admisible mejor que la definida en *b*), que sea computable en tiempo polinomial en el tamaño de la grilla. Justifique que es admisible.

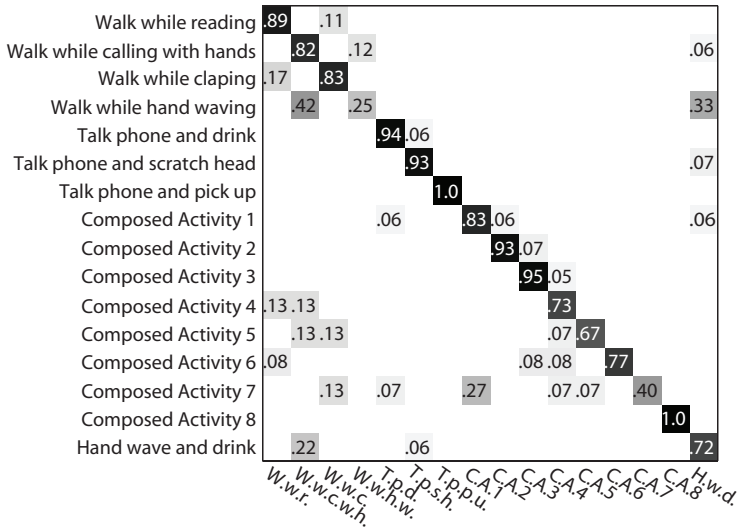
   Respuesta: Una posibilidad es no relajar el hecho que en una posición objetivo no puede haber más de una caja a la vez. (en b, si estábamos relajando esa restricción).

   En ese caso, podemos suponer que la posición objetivo se puede ocupar con una sola caja. Entonces, podemos resolver encontrar una heurística con el siguiente algoritmo greedy.

   1) $Disp := Obj$.
   2) $Cajas := C$.
   3) $H := 0$.
   4) Mientras $Cajas \neq \emptyset$
      4.1) Sean $c \in Cajas$, $c_d \in Disp$ tales que minimizan $d_m(c, c_d)$
      4.2) $H := H + d_m(c, c_d)$
      4.3) $Disp := Disp \setminus \{c_d\}$
      4.3) $Cajas := Cajas \setminus \{c\}$
   5) return H

Claramente el algoritmo es polinomial en el tamaño del problema. Solo se realizan $|Obj|$ iteraciones. En cada una se calcula una operación de mínimo que también es polinomial. La heurística es admisible porque también resuelve un problema relajado. Esta heurística es también superior a la anterior, puesto que retorna en muchos casos estimaciones mayores que la heurística en b).

3. El uso de una matriz de confusión es una herramienta comúnmente utilizada para evaluar el rendimiento de un algoritmo de clasificación. En la siguiente figura se utiliza esta herramienta para observar el rendimiento de un clasificador en el set de test para una aplicación de reconocimiento de actividades.



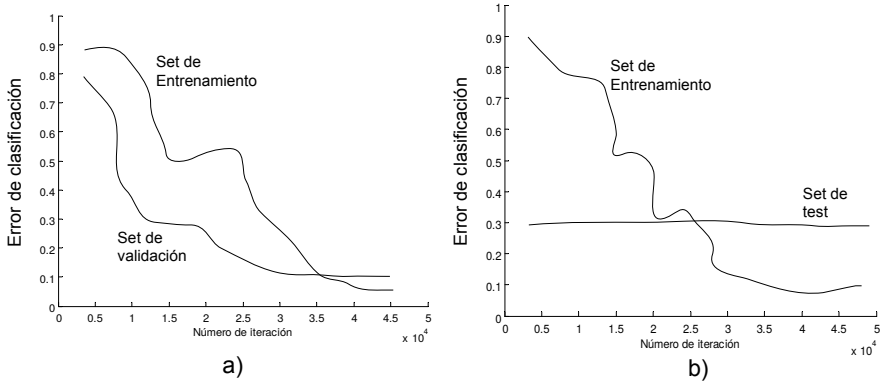
a) (I) (0.75 puntos)¿Cuál es el error promedio del clasificador en términos de las 16 clases posibles, asuma que cada clase tiene el mismo número de instancias en el set de test ?.
Respuesta: Corresponde al promedio de la diagonal de la matriz de confusion: 0.7913.

(II) (0.75 puntos)¿Cuál es la clase que obtiene la mejor exactitud de clasificación?
Respuesta: talk-phone-and-pickup y composed-activity-8, ambas con clasificacion perfecta.

(III) (0.75 puntos)¿Cuál es la clase que obtiene la peor exactitud de clasificación?
Respuesta: Walk while hand waving con un 25 % de exactitud en la clasificacion.

(IV) (0.75 puntos)¿Cuál es el tipo de error de exactitud de clasificación más frecuente?
Respuesta: Walk-while-hand-waving es confundido con walk-while-calling-with-hands en un 42 % de los casos de esta clase.

b) (3 puntos) En las siguientes figuras comente acerca de la forma de la curva de error en los set de datos indicados.



Respuesta:

A)

1) Ambas curvas, entrenamiento y validacion, muestran un comportamiento adecuado al proceso de entrenamiento, en el sentido que la tendencia es que el error va disminuyendo a medida que el algoritmo va iterando sobre los datos.
2) A partir de aproximadamente la iteracion 3.5x104 comienza a observarse sobreajuste (overfitting).

B)

1) La curva de entrenamiento tiene un comportamiento normal, sin embargo, la curva de test indica que los datos de entrenamiento no son una muestra representativa del problema. Esto pues el rendimiento en el set de test es indiferente al proceso de aprendizaje que se observa durante el entrenamiento (error en set de entrenamiento disminuye pero no ası en set de test).