



Universidad del Quindío
Programa Ingeniería de Sistemas y Computación
Asignatura Inteligencia Artificial

Guía de laboratorio

Profesor: Leonardo Hernández R.

Semestre académico: Primer semestre académico de 2018

Laboratorio	5-Regresión Logística
Estudiantes por grupo	Máximo 3
Tiempo promedio por estudiante estimado	6h

1 Objetivo

Uno de los temas más destacados en orientación profesional es la asesoría a los estudiantes en la elección de carrera que deben escoger. En el presente laboratorio apunta en dirección a este tema.

El objetivo del laboratorio es:

Implementar una regresión logística para predecir si se recomienda a un estudiante cursar la carrera de Ingeniería de Sistemas y Computación o no cursarla.

La predicción se basará en las siguientes características:

- x_1 : El puntaje de una prueba de razonamiento matemático(0.0 a 5.0)
- x_2 : El puntaje de una prueba de razonamiento verbal(0.0 a 5.0)

Para el aprendizaje del modelo se utilizarán datos sintetizados de 910 estudiantes, que serán suministrados con los archivos del laboratorio.

Adicionalmente, se suministran para el laboratorio una guía, en formato PDF, y una plantilla en formato Word. En la guía se les solicita a los estudiantes que realicen

diferentes actividades, consistentes principalmente en la elaboración de código Octave y en el diligenciamiento de los resultados del experimento en la plantilla.

La plantilla debe ser diligenciada con los resultados del experimento que en ella se solicitan

2 Convenciones

En Octave utilizaremos las siguientes convenciones

m : Número de ejemplos de entrenamiento, en este caso $m = 910$

n : Número de características, en este caso $n = 2$

XI : Matriz de $m \times n$ elementos con las características de todos los ejemplos de entrenamiento

y : Vector columna de m elementos con los valores para la variable a predecir de todos los ejemplos de entrenamiento

XN : Matriz de $m \times n$ elementos, con las características normalizadas de todos los ejemplos de entrenamiento.

μ : Vector de n elementos con las medias de las características de los ejemplos de entrenamiento

σ : Vector de n elementos con las desviaciones estándar de las características de los ejemplos de entrenamiento

X : Matriz $m \times n + 1$ conformada por una columna de unos y , a su derecha, la matriz XN

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

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

α : Tasa de aprendizaje

num_iter : Número de iteraciones

θ : Vector columna con los parámetros del modelo

$J_historia$: Un vector columna con el valor de la función de costo antes de ejecutarse la primera iteración, y los valores de esta función al final de cada iteración. Es decir, $J_historia(1)$ es el costo $J(\theta)$ antes de iniciarse la ejecución de la primera iteración, $J_historia(2)$ es el costo $J(\theta)$ al final de la primera iteración, $J_historia(3)$ es el costo $J(\theta)$ al final de la segunda iteración, y así sucesivamente.

sigmoide : Función sigmoide $g(z)$

En la notación matemática utilizaremos las siguientes convenciones:

m : Número de ejemplos de entrenamiento

n : Número de características

X : Matriz de $m \times n + 1$ conformada por una columna de unos y a su derecha las características normalizadas de los ejemplos de entrenamiento.

y : Vector columna de m filas con los valores de la variable a predecir de todos los ejemplos de entrenamiento

$x^{(i)}$: 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 del i -ésimo ejemplo de entrenamiento.

μ : Vector fila de n elementos, donde μ_i es la media de la característica i de los ejemplos de entrenamiento

s : Vector fila de n elementos, donde s_i es la desviación estándar de la característica i de los ejemplos de entrenamiento

θ : Vector columna con los parámetros del modelo

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

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

$J(\theta)$: Función de costo

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

α : Tasa de aprendizaje

$.^{\wedge}$: 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 : Función sigmoide $g(x)$

3 Procedimiento

3.1 Lectura de los datos de entrenamiento

En la carpeta *lab4*, suministrada con los archivos del laboratorio, se ha incluido el archivo *lab4datos.txt*, que contiene el conjunto de entrenamiento para este laboratorio, con datos simulados de 893 estudiantes. Cada registro contiene 4 valores separados por comas, que corresponden respectivamente a:

- Código del estudiante
- Característica x_1
- Característica x_2
- Variable a predecir y

Su tarea es elaborar en la carpeta *lab4* una función que lea el archivo mencionado archivo. Las especificaciones de la función son las siguientes:

Nombre de la función:

leerDatos().

Parámetros: Ninguno

Retorno:

XI: Matriz de $m \times n$ elementos con las características de todos los ejemplos de entrenamiento

y: Vector columna de m elementos con los valores de la variable a predecir de todos los ejemplos de entrenamiento

Para probar la función, una vez implementada, ejecute el programa *lab4a.m*, incluido en la carpeta *lab4*. Debe obtener la salida de la Figura 1.

Figura 1. Prueba de la función que lee los datos de entrenamiento de un archivo de texto

DATOS DE ENTRENAMIENTO		
x1	x2	y
1.4	3.2	0
2.6	3.4	0
2.2	3.7	0
1.0	2.8	0
3.1	1.2	0
2.8	2.9	0
1.0	2.5	0
0.2	0.4	0
0.1	3.7	0
3.3	3.9	1
...
x1: Puntaje obtenido en prueba de razonamiento matematico(0.0-5.0)		
x2: Puntaje obtenido en prueba de razonamiento verbal(0.0-5.0)		
y: Se le recomienda estudiar Ingenieria de Sistemas y Computacion?(1/0)		
Total ejemplos de entrenamiento (estudiantes): 910		

3.2 Gráfica de los datos

En este punto debe elaborar una función para graficar en un plano cartesiano los datos suministrados. Las especificaciones de la función son las siguientes:

Nombre de la función:

graficarDatos()

Parámetros:

XI: Matriz de $m \times n$ elementos con las características de todos los ejemplos de entrenamiento

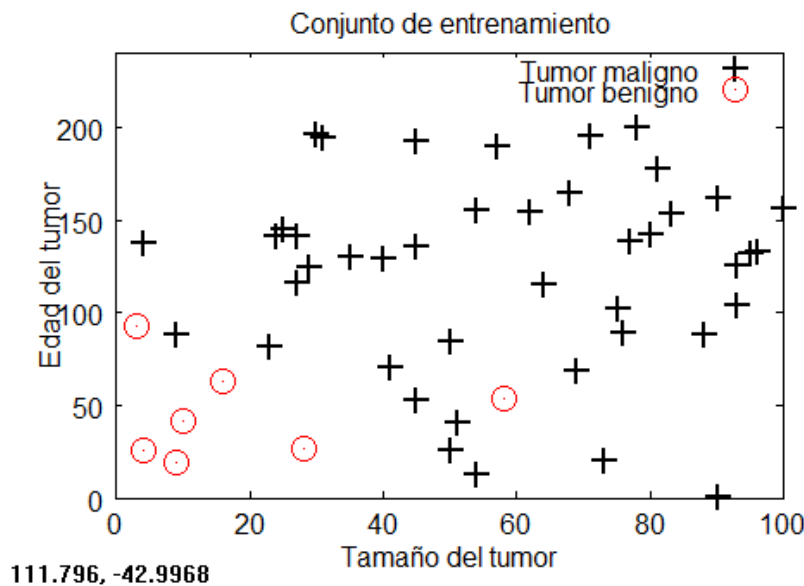
y: Vector columna de m elementos con los valores de la variable a predecir de todos los ejemplos de entrenamiento

Retorno:

Ninguno

Puede probar su función ejecutando el programa *lab4a.m*. Se debe obtener una gráfica del tipo del de la Figura 2, pero correspondiente al presente laboratorio.

Figura 2. Gráfica del conjunto de entrenamiento NO correspondiente al presente laboratorio



Notas de implementación

La función requiere dos instrucciones `plot`, una para graficar los signos más (+) y otra para graficar en forma superpuesta los círculos(o). A continuación se presenta el código correspondiente para la inicialización de dos vectores `x1p` y `x2p` requeridos para la primera de estas instrucciones `plot`.

```
% Inicializa el arreglo pos con las posiciones 'i' del arreglo 'y'
% tales que y(i)=1, es decir con los índices de los casos positivos
pos = find(y==1);
% Inicializa el vector x1p con los valores de la primera característica
% para los casos positivos
x1p=XI(pos, 1);
% Inicializa el vector x2p con las valores de la segunda característica
% para los casos positivos
x2p=XI(pos, 2);
```

3.3 Normalización de características

Seguidamente, debe proceder a la normalización de los datos, lo que reducirá enormemente el número de iteraciones requeridas por el algoritmo de aprendizaje.

Específicamente debe elaborar en la carpeta *lab4* una función con las siguientes especificaciones.

Nombre de la función:
normalizarConjunto()

Parámetros:

XI: Matriz de $m \times n$ elementos con las características de todos los ejemplos de entrenamiento

Retorno:

XN: Matriz de $m \times n$ elementos con los valores normalizados de *XI*

mu: Vector de n elementos con las medias de las características de todos los ejemplos de entrenamiento

sigma: Vector de n elementos con las desviaciones estándar de las características de los ejemplos de entrenamiento

Para probar la función, ejecute el programa *lab4a.m*, que producirá una salida donde se podrán observar entre otras cosas las medias y las desviaciones estándar de las características normalizadas. Si la normalización fue correcta, estas medias deben ser

todas iguales a cero, y las desviaciones estándar, todas iguales a uno. Lo anterior se debe al efecto de la normalización.

3.4 Agregar columna de unos

Seguidamente, debe elaborar en la carpeta *lab4* una función para agregar una columna inicial de unos a la matriz XN , los unos de conveniencia de notación. Las especificaciones de la función son las siguientes.

Nombre:

agregarUnos ()

Parámetro:

XN : Matriz de $m \times n$ elementos con los valores normalizados de la matriz XI , que es una matriz de $m \times n$ elementos con las características de todos los ejemplos de entrenamiento

Retorno:

X : Matriz de $m \times n + 1$, una columna de unos y, a su derecha, la matriz XN

Para probar la función, una vez implementada, ejecute el programa *lab4a.m*.

3.5 Función sigmoide

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

En este punto debe implementar la función anterior, con las siguientes especificaciones:

Nombre:

sigmoide ()

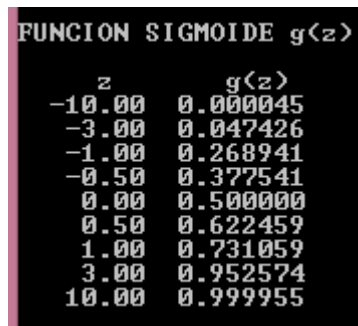
Parámetro:

z : un arreglo de números

Retorno:

g : Un vector con el mismo número de elementos del vector z , donde $g_i = \frac{1}{1+e^{-z_i}}$

Después de implementada la función, el programa *lab4a.m*, deberá mostrar la siguiente salida:



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

Recuerde que la función sigmoide varía entre 0 y 1, a medida que la variable independiente varía entre $-\infty$ y $+\infty$. En $z = 0$, la función toma un valor de 0.5. La salida anterior es ilustra este comportamiento.

3.6 Predicciones provisionales para todo el conjunto de entrenamiento

El siguiente paso es realizar una función que realice las predicciones para todo el conjunto de entrenamiento. Recuerde que la fórmula para realizar las predicciones para todo el conjunto de entrenamiento es:

$$h = g(X \cdot \theta) \text{ donde } g(z) = \frac{1}{1+e^{-z}}$$

Las especificaciones de la función son las siguientes.

Nombre:

prediccionesConjunto()

Parámetros:

X: Matriz de $m \times n + 1$, una columna de unos y, a su derecha, los valores normalizados de las características de todos los ejemplos de entrenamiento

theta: Parámetros del modelo

Retorno:

h: Vector columna de $m \times 1$ con las predicciones de probabilidad para todos los ejemplos de entrenamiento

En la implementación de esta función, es conveniente utilizar la función `sigmoide()` implementada previamente.

Puede probar la función, con parámetros provisionales, ejecutando el programa *lab4a.m*. Deberá obtener la salida de la Figura 3

Figura 3. Predicciones provisionales para todo el conjunto de entrenamiento

PREDICCIONES PROVISIONALES PARA TODO EL CONJUNTO DE ENTRENAMIENTO

Con parametros theta provisionales

Predicciones para todo el conjunto de entrenamiento

y	h
0	0.51
0	0.58
0	0.58
0	0.47
0	0.45
0	0.55
0	0.45
0	0.29
0	0.49
1	0.64
...	...

y: Se le recomienda estudiar Ingenieria de Sistemas y Computacion?(1/0)

h: Prediccion de probabilidad de que le sea recomendable estudiar Ingenieria de Sistemas y Computacion

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.

Presione enter para continuar:

Las predicciones no son correctas, debido a que se han realizado con parámetros provisionales. En un punto posterior, se obtendrán predicciones correctas, cuando de utilicen los parámetros aprendidos por el algoritmo correctamente ajustado.

3.7 Función de costo

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:

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

donde

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

Su tarea es implementar en la carpeta *lab4* la función de costo, con las siguientes especificaciones:

Nombre:

costo()

Parámetros:

y: Vector columna de *m* elementos con los valores de la variable a predecir de todos los ejemplos de entrenamiento

h: Vector columna de $m \times 1$ con las predicciones de probabilidad para todos los ejemplos de entrenamiento

Retorno:

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

Puede probar la función ejecutando el programa *lab4a.m*, debiendo obtener un costo de 0.607255. Más adelante en el laboratorio, después de que se implemente y se ajuste el algoritmo de aprendizaje, se obtendrá un valor mucho menor para esta función.

3.8 Descenso por el gradiente

El algoritmo de aprendizaje de la regresión logística minimiza la función de costo, utilizando descenso por el gradiente. A continuación se presenta este algoritmo para *n* características:

$\theta_0 \leftarrow 0$
 $\theta_1 \leftarrow 0$
 $\theta_2 \leftarrow 0$
 $\theta_n \leftarrow 0$
Repetir

$$\begin{aligned}
 g_0 &\leftarrow \frac{\partial J(\theta)}{\partial(\theta_0)} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x_0^{(i)} \\
 g_1 &\leftarrow \frac{\partial J(\theta)}{\partial(\theta_1)} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x_1^{(i)} \\
 g_2 &\leftarrow \frac{\partial J(\theta)}{\partial(\theta_2)} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x_2^{(i)} \\
 &\dots \\
 g_n &\leftarrow \frac{\partial J(\theta)}{\partial(\theta_n)} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x_n^{(i)} \\
 \theta_0 &\leftarrow \theta_0 - \alpha g_0 \\
 \theta_1 &\leftarrow \theta_1 - \alpha g_1 \\
 \theta_2 &\leftarrow \theta_2 - \alpha g_2 \\
 &\dots \\
 \theta_n &\leftarrow \theta_n - \alpha g_n
 \end{aligned}$$

Donde:

$h_{\theta}(x) = g(\theta^T x)$
 α es la tasa de aprendizaje.

El anterior algoritmo en forma vectorizada y con unos sencillos ajustes para su seguimiento, queda de la siguiente manera:

$$\begin{aligned}
 \theta &\leftarrow \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \\
 h &= g(X\theta) \\
 J_historia_1 &= \frac{1}{m} \sum_{i=1}^m cost(h_{\theta}(x^{(i)}), y^{(i)}) \\
 &\text{para } i = 1 \text{ hasta } num_iter \\
 &\quad grad \leftarrow \frac{1}{m} ((h - y)^T X)^T \\
 &\quad \theta \leftarrow \theta - \alpha \cdot grad \\
 &\quad h = g(X\theta) \\
 &\quad J_historia_{i+1} = \frac{1}{m} \sum_{i=1}^m cost(h_{\theta}(x^{(i)}), y^{(i)})
 \end{aligned}$$

En el anterior pseudocódigo, g representa la función sigmoide.

La siguiente tarea es implementar el anterior algoritmo de aprendizaje. Para los dos cálculos de h y de $J_{historia_i}$ en el anterior algoritmo, se recomienda utilizar las funciones *prediccionesConjunto()* y *costo()* respectivamente, funciones ya desarrolladas previamente. Las especificaciones de la función son las siguientes:

Nombre:

descensoGradiente()

Parámetros:

X: Matriz de $m \times n + 1$, una columna de unos y, a su derecha, los valores normalizados de las características de todos los ejemplos de entrenamiento

y: Vector columna de m elementos con los valores de la variable a predecir de todos los ejemplos de entrenamiento

alpha: Tasa de aprendizaje α

num_iter: Número de iteraciones que debe realizar el algoritmo de aprendizaje

Retorno:

theta: Vector con los parámetros aprendidos por el algoritmo

J_historia: Un vector columna con el valor de la función de costo antes de ejecutarse la primera iteración, y los valores de esta función al final de cada iteración. Es decir, $J_{historia}(1)$ es el costo $J(\theta)$ antes de iniciarse la ejecución de la primera iteración, $J_{historia}(2)$ es el costo $J(\theta)$ al final de la primera iteración, $J_{historia}(3)$ es el costo $J(\theta)$ al final de la segunda iteración, y así sucesivamente.

Una vez implementada la función, y cuando ejecute el programa *lab4a.m*, este deberá mostrar la salida de la Figura 4

Figura 4. Prueba de la implementación del algoritmo de aprendizaje.

```
DESCENSO POR EL GRADIENTE

Alpha=      0.300000
Numero de iteraciones=  50

... Ejecutando el algoritmo ...

Puede tardar un poco ...
 10%  20%  30%  40%  50%  60%  70%  80%  90% 100%

# de iteracion      Costo (J)
      0      0.6931471806
      1      0.6440300262
      2      0.6023501677
      3      0.5668384852
      4      0.5363957620
      5      0.5101075835
      6      0.4872307897
      7      0.4671684043
      8      0.4494427888
      9      0.4336714399
     10      0.4195468388
     41      0.2559981014
     42      0.2538275469
     43      0.2517295388
     44      0.2497001631
     45      0.2477357930
     46      0.2458330625
     47      0.2439888432
     48      0.2422002230
     49      0.2404644874
     50      0.2387791028

Theta encontrado por descenso por el gradiente:
-1.226519
 1.328246
 1.456077
>>
```

Más adelante se le solicitará ajustar la tasa de aprendizaje α y el número de iteraciones, para obtener un costo que sea el mínimo o prácticamente el mínimo.

3.9 Gráfica del costo vs el número de iteración para varias tasas de aprendizaje

A partir de este momento, las funciones se prueban con el programa *lab4b.m* y no con el

Se acostumbra usar una de las siguientes tasas de aprendizaje:

... 1000 300 100 30 10 3 1 0.3 0.1 0.03 0.01 0.003 0.001 0.0003 0.0001...

Para visualizar el comportamiento de una tasa, se ejecutan 50 iteraciones con la tasa y se grafica en el plano cartesiano el valor de la función de costo versus el número de iteración.

La mejor tasa es aquella cuya gráfica descienda más, sin nunca ascender ni oscilar.

Su siguiente tarea es elaborar una función que realice las gráficas descritas anteriormente para las 3 mejores tasas de aprendizaje. Estas gráficas deben estar superpuestas en el mismo plano cartesiano. Use el método de ensayo y error, para saber cuáles son las 3 mejores tasas.

La función, que debe ubicarse en la carpeta *lab4*, debe tener las siguientes especificaciones:

Nombre:

graficarTasas()

Parámetros:

X: Matriz de $m \times n + 1$, una columna de unos y, a su derecha, los valores normalizados de las características de todos los ejemplos de entrenamiento

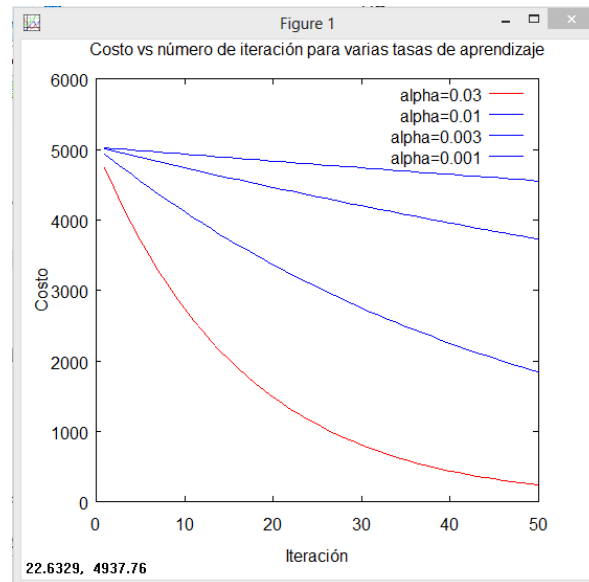
y: Vector columna de m elementos con los valores de la variable a predecir de todos los ejemplos de entrenamiento

Retorno:

Ninguno

Pruebe la función ejecutando el programa *lab4b.m*, debiendo obtener una gráfica del tipo de la de la Figura 5, pero correspondiente al presente laboratorio.

Figura 5. Gráfica del Costo versus el número de iteración para varias tasas de aprendizaje NO correspondiente al presente laboratorio



3.10 Ajuste de la tasa de aprendizaje y del número de iteraciones

En este punto debe ajustar la tasa de aprendizaje y el número de iteraciones, para que el algoritmo de aprendizaje obtenga un costo sea el mínimo o prácticamente el mínimo. Concretamente, debe realizar lo siguiente:

- 1) Cambie en el programa *lab4a.m* el valor que se le asigna a la variable *alpha*, por el valor de la tasa de aprendizaje más prometedora, valor que fue determinado en la sección 3.9.
- 2) Ejecute el programa *lab4a.m*
- 3) En la sección de *Descenso por el gradiente* de la salida del programa *lab4a.m*, observe los valor de la función de costo en las últimas iteraciones, columna *Costo (J)*.
- 4) La meta es minimizar la función de costo. Si no se ha logrado aún esta meta, incremente el número de iteraciones hasta el valor que sea necesario. Para ello cambie el valor que se le asigna a la variable *num_iter* en el programa *lab4a.m*.

3.11 Predicciones para el conjunto de entrenamiento

Nota: A partir de este momento, las funciones se prueban con el programa *lab4b.m*. No es necesario volver a ejecutar el programa *lab4a.m*

Ejecute el programa *lab4b.m*. Debe aparecer el informe correspondiente al último entrenamiento y, seguidamente, las predicciones para los primeros ejemplos de entrenamiento.

Las predicciones, h , deben ser muy similares a las respuestas correctas, representadas por la variable y . Es decir, que la predicción de probabilidad debe ser cercana a 1 o a 0, según se le haya recomendado al estudiante la carrera de Ingeniería de Sistemas y Computación o no.

3.12 Predicciones para nuevos ejemplos

El objetivo del aprendizaje de máquina es crear un modelo para hacer predicciones. Su tarea en este punto es hacer una función sencilla para realizar predicciones para nuevos ejemplos, utilizando los parámetros θ_i aprendidos por el algoritmo. La expresión 'nuevos ejemplos' de refiere a ejemplos que no se encuentren en el conjunto de entrenamiento.

Recuerde que la fórmula para calcular una predicción mediante la regresión logística es $h_{\theta}(x) = g(\theta^T x)$ con $g(z) = \frac{1}{1+e^{-z}}$, previa normalización de las características.

Su tarea en este punto es implementar una función con un interfaz similar a la ilustrada en el prototipo de la Figura 6.

Figura 6. Prototipo de interfaz de la función para realizar predicciones para nuevos ejemplos

PREDICCIONES PARA NUEVOS EJEMPLOS

Continuar(s/n):s

Ingrese el resultado obtenido en la prueba de razonamiento matematico(0.0-5.0):

D.D

Ingrese el puntaje obtenido en prueba de razonamiento verbal(0.0-5.0):

D.D

Prediccion de probabilidad de que le sea recomendable estudiar Ingenieria de Sistemas y Computacion:

D.DDDD

Continuar(s/n):s

Ingrese el resultado obtenido en la prueba de razonamiento matematico(0.0-5.0):

D.D

Ingrese el puntaje obtenido en prueba de razonamiento verbal(0.0-5.0):

D.D

Prediccion de probabilidad de que le sea recomendable estudiar Ingenieria de Sistemas y Computacion:

D.DDDD

Continuar(s/n):n

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.

Las especificaciones de la función son las siguientes.

Nombre:

predicciones ()

Parámetros:

theta: Vector de $n + 1$ elementos con los parámetros aprendidos por el algoritmo

mu: Vector de n elementos con las medias de las características de todos los ejemplos de entrenamiento

sigma: Vector de n elementos con las desviaciones estándar de las características de los ejemplos de entrenamiento

Retorno:

Ninguno

Pruebe su función ejecutando el programa *lab4b.m*. Deberá obtener predicciones razonables de acuerdo con la tendencia de los ejemplos del conjunto de entrenamiento.

3.13 Exactitud del entrenamiento

La exactitud del entrenamiento se encuentra con la siguiente fórmula:

$$\text{Exactitud del entrenamiento} = \frac{\text{Número de aciertos con el conjunto de entrenamiento}}{m} * 100$$

Donde m es el número de ejemplos de entrenamiento.

Su tarea específica en este punto es elaborar en la carpeta *lab4* una función que calcule la exactitud del entrenamiento y que tenga las siguientes especificaciones:

Nombre:

calcularExactitud ()

Parámetros:

y: Vector columna de m elementos con los valores de la variable a predecir de todos los ejemplos de entrenamiento

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

Retorno:

exactitudEntrenamiento: La exactitud del entrenamiento hallada con la fórmula explicada previamente.

Al probar la función, ejecutando el programa *lab4b*, debe obtener una exactitud del entrenamiento superior al 99%, pero inferior al 100%.

3.14 Frontera de decisión

En el presente laboratorio, la frontera de decisión es una línea recta que intenta separar lo mejor posible los signos + de los círculos en la gráfica elaborada en la sección 3.2. La frontera es una recta, debido a que la predicción se realiza con base en exactamente dos características, y sin características polinomiales.

A continuación se explica cómo elaborar la gráfica de la frontera de decisión en el contexto de un problema de predecir si un estudiante va a desertar o no de un programa académico y con un pequeño conjunto de entrenamiento. Ni el problema ni los datos corresponden al presente laboratorio. El conjunto de entrenamiento se presenta en la siguiente tabla:

	x_1	x_2	y
Estudiante	Ingresos familiares (millones de pesos)	Promedio de calificaciones	Desertó
1	1.5	1.3	1
2	1.9	1.4	0
3	0.3	2.0	1
4	0.4	2.6	0

Las medias y las desviaciones estándar son: $\mu = [1.0 \ 1.8]$ y $s = [0.8 \ 0.6]$ y los parámetros aprendidos son $\theta = \begin{bmatrix} 0.7 \\ -16.7 \\ -17.0 \end{bmatrix}$ (2)

Primero hay que determinar la ecuación de frontera de decisión, dada en este caso por:

$$x_2 = -\frac{\theta_1}{\theta_2}x_1 - \frac{\theta_0}{\theta_2} \quad (1)$$

Reemplazando (2) en (1) tenemos

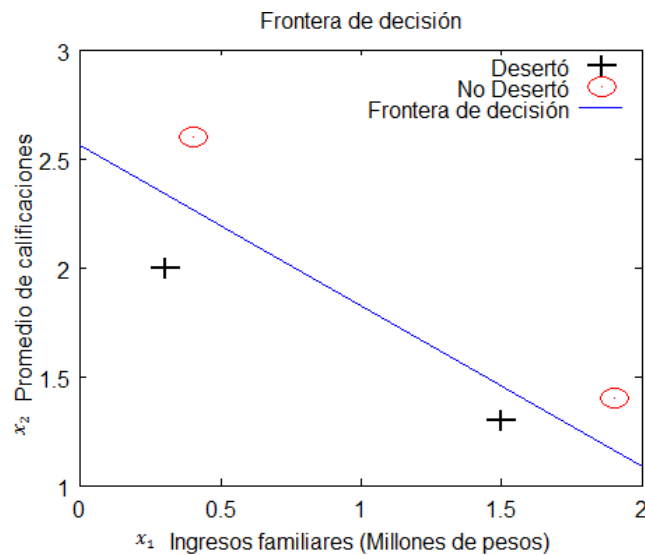
$$x_2 = -\frac{\theta_1}{\theta_2}x_1 - \frac{\theta_0}{\theta_2} = -\frac{(-16.7)}{(-17)}x_1 - \frac{0.7}{-17} = -0.98x_1 + 0.04$$

Como es una recta, basta hallar dos puntos de ella para realizar la gráfica, digamos en $x_1 = 0$ y en $x_1 = 2$. Se han escogido estos dos valores debido a que el rango de la variable *ingresos familiares* es (0,3 a 1.9)

x_1 no normalizado*	0	2
x_1 normalizado $\leftarrow \frac{x_1 \text{ no normalizado} - \mu_1}{s_1}$	-1.25	1.25
x_2 normalizado $\leftarrow -0.98x_1 \text{ normalizado} + 0.04$	1.27	-1.19
x_2 no normalizado $\leftarrow x_2 \text{ normalizado } s_2 + \mu_2$	2.56	1.09

Seguidamente, se grafican los datos y la correspondiente frontera de decisión en un mismo plano cartesiano, véase la Figura 7. La frontera de decisión es una recta que pasa por los puntos hallados (0, 2.56) y (2, 1.09).

Figura 7. Gráfica de la frontera de decisión en el contexto de un problema y datos que NO son los del laboratorio



La tarea en concreto en este punto es implementar en la carpeta *lab4* una función que grafique la frontera de decisión y los datos del conjunto de entrenamiento, superpuestos en el mismo plano cartesiano. Las especificaciones de la función son las siguientes.

Nombre:
`fronteraDecision ()`

Parámetros:

XI: Matriz de $m \times n$ elementos con las características de todos los ejemplos de entrenamiento

y: Vector columna de m elementos con los valores de la variable a predecir de todos los ejemplos de entrenamiento

theta: Vector de $n + 1$ elementos con los parámetros aprendidos por el algoritmo

mu: Vector fila de n elementos con las medias de las características de todos los ejemplos de entrenamiento

sigma: Vector fila de n elementos con las desviaciones estándar de las características de los ejemplos de entrenamiento

Retorno:

Ninguno

Al probar la función, ejecutando el programa *lab4b*, se debe obtener una gráfica similar a la de la Figura 7, pero correspondiente al laboratorio. La gráfica deberá tener por lo tanto signos más, signos menos y una recta intentando separarlos lo mejor posible.

3.15 Estadísticas

La tarea en este punto es elaborar una función que calcule las estadísticas de la prueba de razonamiento verbal que se indican en el prototipo de interfaz que se presenta en la Figura 8. Adicionalmente, la función debe mostrar los resultados en la consola, de acuerdo con este prototipo. Observe que se solicitan dos decimales para los primeros cuatro valores.

Figura 8. Prototipo de interfaz de estadísticas de la prueba de razonamiento verbal

```
ESTADISTICAS DE LA PRUEBA DE RAZONAMIENTO VERBAL

Porcentaje de estudiantes que obtuvieron 3.0 o mas: DDD.DD%
Porcentaje de estudiantes que obtuvieron menos de 3.0: DDD.DD%
Porcentaje total: DDD.DD%
Puntaje promedio: D.DD
Número de estudiantes que obtuvieron 4.5 o mas: DDDD
```

Las especificaciones de la función son las siguientes.

Nombre:

estadisticas()

Parámetros:

x_2 : Vector columna con los puntajes obtenidos por los estudiantes en la prueba de razonamiento verbal.

Retorno:

Ninguno

Pruebe la función *estadisticas()* ejecutando el programa *lab3b.m*.

4 Evaluación y Forma de entrega

En la evaluación se tendrá en cuenta lo siguiente:

- a) Que el código implementado esté correcto
- b) Que la plantilla incluya todo lo en ella solicitado, y que las imágenes y valores presentados estén correctos
- c) Si se solicitan gráficas en el plano cartesiano, deben incluir el título de la gráfica, el nombre de los ejes y, si se requiere, las convenciones

Para la entrega del informe, deben comprimir en un solo archivo la carpeta *lab4* con el código implementado y la plantilla diligenciada. Luego, deben subir este archivo comprimido a la plataforma Classroom, utilizando el correspondiente enlace. Solo es necesario que uno de los integrantes del grupo suba el archivo comprimido, ya que en ella estarán incluidos los nombres de todos los integrantes.

La plantilla se debe encontrar en formato Word para poderle hacer comentarios.

Para informes entregados con posterioridad a la fecha-hora de plazo establecida habrá las siguientes penalizaciones:

- Hasta 24 horas de retraso, 1 unidad
- De 24 a 48 horas de retraso, 2 unidades
- De 48 a 72 horas de retraso, 3 unidades
- De 72 a 96 horas de retraso, 4 unidades
- De 96 horas de retraso en adelante, 5 unidades