



Universidad del Quindío

Programa Ingeniería de Sistemas y Computación

Espacio Académico Inteligencia Artificial

Primer semestre académico de 2018

Guía de laboratorio

Profesor: Leonardo Hernández R.

Laboratorio	5-Regularización
Estudiantes por grupo	Máximo 3
Tiempo promedio por estudiante estimado	5h 00m

# 1 Objetivo

Para una institución de educación superior es muy útil predecir cuantos estudiantes se van a registrar en un espacio académico electivo, para iniciar el proceso de asignación de profesor, aula y horario. En este contexto, el objetivo de laboratorio es el siguiente:

*Experimentar con una regresión logística con características polinomiales y regularización para predecir si un estudiante registrará o no el espacio académico electivo Interacción Humano-Computador, HCI. La predicción de realizará con base en los resultados de las calificaciones de los espacios académicos Ingeniería de Software I y Lenguaje de Programación.*

Las características y la variable a predecir por parte de la regresión son por lo tanto:

- $x_1$ : Calificación de Ingeniería de Software I (0.0-5.0)
- $x_2$ : Calificación de Lenguaje de Programación (0.0-5.0)
- $x_3 = x_1^2$
- $x_4 = x_1 x_2$
- $x_5 = x_2^2$
- $x_6 = x_1^3$
- $x_7 = x_1^2 x_2$
- $x_8 = x_1 x_2^2$
- $x_9 = x_2^3$
- ...
- $y$ : ¿Registró Interacción Humano-Computador, HCI? (0/1)

Para el aprendizaje del modelo, se utilizarán datos sintetizados de 980 estudiantes, que serán suministrados con los archivos del laboratorio.

## 2 Convenciones

$m$ : Número de ejemplos de entrenamiento

$m_{cv}$ : Número de ejemplos de validación cruzada

$m_{test}$ : Número de ejemplos de prueba, para este laboratorio  $m + m_{cv} + m_{test} = 980$

$n$ : Número de características

$XI$ : Matriz de  $m$  filas con los valores de las características en el conjunto de entrenamiento

$XI_{cv}$ : Matriz de  $m_{cv}$  filas con los valores de las características en el conjunto de validación cruzada

$XI_{test}$ : Matriz de  $m_{test}$  filas con los valores de las características en el conjunto de prueba

$XP$ : Matriz resultado de agregar a la matriz  $XI$  las características polinomiales

$XP_{cv}$ : Matriz resultado de agregar a la matriz  $XI_{cv}$  las características polinomiales

$XP_{test}$ : Matriz resultado de agregar a la matriz  $XI_{test}$  las características polinomiales

$XN$ : Matriz resultado de normalizar la matriz  $XP$

$XN_{cv}$ : Matriz resultado de normalizar la matriz  $XP_{cv}$

$XN_{test}$ : Matriz resultado de normalizar la matriz  $XP_{test}$

$X$ : Matriz resultado de agregar una columna de unos a la izquierda de la matriz  $XN$

$X_{cv}$ : Matriz resultado de agregar una columna de unos a la izquierda de la matriz  $XN_{cv}$

$X_{test}$ : Matriz resultado de agregar una columna de unos a la izquierda de la matriz  $XN_{test}$

$y$ : Vector columna de  $m$  elementos con los valores para la variable a predecir en el conjunto de entrenamiento

$y_{cv}$ : Vector columna de  $m_{cv}$  elementos con los valores para la variable a predecir en el conjunto de validación cruzada

$y_{test}$ : Vector columna de  $m_{test}$  elementos con los valores para la variable a predecir en el conjunto de prueba

$x^{(i)}$ : Vector columna de  $n + 1$  elementos: un uno y los valores de las características del  $i$ -ésimo ejemplo de entrenamiento.

$x_j^{(i)}$ : Característica  $j$ -ésima del  $i$ -ésimo ejemplo de entrenamiento.

$y^{(i)}$ : Valor en el conjunto de entrenamiento de la variable a predecir para el ejemplo de entrenamiento  $i$ .

$h$ : Vector columna de  $m$  elementos con las predicciones de probabilidad del modelo para todos los ejemplos de entrenamiento

$h_{cv}$ : Vector columna de  $m_{cv}$  elementos con las predicciones de probabilidad del modelo para todos los ejemplos de validación cruzada

$h_{test}$ : Vector columna de  $m_{test}$  elementos con las predicciones de probabilidad del modelo para todos los ejemplos de prueba

$\mu$ : Vector fila de  $n$  elementos, donde  $\mu_i$  es la media de los valores del conjunto de entrenamiento para la característica  $i$

$s$ : Vector fila de  $n$  elementos, donde  $s_i$  es la desviación estándar de los valores en el conjunto de entrenamiento para la característica  $i$

$num\_iter$ : Número de iteraciones

$\theta, theta$ : Vector columna con los parámetros del modelo

$\alpha, alpha$ : Tasa de aprendizaje

$J(\theta), J$ : Resultado de la función de costo

$grad$ : Vector con derivadas parciales de la función de costo con respecto a cada uno de los parámetros  $\theta_i$

$J\_historia$ : Un vector columna con los valores de función de costo al final de cada iteración. Es decir,  $J\_historia(1)$  es el costo  $J(\theta)$  al final de la primera iteración,  $J\_historia(2)$  es el costo  $J(\theta)$  al final de la segunda iteración, y así sucesivamente.

$J_{train}$ : Error de entrenamiento

$J_{cv}$ : Error de validación cruzada

$J_{test}$ : Error de prueba

$h_{\theta}(x^{(i)})$ : Hipótesis para el  $i$ -ésimo ejemplo de entrenamiento

$.^{\wedge}$ : un operador que eleva a una potencia cada uno de los elementos de un vector

$sum$ : una función que retorna la suma de los elementos de un vector

$g, sigmoide$ : Función sigmoide,  $g(z) = \frac{1}{1+e^{-z}}$

## 3 Procedimiento

### 3.1 Lectura de los datos

En la carpeta *lab5*, suministrada para la elaboración del laboratorio, se ha incluido el archivo *lab5datos.txt*, que contiene los datos simulados de 980 estudiantes. Cada registro contiene 3 valores separados por comas, que representan lo siguiente:

- Característica  $x_1$
- Característica  $x_2$
- Variable a predecir  $y$

En este laboratorio se partirán los datos del archivo *lab5datos.txt* en tres conjuntos:

- Conjunto de entrenamiento, con el primer 60% de los registros
- Conjunto de validación cruzada, con el siguiente 20% de los registros
- Conjunto de prueba, con el último 20% de los registros

Antes de extraer los tres conjuntos, los ejemplos deben ordenarse aleatoriamente. Sin embargo, en el presente laboratorio se omitirá este proceso, debido a que ya se encuentran en un orden aleatorio.

Los arreglos correspondientes a los tres conjuntos son los siguientes:

$XI$ : Matriz de  $m$  filas con los valores de las características en el conjunto de entrenamiento

$y$ : Vector columna de  $m$  elementos con los valores para la variable a predecir en el conjunto de entrenamiento

$XI_{cv}$ : Matriz de  $m_{cv}$  filas con los valores de las características en el conjunto de validación cruzada

$y_{cv}$ : Vector columna de  $m_{cv}$  elementos con los valores para la variable a predecir en el conjunto de validación cruzada

$XI_{test}$ : Matriz de  $m_{test}$  filas con los valores de las características en el conjunto de prueba

$y_{test}$ : Vector columna de  $m_{test}$  elementos con los valores para la variable a predecir en el conjunto de prueba

La siguiente función Octave lee el archivo de datos *lab5datos.txt* y divide estos datos en los tres conjuntos requeridos:

```

function [XI, y, XIcv, ycv, XItest ytest] = leerDatos()
    d = load('lab5datos.txt');

    % TO DO
    % Implemente a continuación la inicialización de los arreglos
    % XI, y, XIcv, ycv, XItest y ytest

    ...

    %
end

```

Su tarea es terminar de implementar la anterior función, para lo cual basta completar la parte indicada por los puntos suspensivos.

Puede probar la función ejecutando el programa *lab5a.m*, incluido en la carpeta *lab5*. Debe obtener la salida de la Figura 1.

**Figura 1. Prueba de la función que lee los datos del archivo de texto**

```

DATOS

Primeros dos ejemplos del conjunto de entrenamiento
    x1    x2    y
    4.6    3.1    1
    2.8    3.2    1
Primeros dos ejemplos del conjunto de validacion cruzada
    x1    x2    y
    2.9    2.4    1
    0.5    1.9    0
Primeros dos ejemplos del conjunto de prueba
    x1    x2    y
    1.7    4.1    0
    1.1    2.3    0

x1: Calificacion de Ingenieria de Software I (0.0-5.0)
x2: Calificacion de Lenguaje de Programacion (0.0-5.0)
y: Registro Interaccion Humano-Computador, HCI? (0/1)

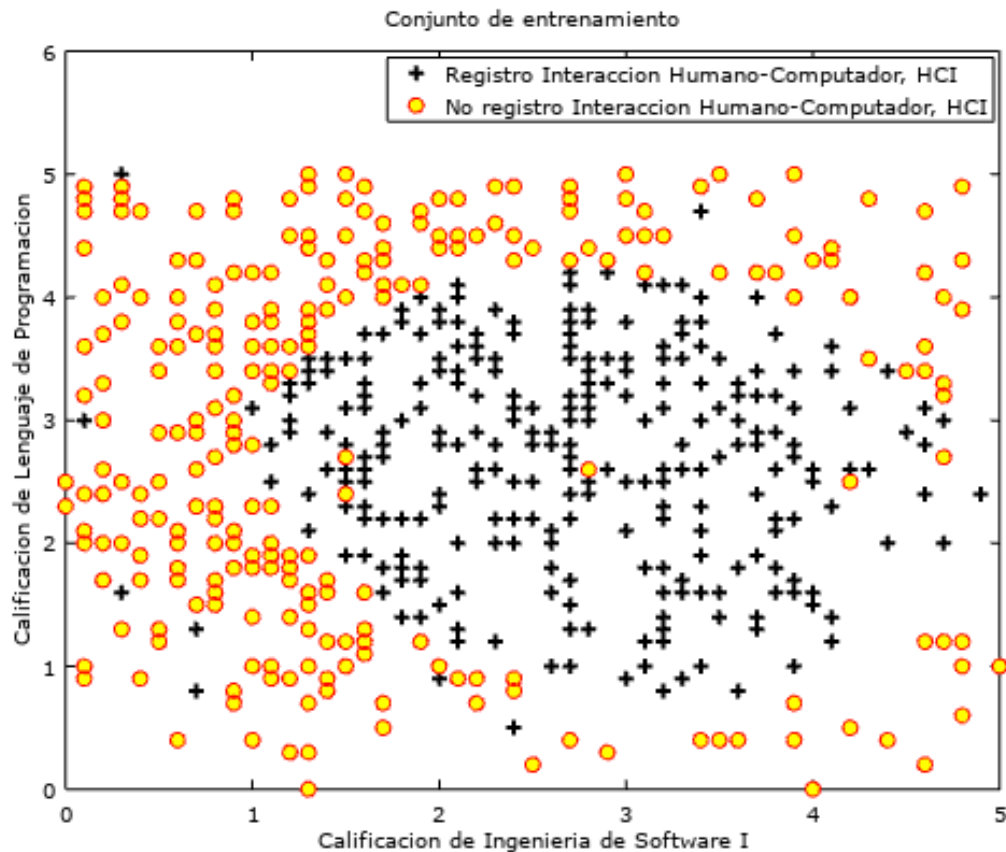
Total ejemplos (estudiantes): 980

```

### 3.2 Gráfica de los datos

En el archivo *lab5\graficarDatos.m*, ya se encuentra implementada la función para graficar los datos. Si ejecuta en este momento el programa *lab5a.m*, deberá obtener la gráfica de la Figura 2. En esta figura se observa que una frontera de decisión lineal no es adecuada para clasificar los ejemplos. Esta situación hará que se requiera utilizar características polinomiales.

Figura 2. Gráfica del conjunto de entrenamiento



*Nota:* No se le solicita realizar ninguna actividad en este punto.

### 3.3 Características polinomiales

En el código inicial, suministrado en la carpeta *lab5*, ya se incluyó la función *mapearCaracteristicas.m*. Esta función permite adicionar las características polinomiales a los conjuntos de entrenamiento, validación cruzada y prueba. La implementación proviene del curso de Aprendizaje de Máquina de la Universidad de Stanford disponible en [www.coursera.org](http://www.coursera.org), y es la siguiente:

```

function XP = mapearCaracteristicas(XI,grado)
% Agrega características polinomiales
% Parámetros:
%     XI: matriz de dos columnas de características, un ejemplo por fila
%     grado: grado del polinomio
% Retorno
%     XP: una nueva matriz de características con más características,
%     que comprenden:
%     X1, X2, X1.^2, X1.*X2, X2.^2, X1.^3, X1.^2.*X2 etc..
%     donde X1 es la primera columna de XI y X2, la segunda
%
    m=size(XI,1);
    X1=XI(:,1);
    X2=XI(:,2);
    XP = XI;
    for i = 2:grado
        for j = 0:i
            XP(:, end+1) = (X1.^(i-j)).*(X2.^(j));
        end
    end
end
end

```

Si se ejecuta el programa lab5a.m, en este momento, debe obtener la siguiente salida.

**Figura 3. Primer ejemplo de cada conjunto con las correspondientes características polinomiales**

#### CARACTERISTICAS POLINOMIALES

Grado del polinomio: 4

Primer estudiante del conjunto de entrenamiento

4.60	3.10	21.16	14.26	9.61
97.34	65.60	44.21	29.79	447.75
301.74	203.35	137.04	92.35	

Primer estudiante del conjunto de validacion cruzada

2.90	2.40	8.41	6.96	5.76
24.39	20.18	16.70	13.82	70.73
58.53	48.44	40.09	33.18	

Primer estudiante el conjunto de prueba

1.70	4.10	2.89	6.97	16.81
4.91	11.85	28.58	68.92	8.35
20.14	48.58	117.17	282.58	



Observe que para el primer estudiante del conjunto de entrenamiento:

- $x_3 = x_1^2 = 4.6^2 = 21.16$
- $x_4 = x_1 x_2 = 4.6 \cdot 3.1 = 14.26$
- $x_5 = x_2^2 = 3.1^2 = 9.61$  Etc.

*Nota:* No se le solicita realizar ninguna actividad en este punto.

### 3.4 Normalización de características

El siguiente paso es normalizar los valores de las características del conjunto de entrenamiento y obtener su media y su desviación estándar. Luego, con esta media y esta desviación estándar, se debe proceder a normalizar los valores de las características del conjunto de validación cruzada y del conjunto de prueba.

La siguiente función realiza lo anteriormente descrito:

```
Function [XN, XNcv, XNtest, mu, sigma] = normalizarConjuntos(XP, XPcv,
                                                             XPtest)

    XN = XP;
    XNcv = XPcv;
    XNtest = XPtest;
    mu=mean(XP);
    sigma=std(XP);

    m=size(XP,1);
    for i=1:m;
        XN(i,:) = (XP(i,:) - mu)./sigma;
    end;

    % TO DO
    % Implemente a continuación las sentencias necesarias para que
    % las matrices XNcv y XNtest queden inicializadas respectivamente
    % con las matrices XPcv y XPtest normalizadas

    ...

    %
end
```

La función retorna:

$XN$ : Matriz resultado de normalizar la matriz  $XP$ , que se ha inicializado previamente

$XN_{cv}$ : Matriz resultado de normalizar la matriz  $XP_{cv}$ , que se ha inicializado previamente

$XN_{test}$ : Matriz resultado de normalizar la matriz  $XP_{test}$ , que se ha inicializado previamente

$\mu$ : Vector fila de  $n$  elementos, donde  $\mu_i$  es la media de los valores del conjunto de entrenamiento para la característica  $i$

$s$ : Vector fila de  $n$  elementos, donde  $s_i$  es la desviación estándar de los valores en el conjunto de entrenamiento para la característica  $i$

Su tarea es terminar de implementar la anterior función, para lo cual basta completar la parte indicada por los puntos suspensivos.

Puede probar la función ejecutando el programa *lab5a.m*, debiendo obtener la salida de la Figura 4

**Figura 4. Primer ejemplo de cada conjunto con los valores de las características normalizados**

Características normalizadas:

Primer estudiante del conjunto de entrenamiento

1.9441	0.2235	2.4573	1.7798	0.0214	2.8089	2.4772	1.1790
-0.1487	3.0428	2.9101	1.8611	0.6827	-0.2710		

Primer estudiante del conjunto de validacion cruzada

0.5673	-0.3562	0.3404	0.1688	-0.5479	0.1196	0.1266	-0.1993
-0.6301	-0.0487	0.0026	-0.1344	-0.3938	-0.6450		

Primer estudiante el conjunto de prueba

-0.4045	1.0516	-0.5761	0.1710	1.0862	-0.5984	-0.3048	0.3957
1.0313	-0.5601	-0.4564	-0.1326	0.4620	0.9312		

medias de las características normalizadas:

-0.00	-0.00	-0.00	0.00	-0.00	-0.00	-0.00	0.00
0.00	-0.00	0.00	-0.00	-0.00	-0.00		

Desviaciones estandar de las características normalizadas:

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00		

### 3.5 Agregar columna de unos

Entre los archivos del laboratorio, se suministra la función *agregarUnos.m*, que permite agrega la columna de unos a la izquierda las matrices  $XN$ ,  $XN_{cv}$  y  $XN_{test}$ , obteniéndose las siguiente matrices:

$X$ : Matriz resultado de agregar una columna de unos a la izquierda de la matriz  $XN$

$X_{cv}$ : Matriz resultado de agregar una columna de unos a la izquierda de la matriz  $XN_{cv}$

$X_{test}$ : Matriz resultado de agregar una columna de unos a la izquierda de la matriz  $XN_{test}$

Si ejecuta *lab5a.m* en este momento, deberá obtener la salida de la Figura 5, donde se puede observar el uno de conveniencia de notación en cada uno de los ejemplos mostrados.

Figura 5. Primer ejemplo de cada uno de los tres conjuntos de datos

AGREGAR COLUMNA DE UNOS							
Primer estudiante del conjunto de entrenamiento							
1.0000	1.9441	0.2235	2.4573	1.7798	0.0214	2.8089	2.4772
1.1790	-0.1487	3.0428	2.9101	1.8611	0.6827	-0.2710	
Primer estudiante del conjunto de validacion cruzada							
1.0000	0.5673	-0.3562	0.3404	0.1688	-0.5479	0.1196	0.1266
-0.1993	-0.6301	-0.0487	0.0026	-0.1344	-0.3938	-0.6450	
Primer estudiante el conjunto de prueba							
1.0000	-0.4045	1.0516	-0.5761	0.1710	1.0862	-0.5984	-0.3048
0.3957	1.0313	-0.5601	-0.4564	-0.1326	0.4620	0.9312	

*Nota:* No se le solicita realizar ninguna actividad en este punto.

### 3.6 Función sigmoide

La función sigmoide está definida como:  $g(z) = \frac{1}{1+e^{-z}}$

Entre los archivos del laboratorio, se ha suministrado el archivo *sigmoide.m* que implementa la anterior función. El programa *lab5a.m* realiza una prueba a la anterior función, desplegando la salida de la Figura 6.

Figura 6. Función sigmoide

FUNCION SIGMOIDE $g(z)$	
$z$	$g(z)$
-10.00	0.000045
-3.00	0.047426
-1.00	0.268941
-0.50	0.377541
0.00	0.500000
0.50	0.622459
1.00	0.731059
3.00	0.952574
10.00	0.999955

*Nota:* No se le solicita realizar ninguna actividad en este punto.

### 3.7 Predicciones provisionales para todo el conjunto de entrenamiento

La fórmula para realizar las predicciones para todo el conjunto de entrenamiento es:

$$h = g(X \cdot \theta)$$

Esta fórmula ya se encuentra implementada en una función Octave contenida en *lab5/prediccionesConjunto.m*.

Puede probar la función ejecutando programa *lab5a.m*, debiendo obtener la salida de la Figura 7

Figura 7. Predicciones provisionales para todo el conjunto de entrenamiento

```
PREDICCIONES PROVISIONALES PARA EL CONJUNTO DE ENTRENAMIENTO
```

```
Con parametros theta provisionales
```

```

y      h
1      0.8470
1      0.5435
0      0.7888
...    ...
```

```
y: Registro Interaccion Humano-Computador, HCI? (0/1)
```

```
h: Prediccion de probabilidad de que registre Interaccion Humano-Computador, HCI
```

```
Nota: Las predicciones anteriores no son correctas,
mas adelante se obtendran predicciones correctas,
cuando se usen los parametros calculados por el algoritmo de aprendizaje
correctamente ajustado.
```

Como se aclara en la figura, las predicciones no son correctas, debido a que en este momento del laboratorio aún no se ha aprendido un modelo correcto.

*Nota:* No se le solicita realizar ninguna actividad en este punto.

### 3.8 Función de costo con regularización

El algoritmo de aprendizaje de la regresión logística minimiza una función, denominada función de costo. Entre menor sea el valor de esta función, mejor el ajuste que la hipótesis hace de los datos. La función de costo está definida por:

$$J(\theta) \leftarrow \frac{1}{m} \sum_{i=1}^m \text{cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

donde

$$\text{cost}(h_{\theta}(x), y) \leftarrow \begin{cases} -\log(h_{\theta}(x)) & \text{si } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{si } y = 0 \end{cases}$$

A la definición anterior, se le puede agregar el término de regularización, para que el algoritmo aprenda un modelo que generalice mejor. La siguiente es la fórmula para agregar el término de regularización.

$$J(\theta) \leftarrow J(\theta) + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2 \quad (1)$$

Note que  $\theta_0$  no está incluido en la sumatoria. Además, recuerde que  $\theta_0$  corresponde en Octave es `theta(1)`.

La siguiente función Octave implementa la función de costo con regularización, sin usar sentencias `for`:

```
Function J = costo(y, h, theta, lambda)
    m = length(y);
    h1=h(find(y==1));
    c1=-log(h1);
    h0=h(find(y==0));
    c2=-log(1-h0);
    J=1/m*sum([c1; c2]);

    % TO DO
    % Implemente a continuación la adición del
    % término de regularización a la función de costo, fórmula (1).

    ...

    %
end
```

Su tarea es terminar de implementar la anterior función, para lo cual basta completar la parte indicada por los puntos suspensivos. Como es frecuente, no se requiere utilizar la sentencia *for* para esta implementación. Puede probar la función ejecutando el programa *lab5a.m*, debiendo obtener la salida de la Figura 8.

Figura 8. Prueba de la implementación de la función de costo, con parámetros provisionales.

```
FUNCION DE COSTO CON REGULARIZACION, CON PARAMETROS PROVISIONALES

Resultado de la funcion de costo con parametros provisionales:    0.797168
Otro resultado de la funcion de costo con parametros provisionales:  0.797235
```

El valor de la función costo en la figura no tiene ningún significado especial en el laboratorio, es solo una prueba de implementación.

### 3.9 Descenso por el gradiente con regularización

El algoritmo de aprendizaje tiene la responsabilidad de calcular unos parámetros  $\theta_i$  que minimicen la función de costo. A continuación se presenta el pseudocódigo del algoritmo, en forma vectorizada y con unos ajustes para su seguimiento:

$$\theta \leftarrow \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

$$h \leftarrow g(X\theta)$$

Para  $i \leftarrow 1$  hasta  $num\_iter$

$$grad \leftarrow \frac{1}{m}((h - y)^T X)^T$$

$$reg \leftarrow \frac{\lambda}{m} * \theta \quad (1)$$

$$reg_1 \leftarrow 0 \quad (2)$$

$$grad \leftarrow grad + reg \quad (3)$$

$$\theta \leftarrow \theta - \alpha \cdot grad$$

$$h = g(X\theta)$$

$$J\_historia_i \leftarrow \frac{1}{m} \sum_{i=1}^m cost(h_\theta(x^{(i)}), y^{(i)})$$

La siguiente función Octave implementa el anterior algoritmo:

```
function [theta, J_historia] = descensoGradiente(X, y, alpha, num_iter,
lambda)
    m = length(y);
    n = size(X,2)-1;
    J_historia = zeros(num_iter, 1);
    theta = zeros(n+1, 1);
    h=prediccionesConjunto(X,theta);
    for i = 1:num_iter
        grad=1/m*((h-y)'*X)';

        % TO DO
        % Implemente a continuación la regularización

        ...

        %
        theta=theta-alpha*grad;
        h=prediccionesConjunto(X,theta);
        J_historia(i) = costo(y, h, theta, lambda);
        deci=floor(num_iter/10);
        if( mod(i,deci)==0)
            fprintf("%4d%%",floor(i/num_iter*100)); fflush(stdout);
        endif
    endfor
endfunction
```

Su tarea es terminar de implementar la anterior función, para lo cual basta completar la parte indicada por los puntos suspensivos. Como es usual, no se requiere ninguna sentencia *for* para esta implementación. Para probar la función, una vez sea completada, ejecute el programa *lab5a.m*, debiendo obtener la salida de la Figura 9

Figura 9. Prueba de la implementación del algoritmo de aprendizaje.

```
DESCENSO POR EL GRADIENTE CON REGULARIZACION

Alpha=      1.000000
Numero de iteraciones= 100000
lambda= 0.00
Grado del polinomio=4

Theta encontrado por descenso por el gradiente:
  0.667085
 -5.287953
 -0.242008
 13.342445
 12.628088
  2.864502
 -7.768514
  6.918382
  0.247650
 -6.078513
 -1.430923
-11.214554
  3.882849
-15.064742
  4.371617

# de iteracion    COSTO, J(theta)
      0           0.693147
      1           0.603628
      2           0.579501
      3           0.558202
      4           0.538994
      5           0.521716
      6           0.506064
      7           0.491863
      8           0.478940
      9           0.467150
     10           0.456369
    99991         0.214148
    99992         0.214148
    99993         0.214148
    99994         0.214148
    99995         0.214148
    99996         0.214148
    99997         0.214148
    99998         0.214148
    99999         0.214148
   100000         0.214148

Error de entrenamiento:    0.214148
Error de validacion cruzada: 0.185203
```



En la Figura 8, también se observa el error de entrenamiento y el error de validación cruzada.

El error de entrenamiento, que se calcula con base en el conjunto de entrenamiento y está dado por la siguiente fórmula.

$$J_{train} \leftarrow \frac{1}{m} \sum_{i=1}^m cost(h_{\theta}(x^{(i)}), y^{(i)})$$

donde

$$cost(h_{\theta}(x), y) \leftarrow \begin{cases} -\log(h_{\theta}(x)) & \text{si } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{si } y = 0 \end{cases}$$

El error de validación cruzada está dado por la siguiente fórmula:

$$J_{cv} \leftarrow \frac{1}{m_{cv}} \sum_{i=1}^{m_{cv}} cost(h_{\theta}(x_{cv}^{(i)}), y_{cv}^{(i)})$$

donde

$$cost(h_{\theta}(x_{cv}), y_{cv}) \leftarrow \begin{cases} -\log(h_{\theta}(x_{cv})) & \text{si } y_{cv} = 1 \\ -\log(1 - h_{\theta}(x_{cv})) & \text{si } y_{cv} = 0 \end{cases}$$

No se requiere implementación adicional para calcular el error de entrenamiento y error de validación cruzada, simplemente se reutilizan las funciones *prediccionConjunto.m* y *costo.m*.

Note que el error de entrenamiento es menor que la función de costo. Esto es usual, porque en el cálculo del error de entrenamiento, no se adiciona el término de regularización.

### 3.10 Ajuste del parámetro de regularización $\lambda$

En esta sección se le solicitará ajustar el parámetro de regularización con el objetivo de obtener un modelo que generalice mejor. Se dice que un modelo generaliza bien, si hace predicciones correctas con ejemplos nuevos, es decir con ejemplos que no han sido usados en la construcción ni en el ajuste del modelo.

Para ajustar el parámetro de regularización, realice los siguientes pasos:

- a) Pruebe el parámetro de regularización  $\lambda = 0$ . Para ello siga las instrucciones que se indican a continuación.
  - i) En el programa *lab5a.m*, al comienzo, cambie la instrucción de asignación que inicializa la variable  $\lambda$  por:
 

```
lambda=0;
```
  - ii) Seguidamente, ejecute el programa *lab5a.m*

iii) Diligencie el primer renglón de la siguiente tabla con los resultados obtenidos

<i>Parámetro de regularización</i> $\lambda$	<i>Error de entrenamiento</i> $J_{train}$	<i>Error de validación cruzada</i> $J_{cv}$
0.00		
0.01		
0.02		
0.04		
0.08		
0.16		
0.32		

- b) Realice nuevamente las instrucciones explicadas en el paso a), pero con  $\lambda = 0.01$  y diligencie el segundo renglón de la tabla.
- c) Realice nuevamente las instrucciones explicadas en el paso a), pero con  $\lambda = 0.02, 0.04, 0.08, 0.16, 0.32$ . Diligencie los restantes renglones de la tabla.
- d) En el programa *lab5a.m*, al comienzo, ajuste la sentencia que inicializa  $\lambda$ , de manera que a  $\lambda$  se le asigne el mejor parámetro de regularización de los probados. Este será aquel que produzca el error de validación cruzada más bajo.

### 3.11 Ajuste del grado del polinomio d

Antes de realizar el procedimiento explicado en esta sección, asegúrese de que en el programa *lab5a.m*,  $\lambda$  se encuentra inicializado con su mejor valor, encontrado de acuerdo con la sección 3.10.

En esta sección se le solicitará ajustar el grado del polinomio. con el objetivo de obtener un modelo que generalice aún mejor.

Para ajustar el grado del polinomio, realice los siguientes pasos:

- a) Pruebe el grado de polinomio 1, es decir pruebe  $d = 1$ . Para ello siga las instrucciones que se indican a continuación.
  - i) En el programa *lab5a.m*, al comienzo, cambie la instrucción de asignación que inicializa la variables  $d$  por  
 $d=1;$
  - ii) Seguidamente, ejecute el programa *lab5a.m*
  - iii) Diligencie el primer renglón de la siguiente tabla con los resultados obtenidos

<i>Grado del polinomio</i>	<i>Error de entrenamiento</i> $J_{train}$	<i>Error de validación cruzada</i> $J_{cv}$
1		
2		
3		
4		
5		

- b) Realice nuevamente las instrucciones explicadas en el paso a), pero con  $d = 2$  y diligencie el segundo renglón de la tabla.
- c) Realice nuevamente las instrucciones explicadas en el paso a), pero con  $d = 3$ ,  $d = 4$  y  $d = 5$  y diligencie los restantes renglones de la tabla.
- d) En el programa *lab5a.m*, al comienzo, ajuste la sentencia que inicializa  $d$ , para que a  $d$  se le asigne el mejor grado de polinomio de los probados. Este será aquel que produzca el error de validación cruzada más bajo.

### 3.12 Predicciones para el conjunto de entrenamiento

A partir de este momento, las funciones se prueban con el programa *lab5b.m*, no con *lab5a.m*

En el código inicial, suministrado en la carpeta *lab5*, ya se incluyó la implementación de la función *prediccionesConjunto()*, que calcula las predicciones del modelo para todo el conjunto de entrenamiento. Es así, como se ejecuta en este momento el programa *lab5b.m*, deberá poder observar en la interfaz con las predicciones correctas para los ejemplos de entrenamiento. Véase la Figura 10.

Figura 10. Predicciones para algunos ejemplos del conjunto de entrenamiento

PREDICCIONES PARA LOS EJEMPLOS DE ENTRENAMIENTO

y	h
1	0.7057
1	0.9963
0	0.5491
0	0.0633
1	0.9513
0	0.0203
0	0.0479
0	0.0136
0	0.0604
0	0.0610
...	...

y: Registro Interaccion Humano-Computador, HCI? (0/1)

h: Prediccion de probabilidad de que registre Interaccion Humano-Computador, HCI

*Nota:* No se le solicita realizar ninguna actividad en este punto.

### 3.13 Predicciones para nuevos ejemplos

El objetivo del aprendizaje de máquina es crear un modelo para hacer predicciones. Recuerde que la fórmula para calcular una predicción mediante la regresión logística es  $h_{\theta}(x) = g(\theta^T x)$ , previa adición de las características polinomiales y normalización de los valores de las características.

La siguiente función permite hacer predicciones para nuevos ejemplos, cuyas características pueden ser ingresadas por el usuario por consola:

```

function predicciones( theta, mu, sigma, d)
continuar=input("Continuar(s/n):","s"); fflush(stdout);
if(length(continuar)==0)
    continuar="n";
endif
while (continuar=='s')
    x1=input(...
        "\nIngrese la calificacion de Ingenieria de Software I (0.0-5.0): ");
    if(length(x1)==0)
        x1=0;
    endif
    x2=input(...
        "\nIngrese la calificacion de Lenguaje de Programacion (0.0-5.0): ");
    if(length(x2)==0)
        x2=0;
    endif
    xi=[x1 x2];

    % TO DO
    % implemente aquí la adición de las características polinomiales
    % hasta un grado d al vector xi, para obtener el vector xp

    ...

    %
    xn=(xp-mu)./sigma;
    x=[1; xn'];
    h = sigmoide(theta'*x);
    fprintf( "\nLa predicción de probabilidad de que registre\n");
    fprintf("Interaccion Humano-Computador, HCI, es:\n%6.4f\n\n", h);
    continuar=input("Continuar(s/n):","s"); fflush(stdout);
    if(length(continuar)==0)
        continuar="n";
    endif
endwhile
endfunction

```

En la Figura 11, puede observarse el prototipo de la interfaz correspondiente a la anterior función.

**Figura 11. Prueba de la función para realizar predicciones para nuevos ejemplos**

```
PREDICCIONES PARA NUEVOS EJEMPLOS

Continuar(s/n):s

Ingrese la calificacion de Ingenieria de Software I (0.0-5.0): D.D
Ingrese la calificacion de Lenguaje de Programacion (0.0-5.0): D.D
La prediccion de probabilidad de que registre
Interaccion Humano-Computador, HCI, es:
D.DDDD

Continuar(s/n):s

Ingrese la calificacion de Ingenieria de Software I (0.0-5.0): D.D
Ingrese la calificacion de Lenguaje de Programacion (0.0-5.0): D.D
La prediccion de probabilidad de que registre
Interaccion Humano-Computador, HCI, es:
D.DDDD

Continuar(s/n):n
```

Su tarea en este punto es terminar de implementar la función anterior, para lo cual basta completar la parte indicada con puntos suspensivos.

Puede asumir que se ingresarán al programa valores válidos, ya que esta no es una herramienta para el usuario final, sino un programa para experimentación por parte del experto en aprendizaje de máquina.

Pruebe su función ejecutando el programa *lab5b.m*. Deberá obtener predicciones de acuerdo a la tendencia de los ejemplos del conjunto de entrenamiento.

### 3.14 Exactitudes y error de prueba

Las fórmulas para la exactitud del entrenamiento, la exactitud de validación cruzada y la exactitud de prueba son las siguientes:

$$E_{train} = \frac{\text{Número de aciertos del modelo con el conjunto de entrenamiento}}{m} * 100\%$$
$$E_{cv} = \frac{\text{Número de aciertos del modelo con el conjunto de validación cruzada}}{m_{cv}} * 100\%$$

$$E_{test} = \frac{\text{Número de aciertos del modelo con el conjunto de prueba}}{m_{test}} * 100\%$$

Entre los archivos del laboratorio, se suministra el archivo *lab5/calcularExactitud.m*, que permite calcular  $E_{train}$ ,  $E_{cv}$  y  $E_{test}$ .

Si en este momento ejecuta el programa *lab5b.m*, Deberá obtener un salida por consola que muestre la tres exactitudes por encima del 94%. Adicionalmente, se mostrará el error de prueba inferior a 0.25.

Lo usual es que la exactitud de prueba sea la menor de las tres exactitudes, ya que el conjunto de prueba no se ingresa al algoritmo de aprendizaje, ni se usa para el ajuste de parámetro de regularización  $\lambda$  ni del grado del polinomio  $d$ . En el presente laboratorio se cumple esta relación.

La exactitud de prueba es un elemento clave para determinar si el modelo podrá ser útil o no en un futuro. Si la exactitud de prueba es baja, el modelo no debe utilizarse todavía para hacer predicciones, así la exactitud de entrenamiento y la exactitud de validación cruzada sean altas.

También aparece en la salida por consola el error de prueba, que estima el error que el modelo cometerá en predicciones con nuevos ejemplos. Está dado por la siguiente fórmula:

$$J_{test} \leftarrow \frac{1}{m_{test}} \sum_{i=1}^m \text{cost} \left( h_{\theta} \left( x_{test}^{(i)} \right), y_{test}^{(i)} \right)$$

donde

$$\text{cost}(h_{\theta}(x_{test}), y_{test}) \leftarrow \begin{cases} -\log(h_{\theta}(x_{test})) & \text{si } y_{test} = 1 \\ -\log(1 - h_{\theta}(x_{test})) & \text{si } y_{test} = 0 \end{cases}$$

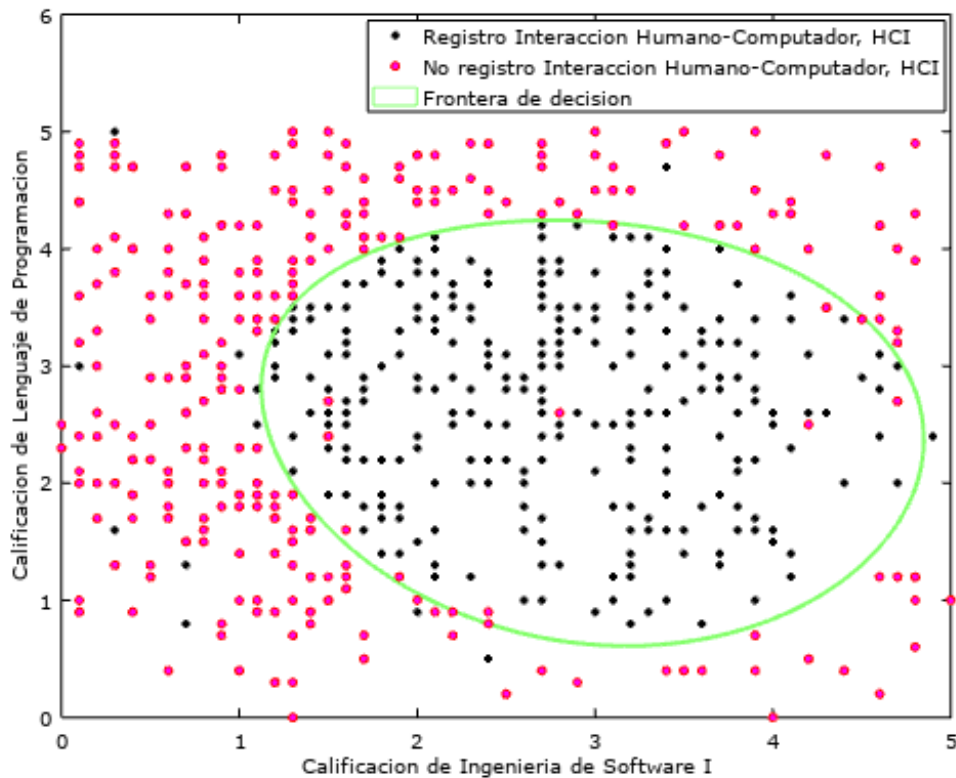
No se requiere implementación adicional para calcular el error de prueba. Simplemente se reutilizan las funciones *prediccionesConjunto.m* y *costo.m*, esta última con un parámetro de regularización  $\lambda = 0$ .

*Nota:* No se le solicita realizar ninguna implementación en este punto.

### 3.15 Frontera de decisión

En el código inicial, suministrado en la carpeta *lab5*, ya se incluyó la implementación de la gráfica de la frontera de decisión. Es así como, si en este momento se ejecuta el programa *lab5b.m*, se obtiene la gráfica la de la Figura 12. Se observa que la frontera no es lineal.

Figura 12. Gráfica de la frontera de decisión



Nota: No se le solicita realizar ninguna actividad en este punto.

### 3.16 Curvas de aprendizaje

Las curvas de aprendizaje permiten diagnosticar si el modelo es correcto, es decir, si no tiene el problema de alta tendencia o el problema de alta varianza. Estas curvas resultan de graficar el error de entrenamiento y el error de validación cruzada contra el número de ejemplos, digamos  $m = 1, 2, 3, \dots, 100$ . Por lo tanto, se debe ejecutar el algoritmo de aprendizaje varias veces, una vez por cada uno de estos valores de  $m$ , para hallar los correspondientes errores y puntos en la gráfica.

Entre los archivos suministrados para el laboratorio, ya se encuentra implementada la función que elabora las curvas de aprendizaje, archivo *curvasAprendizaje.m*. Puede probar la función ejecutando el programa *lab5b.m*, debiendo obtener la gráfica de la Figura 13.

Nota: La ejecución de esta función para el número de ejemplos de entrenamiento variando de 1 a 50 puede tomar más de 30 minutos.



Figura 13. Curvas de aprendizaje



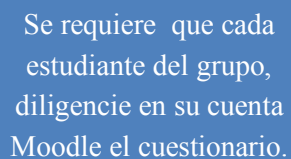
En la gráfica se observar el comportamiento de las curvas de aprendizaje de un modelo correcto. El error de entrenamiento empieza en cero y sube un poco, pero se mantiene bajo. El error de validación cruzada empieza con valores altos, descendiende rápidamente y se mantiene un poco mayor que el error de entrenamiento. Entre más se incremente el número de ejemplos, más se acercará el error de validación cruzada al error de entrenamiento.

*Nota:* No se le solicita realizar ninguna actividad en este punto.

## 4 Evaluación - Cuestionario Moodle

Para la evaluación, Cada uno de los estudiantes del grupo deberá diligenciar un cuestionario Moodle sobre el laboratorio, pudiendo en este proceso recibir ayuda de sus compañeros de equipo.

Inmediatamente diligencien el cuestionario, aparecerá la nota del laboratorio. Pueden hacer un número ilimitado de intentos, dentro de plazo fijado, quedando como nota de laboratorio la nota más alta obtenida.

A blue rounded rectangle box with a thin dark blue border, containing white text.

Se requiere que cada  
estudiante del grupo,  
diligencie en su cuenta  
Moodle el cuestionario.

[Verificado LHR 2018-04-18]