Grado en Ingeniería Informática en Sistemas de Información
Inteligencia Artificial - Curso 2016/17
EXAMEN ENERO-18/01/2017

APELLIDOS:_____**NOMBRE:**_____**D.N.I.:**_____

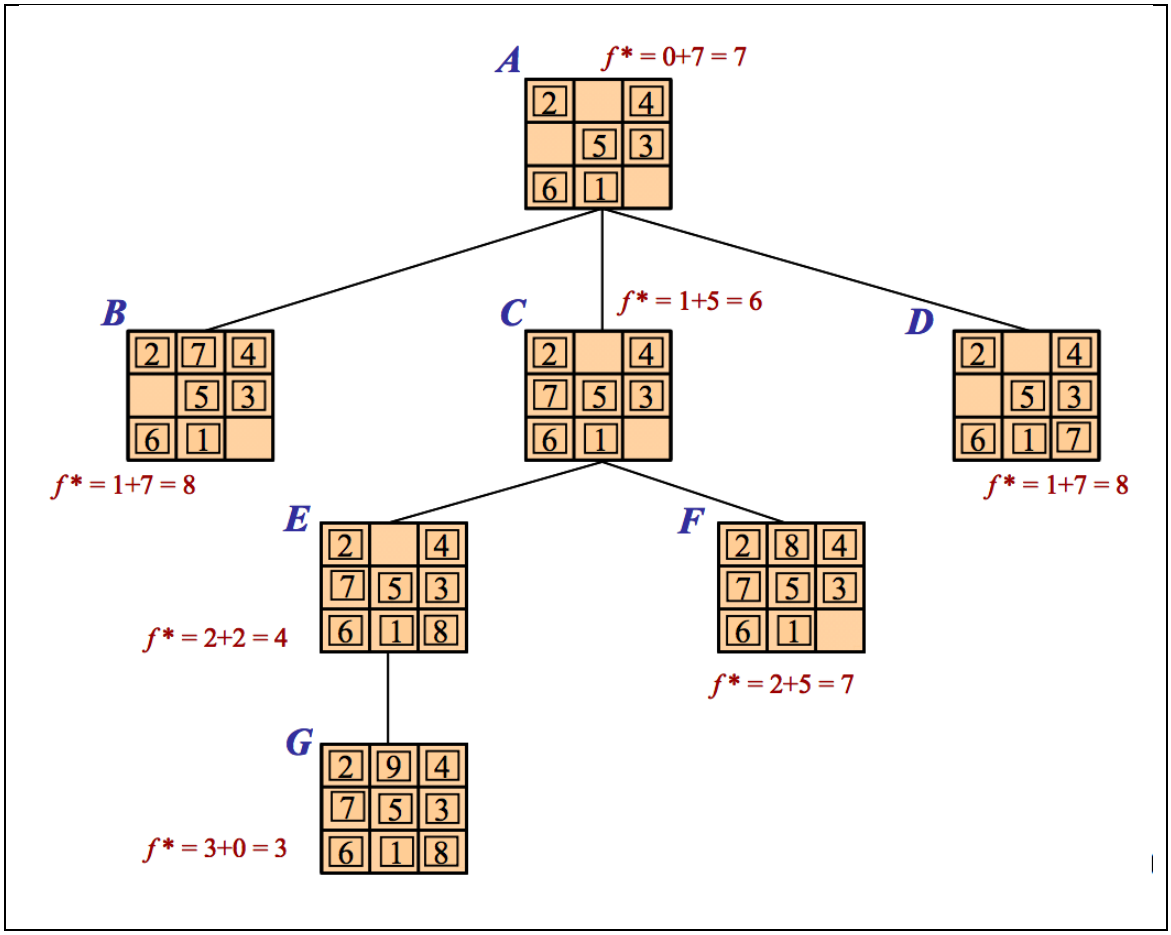
IMPORTANTE: Descarga del Aula Virtual los ficheros necesarios para el desarrollo del examen. Debe entregar a través de la actividad habilitada para el examen un fichero llamado TusApellidos.zip que contenga los ficheros fuentes .m que hayáis creado/modificado para el EJ2. Además debe entregar al profesor en mano este enunciado con el EJ1 y los apartados b) y c) del EJ2 rellenos.

EJ1.- **(5 puntos)** Un cuadrado mágico es un tablero 3x3 donde cada celda está rellena con un número distinto del 1 al 9 de tal forma que tanto las diagonales como las filas como las columnas suman siempre 15. Aplicar el algoritmo A* al tablero de la figura, donde la meta es llegar a un tablero que sea un cuadrado mágico. Las únicas acciones posibles son incorporar el número más pequeño de los que queden por colocar en una celda vacía. Cada acción tiene un coste de una unidad. Use como función heurística h el número total de filas, columnas y diagonales que no suman 15. En el caso de la figura, el valor de h es 7 ya que hay 3 filas, 3 columnas y una diagonal que no suman 15.

2		4
	5	3
6	1	

a) Dibujar el árbol de búsqueda.

En el árbol debe indicar para cada nodo del árbol de búsqueda los valores de f desglosado como f = coste + h.



b) Indicar la frontera en cada iteración.

iter = 0 A
iter = 1 CBD
iter = 2 EFBD
iter = 3 GFBD

c) Indicar la solución obtenida, el coste de la solución y el número de nodos expandidos.

Coste: 3
Número de nodos expandidos: 4
Solución:

2		4
	5	3
6	1	

2		4
7	5	3
6	1	

2		4
7	5	3
6	1	8

2	9	4
7	5	3
6	1	8

d) ¿Es la heurística h admisible? En caso afirmativo, diga por qué y en caso negativo indique un nodo que no cumpla la propiedad.

No, por ejemplo el nodo E tiene como valor h(E)=2 y el coste para alcanzar el nodo objetivo G desde E es 1 pues solo se necesita una acción.

e) ¿Es el algoritmo A* óptimo con la heurística h?

Sí, porque alcanza el coste mínimo.

EJ2.- (5 puntos) Realizar un estudio de regresión logística para predecir si un microchip será aceptado o rechazado en base a los resultados de dos tests realizados sobre el mismo. Para ello, se dispone de X piezas de microchips almacenadas en el conjunto de datos ex2data2.txt. Cada fila de este conjunto está formada por 3 valores: 2 atributos con los resultados obtenidos de los dos tests realizados y un último valor que indica si la pieza de microchip fue aceptada (valor 1) o rechazada (valor 0).

Descripción de los ficheros:

- `main`: programa principal a utilizar.
- `holdout(X,y,m,percent)`: realiza una partición en el conjunto X e Y según el porcentaje indicado.
- `sigmoid(z)`: calcula la función sigmoide.
- `costFunctionReg(theta, X, y, lambda)`: calcula el coste en la regresión logística teniendo en cuenta la regularización.
- `plotDecisionBoundary(theta,X, y,)`: dibuja la frontera de decision.
- `predict(theta, X)`: predice el conjunto X usando el modelo `theta` de regresión logística.
- `mapFeature(X1,X2)`: crea nuevos atributos a partir de los atributos X1 y X2.

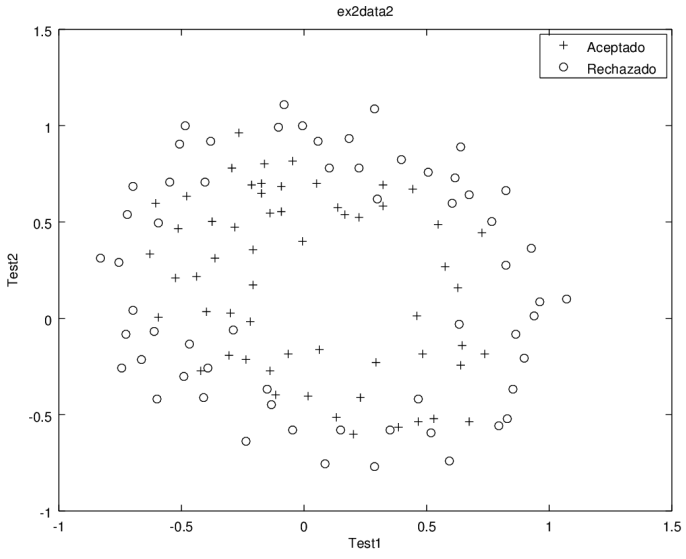


Figura 1. Resultado de la aceptación de microchips

a) Implementar la función `LogReg.m` para obtener una regresión logística con regularización que se evalúe mediante holdout. Esta función recibirá los conjuntos de entrenamiento y de test junto al parámetro lambda y devolverá la tasa de acierto en porcentaje obtenida cuando se predice el conjunto de test y el conjunto de training junto con el modelo obtenido.

```
function [acc_test, acc_train, theta] = LogReg(X_train, y_train,X_test,y_test, lambda)

    % Initialize Theta
    initial_theta = zeros(size(X_train, 2), 1);


    % Set Options
    iteraciones = 100;
    options = optimset('GradObj', 'on', 'MaxIter', iteraciones);

    % Optimize
    [theta, J, exit_flag] = ...
     fminunc(@(t)(costFunctionReg(t, X_train, y_train, lambda)), initial_theta, options);

    %Dibujar coste
    plot(J);

    %Predecir
    p1 = predict(theta, X_test);
    p2 = predict(theta, X_train);

    %Calcular el error
    acc_test = mean(double(p1 == y_test)) * 100;
    acc_train = mean(double(p2 == y_train)) * 100;

end
```

b) Haciendo uso de la función del apartado anterior, implementar en el programa principal una regresión logística sin regularización y evaluar el modelo mediante holdout. Debe imprimir en la pantalla la tasa de acierto para el conjunto de training y para el conjunto de test y debe dibujar la frontera de decisión usando para ello la función `plotDecisionBoundary`. ¿Por qué no funciona la regresión logística? Responda brevemente.

Porque los puntos no son separables linealmente

c) Como se puede comprobar en la Figura 1, los puntos no son separables linealmente. Una solución es crear más atributos mediante la función `mapFeature.m`, que mapea los atributos en términos polinomiales de x_1 y x_2 hasta la sexta potencia (x_1, x_2, x_1^2, x_2^2, x_1*x_2, $x_1*x_2^2$,…) y devuelve una matriz con 28 atributos, permitiendo de esta manera obtener una frontera de decisión mucho más compleja que se pueda adaptar mejor a los puntos. Implementar en el programa principal una regresión logística sin regularización con los datos mapeados con 28 atributos y evaluar el modelo mediante hold-out. Debe imprimir en la pantalla la tasa de acierto para el conjunto de training y para el conjunto de test y debe dibujar la frontera de decisión. ¿Por qué no funciona la regresión logística? Responda brevemente.

Porque hay overfitting

d) Implementar la función `estudioLambdaLogReg.m` para obtener el mejor modelo posible de regresión logística con regularización en función del parámetro lambda. Esta función recibe un vector con posibles valores para lambda y debe devolver el valor de lambda que hace mínimo el error cometido en la predicción de un conjunto de validación.

```
function lambda = estudioLambdaLogReg(X_train,y_train,lambda_vec,
percent)

[X_train y_train X_val y_val] = holdout(X_train,y_train,percent);


error_val = zeros(length(lambda_vec),1);

  for j= 1:length(lambda_vec)

   lambda = lambda_vec(j);
   fprintf('\nlambda = %d...\n', lambda)
   [acc_val,          acc_train]          =          LogReg(X_train,
y_train,X_val,y_val,lambda);
   error_val(j) = acc_val;
  end


[a b] = min(error_val);

lambda = lambda_vec(b);

end
```

e) Una vez obtenido el parámetro lambda óptimo, implementar una regresión logística con regularización con dicho parámetro y evaluar el modelo mediante hold-out, imprimiendo en la pantalla la tasa de acierto para el conjunto de training y para el conjunto de test y debe dibujar la frontera de decisión.

```
Programa Principal

%% Examen convocatoria enero: Machine Learning

%% Inicialización
clear ; close all; clc

%% ===b) Regresión logística sin regularización con datos originales
fprintf('\n\nApartado b): Regresión logística sin regularización con
datos originales...\n')

fprintf('\nCargando datos...\n');
data = load('ex2data2.txt');

X = data(:,1:2);
y = data(:,3);

m = size(X,1);
X = [ones(m, 1) X];
percent = 0.7;

[X_train y_train X_test y_test] = holdout(X,y,percent);

lambda =0;
[acc_test,  acc_train,theta]  =  LogReg(X_train,y_train,X_test,y_test,
lambda);
```

```
%Dibujar frontera de decisión
plotDecisionBoundary(theta, X, y);

fprintf('\nExactitud del conjunto de test: %f\n', acc_test);
fprintf('\nExactitud del conjunto de entrenamiento: %f\n', acc_train);

pause

% ====== c) Regresión logística sin regularización con datos mapeados
fprintf('\n\nApartado c): Regresion logistica sin regularizaciÛn con
datos mapeados...\n')

fprintf('\nCargando datos mapeados...\n');

data = load('ex2data2.txt');

X = mapFeature(data(:,1), data(:,2));
y = data(:,3);


[X_train y_train X_test y_test] = holdout(X,y,percent);

lambda =0;
[acc_test,              acc_train,              theta]              =
LogReg(X_train,y_train,X_test,y_test,lambda);

%Dibujar frontera de decisión
plotDecisionBoundary(theta, X, y);

fprintf('\nExactitud del conjunto de test: %f\n', acc_test);
fprintf('\nExactitud del conjunto de entrenamiento: %f\n', acc_train);

pause

% ======== d) Parámetro lambda óptimo   ===================
fprintf('\n\nApartado d): Parámetro lambda óptimo con Regresión
logistica con datos mapeados...\n')

lambda_vec = [0.01 0.05 0.1 0.5 1]';

lambda_opt = estudioLambdaLogReg(X_train,y_train,lambda_vec, percent);


fprintf('\nEl valor ÛptimO de lambda es %f\n',lambda_opt);

pause
%===e)   Regresión   logística   con   regularización   con   datos
mapeados==================
fprintf('\n\nApartado   e):   Regresión   logística   con   parámetros
óptimos...\n')

[acc_test, acc_train, theta] = LogReg(X_train,y_train,X_test,y_test,
lambda_opt);

%Dibujar frontera de decisión
plotDecisionBoundary(theta, X, y);

fprintf('\nExactitud del conjunto de test: %f\n', acc_test);
fprintf('\nExactitud del conjunto de entrenamiento: %f\n', acc_train);
```