Instituto Tecnológico Superior de Cajeme
Ingeniería en Sistemas Computacionales

Inteligencia Artificial

Manual de clase

Elaborado por:
José Luis Beltrán Márquez
lbeltran@itesca.edu.mx

Cd. Obregón, Sonora, México.

Índice general

Índice de figuras

Índice de tablas

Código

Capítulo 1

Introducción a Inteligencia Artificial

1.1. Introducción

”La inteligencia artificial (IA) es el estudio del comportamiento inteligente en artefactos”. [3]

Figura 1.1: Inteligencia Artificial

”El objetivo fundamental de la IA es la invención de una teoría de la inteligencia que sirva para explicar el comportamiento de entidades inteligentes de la naturaleza y para guiar la construcción de entidades artificiales capaces de exhibir el comportamiento inteligente”.[1]

Figura 1.2: Robot

1.2. Machine Learning

El aprendizaje de máquina es una de las tecnologías recientes más emocionantes.
Probablemente haz usado un algoritmo de aprendizaje docenas de veces al día sin conocerlo. Como por ejemplo, cada vez que utilizas una máquina de búsqueda en web (como Google o Bing) para buscar por Internet.

Figura 1.3: Busqueda web

Una de las razones por las que trabajan tan bien estos buscadores, es porque utilizan un algoritmo de aprendizaje, que ha aprendido cómo alinear páginas Web.
Cada vez que utilizas Facebook, o la aplicación foto de Apple, y se reconocen las fotos de tus amigos, también estás usando aprendizaje de máquina.

Figura 1.4: Apple

Cada vez que lees tu correo electrónico y el filtro del Spam te ahorra tener que lidiar con toneladas de correo electrónico basura, eso es también un algoritmo de aprendizaje.

El sueño de los investigadores de I.A. es construir máquinas tan inteligentes como un humano; pero aún hay un largo camino hasta esa meta. Muchos investigadores de I.A. creen que la mejor manera de llegar hacia a esa meta, es con los algoritmos de aprendizaje que intentan imitar cómo el cerebro humano aprende.
El aprendizaje de máquina inicialmente creció fuera del campo de la I.A., y estaba enfocado principalmente en problemas del tipo de computación, como encontrar la ruta más corta en un grafo de A al B, saber si un email es spam o no, etc.
Actualmente el aprendizaje de máquina esta considerado dentro de la I.A., ya que aunque fue desarrollado inicialmente para problemas asociados a las computadoras; en la actualidad toca muchos segmentos de la industria y de la ciencia básica.

1.3. Ejemplos de Aplicaciones

Reconocimiento de Patrones (SNN) Usando características de algún objeto complejo, es posible identificar comportamientos del objeto, que difícilmente se puede identificar a simple vista.

Etiquetado de Fotos (CNN) Dada una imagen, buscar en una base de datos de imágenes y encontrar la imagen más parecida.

Reconocimiento del Habla (RNN) Dado un audio, encontrar el texto que resume el audio.

Conversión de idioma (RNNC) Tomar un texto en un idioma y convertirlo a su equivalente a otro idioma.

Minería de base de datos El crecimiento tan grande de la Web ha ocasionado que tengamos un conjunto de datos más grandes que cualquier tiempo anterior.

Expedientes médicos Con la llegada de la automatización, ahora tenemos informes médicos electrónicos; ya que podemos comenzar a entender las enfermedades mejor, y con información del historial del paciente y con análisis clínicos actuales podemos predecir la existencia de una enfermedad.

Biología de cómputo Con la automatización, los biólogos están recogiendo porciones de datos sobre las secuencias del gen, las secuencias del DNA, y así sucesivamente. Los algoritmos nos están dando una mucho mejor comprensión del genoma humano, y lo qué significa ser humano.

Ingeniería Por ejemplo, el trabajo en artefactos voladores autónomos, o en automóviles autónomos, donde se necesita analizar información de imagen, radar, gps, del mismo automóvil, etc.

Marketing (SNN) Cada vez que entras al Amazon o al Netflix, te recomiendan música, libros, películas o productos relacionados con tus compras anteriores. Si piensas en esto y tomas en cuenta que estas compañias tienen millones de clientes y es prácticamente imposible escribir un programa para cada cliente. La única manera para que el software pueda realizar recomendaciones personalizadas es que el software aprenda las preferencias del cliente. Así pues, las compañias de Silicon Valley recojen el comportamiento de los usuarios de la web (también llamado clickstream) y están intentando utilizar algoritmos de aprendizaje de máquina para minar estos datos, para entender a los usuarios mejor y para servir a los usuarios mejor.

1.4. Datos de Entrada

Datos estructurados Tablas tipo excel

Datos sin estructurar Audio, imágenes, texto, etc

Capítulo 2

Revisión de Álgebra Lineal

2.1. Matrices

Una matriz es una arreglo rectangular de números dispuestos en renglones y columnas [2], por ejemplo: $M = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{pmatrix}$
En este ejemplo es una matriz de nombre M que tiene 4 renglones y 3 columnas; donde el elemento $m_{11} = a$; y el elemento $m_{43} = l$
La dimensión de la matriz se denota de la siguiente manera $M \in \Re^{4*3}$
Es importante notar que existen 2 maneras de indexar los elementos de una matriz:

- Indexado empezando con cero.- Es el más común en los lenguajes de programación.
- Indexado empezando con uno.- Para ser compatible con octave o Matlab.

Se usa nombrar con mayúsculas a las matrices; pero cuando se hace referencia a un elemento de la matriz, se escribe el nombre de la matrix en minúscula y 2 subíndices: primero el correspondiente al renglón y después el de columna. Por ejemplo, el elemento f de la matrix M viene siendo $m_{2,3}$.

2.1.1. Matriz Identidad

la matriz identidad es una matriz que cumple la propiedad de ser el elemento neutro del producto de matrices. Esto quiere decir que el producto de cualquier matriz por la matriz identidad no tiene ningún efecto.
Como el producto de matrices sólo tiene sentido si sus dimensiones son compatibles, existen infinitas matrices identidad dependiendo de las dimensiones.
I_n es la matriz identidad de tamaño n, y se define como la matriz diagonal que tiene valor 1 en cada una de las entradas de la diagonal principal, y 0 en el resto. Así por ejemplo:

$$I_1 = \begin{pmatrix} 1 \end{pmatrix}, I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, ..., I_n = \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1 \end{pmatrix}$$

2.1.2. Matriz Inversa

La inversa de la matriz A, es otra matriz, del mismo orden que A, tal que al efectuar el producto, se obtiene como resultado la matriz identidad.
Sólo las matrices cuadradas pueden tener inversa, sin embargo, el hecho de que una matriz sea cuadrada no garantiza la existencia de una matriz inversa.

si A es una matriz m x m, y y si ésta tiene inversa:

$A \cdot A^{-1} = A^{-1} \cdot A = I$

Ejemplo: $\begin{pmatrix} 3 & 4 \\ 2 & 16 \end{pmatrix}\begin{pmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2$

2.1.3. Matriz Regular

Es aquella que si tiene matriz inversa, ya que su determinante es diferente de cero.

2.1.4. Matriz Singular

Es aquella matriz que no tiene inversa porque su determinante es igual a cero.

2.1.5. Determinante de una Matriz

El determinante de una matriz A es un escalar único asociado con los elementos de una matriz, por medio de operaciones definidas.
El determinante se simboliza por: det(A) ó bien, envolviendo la matriz con barras verticales $|A|$
El valor de un determinante sólo se obtiene en matrices cuadradas.
Existen varios métodos para encontrar determinantes, para matrices de 2^o y 3^o orden se usa el método de diagonales que consiste en: $|A| = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11}a_{22} - a_{21}a_{12} = det(A)$
Para matrices mayores se puede aplicar el método de cofactores, siempre y cuando sea una matriz cuadrada.

2.1.6. Matriz Cofactor

La matriz cofactor se representa por A^C y esta formada por los cofactores de cada elemento de la matriz original.
Sólo las matrices cuadradas tienen matriz cofactor, la fórmula para obtener cofactores es: $\alpha_{ij} = (-1)^{i+j}M_{i,j}$
$M_{i,j}$.- Es un determinante de menor orden que la original y se obtiene al eliminar la fila y la columna del elemento con que se esta trabajando.
Ejemplo:

$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}, A^C = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} \end{pmatrix}$

donde por ejemplo: $\alpha_{3,3} = (-1)^{3+3}\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = (1) * (a_{11}a_{22} - a_{21}a_{12})$

2.1.7. Matriz Transpuesta

Sea A una matriz. La matriz transpuesta, denotada con A^t está dada por aquella matriz donde el elemento a_{ji} de la matriz original A se convertirá en el elemento a_{ij} de la matriz transpuesta A^t.

2.1.8. Matriz Adjunta

La matriz adjunta es la transpuesta de la matriz cofactor y se representa por ADJ A. Su fórmula es: $ADJ\ A = (A^C)^T$

2.2. Vectores

Los vectores son un caso especial de matrices donde una de sus dimensiones es 1; de esta forma tenemos vector renglón, y vector columna.
Ejemplo de vector renglón: $X = \begin{pmatrix} m & n & o & p & q \end{pmatrix}$
Ejemplo de vector columna: $Y = \begin{pmatrix} r \\ s \\ t \\ u \end{pmatrix}$
La dimensión de un vector se denota de la siguiente manera $X \in \mathfrak{R}^5$; $Y \in \mathfrak{R}^4$
Cuando se hace referencia a un elemento del vector, se escribe el nombre del vector y 1 subíndice. Por ejemplo, el elemento p del vector X viene siendo x_4; o el elemento s del vector Y viene siendo y_2

2.3. Operaciones

2.3.1. Suma y Resta de matrices

La suma y resta de matrices solo se puede realizar cuando las 2 matrices tienen la misma dimensión, y la suma o resta se realiza con los elementos correspondientes, es decir, los elementos que tienen los mismos subíndices, ejemplo: $A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$, $B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$
$A + B = \begin{pmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} \end{pmatrix}$

2.3.2. Multiplicación de matrices

La multiplicación de matrices solo se puede realizar cuando la segunda dimensión de la primer matriz es igual a la primera dimensión de la segunda matriz, y la matriz resultante tiene como dimensiones la primera dimensión de la primer matriz y la segunda dimensión de la segunda matriz; en otras palabras, suponiendo $A \in \mathfrak{R}^{M*N}$ y $B \in \mathfrak{R}^{N*P}$; $(A * B) \in \mathfrak{R}^{M*P}$
De esta forma, el elemento $(A * B)_{ij}$ se forma al multiplicar el renglón i de la matriz A, por la columna j de la matriz B.
Ejemplo:
$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$, $B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$

$$(A * B) = \begin{pmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} & a_{1,1} \cdot b_{1,2} + a_{1,2} \cdot b_{2,2} \\ a_{2,1} \cdot b_{1,1} + a_{2,2} \cdot b_{2,1} & a_{2,1} \cdot b_{1,2} + a_{2,2} \cdot b_{2,2} \end{pmatrix}$$

2.3.3. División de matrices

La división de matrices no existe, se usa la multiplicación por la matriz inversa que es lo más parecido a dividir en el mundo de las matrices.

2.3.4. Multiplicación de una matriz por un escalar

Cada elemento de la matriz se multiplica de manera individual por el escalar, ejemplo:

$$A = \begin{pmatrix} 2 & 2 \\ 4 & 6 \end{pmatrix}$$

$$3 * A = \begin{pmatrix} 3 * 2 & 3 * 2 \\ 3 * 4 & 3 * 6 \end{pmatrix}$$

$$3 * A = \begin{pmatrix} 6 & 6 \\ 12 & 18 \end{pmatrix}$$

2.3.5. División de una matriz por un escalar

Cada elemento de la matriz se multiplica de manera individual por el reciproco del escalar, ejemplo:

$$A = \begin{pmatrix} 2 & 2 \\ 4 & 6 \end{pmatrix}$$

$$\frac{A}{3} = \frac{1}{3} * A = \begin{pmatrix} \frac{1}{3} * 2 & \frac{1}{3} * 2 \\ \frac{1}{3} * 4 & \frac{1}{3} * 6 \end{pmatrix}$$

$$\frac{1}{3}A = \begin{pmatrix} \frac{2}{3} & \frac{2}{3} \\ \frac{4}{3} & 2 \end{pmatrix}$$

2.3.6. Ecuación matricial

Una ecuación matricial es una ecuación que podemos expresar por medio de matrices, o bien una ecuación en la que intervienen matrices.

Ejemplo 1

Resuelva (obtenga X) la siguiente ecuación matricial $A + B–X = C$, donde A, B, C y X son matrices.

$$A = \begin{pmatrix} 5 & 4 \\ 3 & 8 \end{pmatrix} B = \begin{pmatrix} 6 & 5 \\ 2 & 3 \end{pmatrix} C = \begin{pmatrix} 5 & 6 \\ 3 & 5 \end{pmatrix}$$

$$A + B - X = C$$

$$\begin{pmatrix} 5 & 4 \\ 3 & 8 \end{pmatrix} + \begin{pmatrix} 6 & 5 \\ 2 & 3 \end{pmatrix} - X = \begin{pmatrix} 5 & 6 \\ 3 & 5 \end{pmatrix}$$

Despejando..: $X = A + B - C$

$$X = \begin{pmatrix} 5 & 4 \\ 3 & 8 \end{pmatrix} + \begin{pmatrix} 6 & 5 \\ 2 & 3 \end{pmatrix} - \begin{pmatrix} 5 & 6 \\ 3 & 5 \end{pmatrix}$$

$$X = \begin{pmatrix} 6 & 3 \\ 2 & 6 \end{pmatrix}$$

Ejemplo 2

resuelva X de la ecuación matricial $AX + B = 3X$, donde:

$A = \begin{pmatrix} 5 & 4 \\ 3 & 8 \end{pmatrix} B = \begin{pmatrix} 4 & 0 & -2 \\ 1 & -3 & -1 \end{pmatrix}$

Para que AX se pueda sumar a B, la multiplicación de AX debe arrojar una matriz de $2 * 3$. Esto se obtiene si X tiene 2 filas y 3 columnas.

$AX + B = 3X$

$\begin{pmatrix} 5 & 4 \\ 3 & 8 \end{pmatrix} X + \begin{pmatrix} 4 & 0 & -2 \\ 1 & -3 & -1 \end{pmatrix} = 3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} X$

Despejando..: $(A - 3)X = -B$

$\left[\begin{pmatrix} 5 & 4 \\ 3 & 8 \end{pmatrix} - 3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right] X = - \begin{pmatrix} 4 & 0 & -2 \\ 1 & -3 & -1 \end{pmatrix}$

$\left[\begin{pmatrix} 5 & 4 \\ 3 & 8 \end{pmatrix} - \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \right] X = \begin{pmatrix} -4 & 0 & 2 \\ -1 & 3 & 1 \end{pmatrix}$

$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix} X = \begin{pmatrix} -4 & 0 & 2 \\ -1 & 3 & 1 \end{pmatrix}$

Despejando..: $X = \begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}^{-1} \begin{pmatrix} -4 & 0 & 2 \\ -1 & 3 & 1 \end{pmatrix}$

$X = \begin{pmatrix} -2.5 & 2 \\ 1.5 & -1 \end{pmatrix} \begin{pmatrix} -4 & 0 & 2 \\ -1 & 3 & 1 \end{pmatrix}$

$X = \begin{pmatrix} 8 & 6 & -3 \\ -5 & -3 & 2 \end{pmatrix}$

2.3.7. Problemas

Problema 1

$A = \begin{pmatrix} 2 & 2 \\ 4 & 6 \end{pmatrix}$

Calcular la inversa de A.

Problema 2

suponemos que $A = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$

y que $B = \begin{pmatrix} 0 & 0 \\ 2 & 0 \end{pmatrix}$

Calcula AxB, y BxA

Problema 3

suponemos que $A \cdot C = B$, y que $A = \begin{pmatrix} 1 & 3 \\ 2 & 5 \end{pmatrix}$

y que $B = \begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}$

Calcula C

2.4. Ejemplos

```python
import numpy as np

#A = np.array([[1, 1 , 0], [1, 0, 1], [0, 1, 0]])
#A = np.array([[1, -1 , 0], [0, 1, 0], [2, 0, 1]])
# = np.array([[1, 2 , 0], [2, 3, 0], [0, 0, 1]])
#A = np.array([[2, 1, 0], [1, 1, 0], [0, 0, 1]])
#A = np.array([[3, 2, -3], [7, -1, 0], [2, -4, 5]])
#A = np.array([[1, 4], [0, 1]])
#A = np.array([[1, 0], [2, 1]])
#A = np.array([[2, 0], [-1, 3]])
A = np.array([[5, -2], [4, 1], [6,7]])

#matriz no invertible
#A = np.array([[3, 2, -3], [3, 2, -3], [2, -4, 5]])

print(f"A= \n{A}")

try:
    invA = np.linalg.inv(A)
except Exception as e:
    print("La matriz no tiene inversa")
else:
    print(f"inversa de A= \n{invA}")
    I = A.dot(invA)
    print(f"I= \n{I}")

renglones, columnas = A.shape
print(f"tiene {renglones} renglones y {columnas} columnas")
if renglones != columnas:
    print("La matriz tiene que ser cuadrada para calcular el determinante")
else:
    detA = np.linalg.det(A)
    print(f"El determinante es: {detA}")

transpuesta_a = np.transpose(A)
print(f"La transpuesta es:\n{transpuesta_a}")
```

code/algebra_lineal.py

```python
import numpy as np
import matplotlib.pyplot as plt

#calorias, proteinas, grasa en 100g
manzana = np.array([56, 1.2, 1.8])
carne= np.array([0, 104, 135])
huevo= np.array([4.4, 52, 99])
papa= np.array([68, 8, 0.9])
```

```
#crear la matriz
A= np.zeros([3, 4])
A[:,0]= manzana
A[:,1]= carne
A[:,2]= huevo
A[:,3]= papa
print(A)

#eje 0 son la columnas, y eje 1 son los renglones
promedios = A.mean(axis = 0)
print(f"promedios= \n{promedios}")
mayor = A.max(axis=1)
print(f"mayor= \n{mayor}")

#
    ----------------------------------------------------------------------------------------

A= np.array([[1, 2 ,3],[4, 5, 6]])


#suma de  un arreglo con un escalar
#A = A+ 100

#suma de  un arreglo con un vector renglon
#A = A+ np.array([100, 200, 300])

#suma de  un arreglo con un vector columna
A = A+ np.array([[100], [200]])
print(f"A=\n{A}")

#poner la suma de vectores
A = np.array([[100], [200]])
B = np.array([100, 200])
C = A + B
print(f"C= \n{C}")

#poner la multiplicacion de vectores
A = np.array([[100], [200]])
#B = np.array([100, 200])
A = A * 2
print(f"A*2= \n{A}")

#
    ----------------------------------------------------------------------------------------

A = np.random.randn(5)
print(f"A=\n{A}")
print(f"shape de A= {A.shape}")
print(f"dimensiones de A= {A.ndim}")
print(f"A transpuesta= \n{A.T}")
#en lugar de lo anterior
A = np.random.randn(1,5)
A = A.reshape((5,1))
print(f"A=\n{A}")
print(f"dimensiones de A= {A.ndim}")
```

```
print(f"shape de A= {A.shape}")
print(f"tipo de A= {type(A[0,0])}")
assert( A.shape == (5,1))
print(f"A transpuesta= \n{A.T}")
#NOTA: puedes asegurate que es un vector columna, usando A = A.reshape([5,1])


#-------------------------------------------------------------------------
x = np.array([0, np.pi/2, np.pi])
y = np.sin(x)

# np.linspace (extremo1, extremo2, num =  numero de intervalos)
x = np.linspace(-2,2, 5)
x = np.linspace(0,2*np.pi, 100)
#x = np.arange(0,2*np.pi+np.pi/4, np.pi/4)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

code/fundamentos_numpy.py

```
import numpy as np

#x = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]], np.float64)
#x = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]], np.int64)
#x = np.loadtxt("arreglo_entero.csv")
#x = np.loadtxt("arreglo_real.csv")
x = np.loadtxt("arreglo_imaginario.csv").view(complex)

print()
print(f"x= {x}")
print(f"Tipo de x: {type(x)}")
print(f"flags de x: {x.flags}")
print(f"shape de x: {x.shape}")
print(f"strides de x: {x.strides}")
print(f"ndim de x: {x.ndim}")
print(f"data de x: {x.data}")
print(f"size de x: {x.size}")
print(f"itemsize de x: {x.itemsize}")
print(f"nbytes de x: {x.nbytes}")
print(f"base de x: {x.base}")
print(f"T de x: {x.T}")
print(f"real de x: {x.real}")
print(f"imag de x: {x.imag}")
print(f"flat de x: {x.flat}")
print(f"ctypes de x: {x.ctypes}")
```

code/matrices.py

```
import matplotlib.pyplot as plt
import numpy as np

A = np.loadtxt("datos/inCanion.csv")
print(A)
angulo = A[0]
```

```
velocidad = A[1]
print(f"angulo grados= {angulo}")
angulo = angulo * np.pi / 180
altura = (velocidad * velocidad * (np.sin(angulo) * np.sin(angulo)))  / ( 2*9.8)
alcance = (velocidad * velocidad * (np.sin(2 * angulo)))  / ( 9.8)
tiempo = (2 * velocidad  * (np.sin(angulo)))  / ( 9.8)

print(f"Altura= {altura}")
print(f"alcance= {alcance}")
print(f"tiempo= {tiempo}")
print(f"angulo= {angulo}")
print(f"velocidad= {velocidad}")


tiempo = np.linspace(0, tiempo, 100)
px = velocidad * np.cos(angulo) * tiempo
py = velocidad *np.sin(angulo) * tiempo - .5 * 9.8 * tiempo * tiempo
plt.plot(px,py)
plt.grid()
plt.title("Tiro parabolico")
plt.xlabel("Alcance m")
plt.ylabel("Altura m")
plt.savefig("tiro_parabolico.png")
plt.show()

salida = np.array([[altura, tiempo[-1], alcance]])
print(f"Salida:\n{salida}")
np.savetxt("datos/outCanion.csv",salida, fmt="%.2f")
# alcance = A[1,0]
# velocidad = A[1,1]
# angulo = np.arcsin( (9.8 * alcance) / ( 2 * velocidad * velocidad ) )
# angulo = (angulo * 180) / (np.pi)
# print(f"angulo= {angulo}")
```

code/graficos.py

Capítulo 3

Tareas

3.1. numpy

Suponer un cañón situado en el origen de un sistema de coordenadas cartesiano.

Se tiene 1 archivo de entrada inCanion.csv, el cual tiene la siguiente información:

ánguloInicial velocidadInicial

Se deberán realizar los cálculos necesarios y escribir los resultados en el archivo outCanion.csv, con el siguiente formato:

alturaMáxima tiempoVuelo alcanceHorizontal

NOTA: Recuerda que (π *radianes* = 180), ($g \approx 9.8$)

Fórmulas necesarias:

$$altura = \frac{Vo^2 \cdot sen^2(\measuredangle)}{2 \cdot g}$$

$$alcance = \frac{Vo^2 \cdot sen(2 \cdot \measuredangle)}{g}$$

$$tiempo = \frac{Vo \cdot sen(\measuredangle)}{g}$$

3.2. matplotlib

Del ejemplo anterior, deberán dibujar una gráfica, donde se muestre la trayectoria del proyectil, recordando que el proyectil no penetra el suelo, que la altura inicial del cañón es 0 (H=0)
Fórmulas necesarias:

$$posicionX = V_x \cdot t = Vo \cdot cos(\measuredangle) \cdot t$$

$$posicionY = H + Vo_y \cdot t - \frac{1}{2} \cdot g \cdot t^2 = H + Vo \cdot sen(\measuredangle) \cdot t - \frac{1}{2} \cdot g \cdot t^2$$

3.3. Ejemplos

```
37 20
```

code/datos/inCanion.csv

```
7.39 2.46 39.24
```

code/datos/outCanion.csv

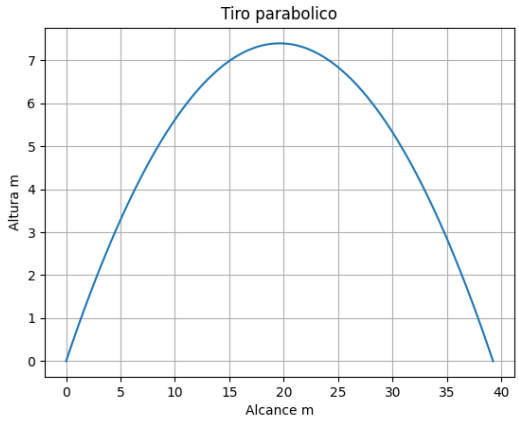


Figura 3.1: tiro parabolico

Capítulo 4

Graficas

4.1. matplotlib

Existen diversos módulos para hacer gráficos en Python, pero el estándar es matplotlib. Se trata de un paquete que contiene principalmente dos módulos básicos: pyplot y pylab.
pyplot ofrece una interfaz fácil para crear gráficos, automatizando la creación de figuras y ejes automáticamente.
Por otra parte, **pylab** combina la funcionalida de pyplot para hacer gráficos con la funcionalidad de numpy para hacer cálculos con arrays.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-3.0, 3.1, 0.5)
y = x*x
print(x)
print(y)
fig = plt.figure()
#gs = fig.add_gridspec(1,2)
#ax = fig.add_subplot(gs[0,0])
ax = fig.add_subplot(121) # Figura con una fila y dos columnas, activo primer subgráfico
ax.plot(x,y)
ax.set_xlabel('X1')
ax.set_ylabel('Y1')
ax.set_title("Grafica 1")

#bx = fig.add_subplot(gs[0,1])
bx = fig.add_subplot(122) # Figura con una fila y dos columnas, activo segundo subgrá
    fico
bx.plot(x,y+2)
bx.set_xlabel('X2 ')
bx.set_ylabel('Y2 ')
bx.set_title("Grafica 2")
plt.show()


x = np.arange(-3,3.1,.5)
y1 = x
y2 = x*x
y3 = x*x*x
plt.plot(x,y1,'-b.', x, y2,'-rd', x, y3,'-g^')
```

```
plt.title("grafica 3")
plt.xlabel("eje X")
plt.ylabel("eje Y")
plt.legend(("$X$", "$X^2$", "$X^3$"), prop = {'size':20}, loc = 'lower right') #upper
plt.show()

plt.plot(x, y2, lw=5, c='y', marker='o', ms=20, mfc='red', label="$X^2$")
plt.legend(loc='lower right')
plt.grid()
#plt.text(-1,3,"$x^2$", fontsize=16)
plt.annotate("$x^2$",
             xytext=(-1,3),
             xy=(0, 0),
             xycoords='data',
             fontsize=20,
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))
plt.show()
```

code/matplotlib_1.py

Si queremos que se cree un nuevo gráfico cada vez que hacemos plot(), debemos añadir el parámetro hold=False a plot():

```
mi_dibujo, = plot(x*2,'o', hold=False)
```

Listing 4.2: nueva gráfica

Para borrar toda la figura se puede usar la función clf(), mientras que cla() sólo borra lo que hay dibujado dentro de los ejes y no los ejes en si.

Se pueden cambiar otras propiedades de la gráfica como parámetros de plot():

Símbolo	Color
b	Azul
g	Verde
r	Rojo
c	Cian
m	Magenta
y	Amarillo
k	Negro
w	Blanco

Símbolo	Descripción
-	Línea continua
–	Línea a trazos
-.	Línea a puntos y rayas
:	Línea punteada
.	Símbolo punto
,	Símbolo Pixel
o	Símbolo círculo relleno
v	Símbolo triángulo hacia abajo
^	Símbolo triángulo hacia arriba
<	Símbolo triángulo hacia la izquierda
>	Símbolo triángulo hacia la derecha
s	Símbolo cuadrado
p	Símbolo pentágono
*	Símbolo estrella
+	Símbolo cruz
x	Símbolo X
D	Símbolo diamante
d	Símbolo diamante delgado

Parámetro	Significado y valores
alpha	grado de transparencia, float (0.0=transparente a 1.0=opaco)
color o c	Color de matplotlib
label	Etiqueta con cadena de texto, string
markeredgecolor o mec	Color del borde del símbolo
markeredgewidth o mew	Ancho del borde del símbolo, float (en número de puntos)
markerfacecolor o mfc	Color del símbolo
markersize o ms	Tamaño del símbolo, float (en número de puntos)
linestyle o ls	Tipo de línea, '-' '–' '-.' ':' 'None'
linewidth o lw	Ancho de la línea, float (en número de puntos)
marker	Tipo de símbolo, '+' '*' ',' '.' '1' '2' '3' '4' '<' '>' 'D' 'H' '^' '_' 'd' 'h' 'o' 'p' 's' 'v' 'x' '|' TICKUP TICKDOWN TICKLEFT TICKRIGHT

Capítulo 5

K medias

5.1. Clasificación

Uno de los objetivos principales del aprendizaje de máquina (Machine Learning) es hacer prediciones precisas, a partir de observaciones con datos previos.
K-medias es un método de agrupamiento, que tiene como objetivo la partición de un conjunto de n observaciones en k grupos en el que cada observación pertenece al grupo cuyo valor medio es más cercano. Es un método utilizado en minería de datos.
Las principales ventajas del método k-means son que es un método sencillo y rápido. Pero es necesario decidir el valor de k y el resultado final depende de la inicialización de los centroides. En principio no converge al mínimo global sino a un mínimo local.
Podemos considerar tres tipos de algoritmos:

1. **Clasificación supervisada:** disponemos de un conjunto de datos, los cuales están etiquetados que pertenecen a una clase específica; y disponemos también de un algoritmo que aprende la forma de los datos de cada una de las clases. Una vez construído el modelo podemos utilizar el algoritmo para clasificar nuevos datos que, en esta fase, ya no necesitan etiqueta para su clasificación, aunque sí la necesitan para evaluar el porcentaje de objetos bien clasificados.

2. **Clasificación no supervisada:** los datos no tienen etiquetas (o no queremos utilizarlas) y estos se clasifican a partir de su estructura interna (propiedades, características).

3. **Clasificación semisupervisada:** algunos datos de entrenamiento tienen etiquetas, pero no todos. Este último caso es muy típico en clasificación de imágenes, donde es habitual disponer de muchas imágenes mayormente no etiquetadas. Estos se pueden considerar algoritmos supervisados que no necesitan todas las etiquetas de los datos de entrenamiento.

5.2. El algoritmo k-medias (k-means)

K-means es un algoritmo de clasificación no supervisada (clusterización) que agrupa objetos en k grupos basándose en sus características.
El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o cluster. Se suele usar la distancia euclidana.
El algoritmo consta de tres pasos:

1. **Inicialización:** una vez escogido el número de grupos, k, se establecen k centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente, o distribuyendo los centroides uniformemente en el espacio de los datos.

2. **Asignación objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.

3. **Actualización centroides:** se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

4. Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso.

El algoritmo k-means resuelve un problema de optimización, siendo la función a optimizar (minimizar) la suma de las distancias euclidianas de cada objeto al centroide de su cluster.

```
import numpy as np
import matplotlib.pyplot as plt

def getDistancia(A, B):
    C = (A - B) ** 2
    dimension = C.ndim
    if dimension > 1:
        C = np.sum(C, axis=1)
    else:
        C = np.sum(C)
    C = np.sqrt(C)
    if dimension > 1:
        C = np.sum(C)
    return C

A = np.genfromtxt("datos/kmedias_1.csv")
renglones, columnas = A.shape


#print(f"renglones= {renglones}")
#print(f"columnas= {columnas}")
indiceMenor = 0
distanciaMenor = float("inf")
cadena = "                    "
for i in range(renglones):
    cadena += f"({A[i,0]:{4}.{2}},{A[i,1]:{4}.{2}})   "
cadena += "  TOTAL\n"
for i in range(renglones):
    cadena += f"({A[i, 0]:{4}.{2}},{A[i, 1]:{4}.{2}})   "
    for j in range(renglones):
        cadena += f"{getDistancia(A[i,:], A[j,:]):{11}.{2}}   "
    distancia = getDistancia(A[i,:], A)
    if distancia < distanciaMenor:
        distanciaMenor = distancia
        indiceMenor = i
    cadena += f"{distancia:{11}.{4}}\n"
print(cadena)

medias = np.mean(A,axis=0)
print(medias)
plt.plot(A[:,0],A[:,1],"g*")
```

```
plt.plot(A[indiceMenor,0], A[indiceMenor,1], "bo")
plt.plot(medias[0], medias[1], "rD")
plt.show()
```

code/kmedias_1.py

```
import numpy as np
import matplotlib.pyplot as plt

def inicializacion(A, K):
    xmin = np.min(A[:,0])
    ymin = np.min(A[:,1])
    xmax = np.max(A[:,0])
    ymax = np.max(A[:, 1])
    ymedio = ymin+(ymax-ymin)/2
    PA = np.array([xmin, ymedio])
    PB = np.array([xmax, ymedio])
    centroides = np.zeros([K, 2])
    if K>1:
        distancia = getDistancia(PA, PB )/(K-1)
        centroides[0, 0] = xmin
        centroides[0, 1] = ymedio
    else:
        centroides[0, 0] = xmin+(xmax-xmin)/2
        centroides[0, 1] = ymedio
    #xmin = xmin - distancia/2

    for i in range(1,K):
        centroides[i,0] = xmin + distancia * (i)
        centroides[i,1] = ymedio
    return centroides

def asignacion(A, centroides):
    renA, colA = A.shape
    renC, colC = centroides.shape
    distancias = np.zeros([renA, renC])
    for i in range(renA):
        for j in range(renC):
            distancias[i,j]= getDistancia(A[i,:],centroides[j,:])
    indices = np.argmin(distancias, axis=1)
    return indices

def actualizacion(A, indices, K):
    datos = list(range(K))
    for i in range(K):
        datos[i] = []
    for i in range(A.shape[0]):
        for j in range(K):
            if j == indices[i]:
                datos[j].append(A[i,:])

    a = 0
    for i in range(K):
        if len(datos[i]) == 0 :
            a = a + 1
    centroides = np.zeros([K-a, 2])
```

```
    a = 0
    for i in range( K ):
        if len(datos[i]) == 0 :
            a = a + 1
            continue
        B = np.array(datos[i])
        promedios = np.mean(B, axis=0)
        centroides[i - a,0] = promedios[0]
        centroides[i - a,1] = promedios[1]
    return centroides


def getDistancia(A, B):
    C = (A - B) ** 2
    dimension = C.ndim
    if dimension > 1:
        C = np.sum(C, axis=1)
    else:
        C = np.sum(C)
    C = np.sqrt(C)
    if dimension > 1:
        C = np.sum(C)
    return C

A = np.genfromtxt("datos/kmedias_2.csv")
K = 3

centroides = inicializacion(A, K)
buscarCentroides = True
# plt.plot(A[:, 0], A[:, 1], "g*")
# plt.plot(centroides[:, 0], centroides[:, 1], "ro")
# plt.grid()
# plt.show()
while(buscarCentroides):
    indices = asignacion(A, centroides)
    centroides2 = actualizacion(A,indices,K)
    if len(centroides) == len(centroides2) :
        diferencia = np.sum(centroides-centroides2)
        if diferencia**2 < 0.1:
            buscarCentroides = False
    centroides = centroides2
    # plt.plot(A[:, 0], A[:, 1], "g*")
    # plt.plot(centroides[:, 0], centroides[:, 1], "ro")
    # plt.grid()
    # plt.show()

colores = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w']
simbolos = ['^', '+', '*', 'v', 's', 'o', 'x', 'D']

distancias = np.zeros([centroides.shape[0]])
for i in range(A.shape[0]):
    for j in range(centroides.shape[0]):
        distancias[j] = getDistancia(A[i,:],centroides[j,:])
    k = np.argmin(distancias)
    plt.plot(A[i,0], A[i,1], colores[k]+simbolos[k])
plt.show()
```

code/kmedias_2.py

5.3. Cráneos del Tibet

Los datos consisten en cinco medidas tomadas en cráneos recogidos en dos zonas del Tibet.
Los datos de craneos1.csv (n= 17) corresponden a tumbas de Sikkim y alrededores.
Los datos de craneos2.csv (n= 15) corresponden a cráneos encontrados en el campo de batalla de Lhasa.
Se cree posible que los dos conjuntos pertenezcan a dos etnias diferentes. Las cinco variables son:

1. Mayor longitud del cráneo
2. Mayor anchura horizontal del cráneo
3. Altura del cráneo
4. Altura de la parte superior de la cara
5. Anchura de la cara entre los huesos de las mejillas

```
190.5 152.5 145 73.5 136.5
172.5 132 125.5 63 121
167 130 125.5 69.5 119.5
169.5 150.5 133.5 64.5 128
175 138.5 126 77.5 135.5
177.5 142.5 142.5 71.5 131
179.5 142.5 127.5 70.5 134.5
179.5 138 133.5 73.5 132.5
173.5 135.5 130.5 70 133.5
162.5 139 131 62 126
178.5 135 136 71 124
171.5 148.5 132.5 65 146.5
180.5 139 132 74.5 134.5
183 149 121.5 76.5 142
169.5 130 131 68 119
172 140 136 70.5 133.5
170 126.5 134.5 66 118.5
```

code/datos/craneos1.csv

```
182.5 136 138.5 76 134
179.5 135 128.5 74 132
191 140.5 140.5 72.5 131.5
184.5 141.5 134.5 76.5 141.5
181 142 132.5 79 136.5
173.5 136.5 126 71.5 136.5
188.5 130 143 79.5 136
175 153 130 76.5 142
196 142.5 123.5 76 134
200 139.5 143.5 82.5 146
```

```
185 134.5 140 81.5 137
174.5 143.5 132.5 74 136.5
195.5 144 138.5 78.5 144
197 131.5 135 80.5 139
182.5 131 135 68.5 136
```

code/datos/craneos2.csv

Capítulo 6

Escalamiento

6.1. Escalamiento de las características

Las variables independientes (características) pueden tener valores muy disparejos, donde algunas características pueden andar en un rango de miles, mientras que otras variables en un rango incluso de fracciones de unidad.
Lo anterior afecta al desempeño de los algoritmos, ya que un pequeño incremento a las características de valor pequeño, afectan bastante al rsultado, mientras que cambios incluso grandes a las características de valor grande, no influyen mucho en el valor final de la conclusión.
Una manera de solucionar lo anterior es realizar una escalamiento de las variables para asegurarnos que queden dentro de un rango pequeño y similar para todas las variables.
Los rangos típicos son:

- $0 \leftarrow\text{----}\rightarrow 1$
- $-1 \leftarrow\text{----}\rightarrow 1$
- $0 \leftarrow\text{----}\rightarrow .5$
- $-0.5 \leftarrow\text{----}\rightarrow 0.5$
- $0 \leftarrow\text{----}\rightarrow 5$
- $\cdots$

Variables Independientes			Salida
X_1	X_2	X_3	Y
3	4	195	460
2	2	131	232
2	3	142	315
1	1	80	178

Una técnica muy usada para el escalamiento, es usar el promedio y el rango. Suponiendo la tabla anterior.
Primero se obtienen los siguientes valores para cada columna:
$min(X_j)$; es el valor mínimo de la columna j.
$max(X_j)$; es el valor máximo de la columna j.
$Rango_j$; es el valor máximo de la columna j menos el valor mínimo de la columna j.

μ_j; es el valor promedio de la columna j.
De forma que el nuevo valor se obtiene con la siguiente fórmula:

$$x_j^i = \frac{x_j^i - \mu_j}{Rango_j}$$

La tabla quedaría entonces:

Variables Independientes			Salida
X_1	X_2	X_3	Y
0.5	0.5	0.504	460
0	-0.167	-0.052	232
0	0.167	0.043	315
-0.5	-0.5	-0.496	178

Capítulo 7

Ecuación Normal

7.1. Sistema de Ecuaciones

Suponemos el siguiente sistema de ecuaciones:

$x + y + z = 12$
$2x - y + z = 7$
$x + 2y - z = 6$

¿Cuánto vale x,y,z? Podemos reescribir el problema de la siguiente manera:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & -1 & 1 \\ 1 & 2 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 12 \\ 7 \\ 6 \end{pmatrix}$$

Si decimos que $A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & -1 & 1 \\ 1 & 2 & -1 \end{pmatrix}$

, que $Y = \begin{pmatrix} 12 \\ 7 \\ 6 \end{pmatrix}$

y que $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$

Nos queda entonces: $A \cdot X = Y$
Lo que queremos es encontrar la matriz X, por lo que despejamos:

1. $A \cdot X = Y$
2. Se multiplica por el inverso de A, en ambos lados de la igualdad.
3. $(A^{-1})A \cdot X = (A^{-1})Y$
4. Como $A^{-1} \cdot A = I$, entonces: $I \cdot X = A^{-1}Y$
5. Como $I \cdot X = X$, entonces: $X = A^{-1}Y$

7.1.1. Ejercicios

Resuelve los siguientes sistemas de ecuaciones:

- $x - 3y + 2z = -3$
  $5x + 6y - z = 13$
  $4x - y + 3z = 8$

- $6x + y - z = 14$
  $4x + 2y - z = 13$
  $x + 2y + 4z = 12$

- $2a + b - 3c = 7$
  $5a - 4b + c = -19$
  $a - b - 4c = 4$

7.2. Ecuación Normal Lineal

Los siguientes datos fueron obtenidos de un muestreo realizado en un sensor de nivel, y su correspondiente verificación manual del nivel:

Salida Sensor	Nivel Real de Nivel %
A	Y
868	90
830	85
787	80
755	75
725	70
675	65
640	60
610	55
535	50
485	45
390	40
360	35
285	30
240	25
200	20
160	15
110	10

Tabla 7.1: Valores obtenidos del Sensor de Nivel y los valores reales de Nivel

Gráfica de los datos anteriores:

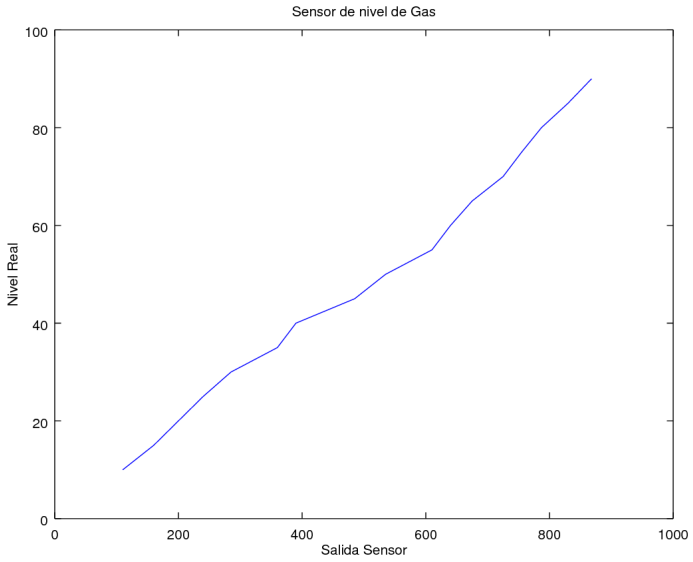


Figura 7.1: Gráfica Obtenida del Sensor contrastada con los valores reales

Lo que buscamos es encontrar la ecuación que caracteriza al sensor, y además asumimos que es lineal, por lo que la ecuación lineal debe tener la siguiente estructura: $C + A \cdot X = Y$, que puede reescribirse como $x_0 + x_1 \cdot A = Y$, donde:

- x_0 es una constante
- A es el valor obtenido del sensor
- x_1 es el coeficiente del valor obtenido del sensor
- Y es el nivel real para esa salida x_1 del sensor

De tal manera que la variable independiente es X y la variable dependiente es Y

Entonces, reescribimos la ecuación $A \cdot X = Y$ de la siguiente manera:

$$\left(A\right) \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = Y$$

Los valores para este ejemplo son:

$$A = \begin{pmatrix} 1 & 868 \\ 1 & 830 \\ 1 & 787 \\ 1 & 755 \\ 1 & 725 \\ 1 & 675 \\ 1 & 640 \\ 1 & 610 \\ 1 & 535 \\ 1 & 485 \\ 1 & 390 \\ 1 & 360 \\ 1 & 285 \\ 1 & 240 \\ 1 & 200 \\ 1 & 160 \\ 1 & 110 \end{pmatrix}, X = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, Y = \begin{pmatrix} 90 \\ 85 \\ 80 \\ 75 \\ 70 \\ 65 \\ 60 \\ 55 \\ 50 \\ 45 \\ 40 \\ 35 \\ 30 \\ 25 \\ 20 \\ 15 \\ 10 \end{pmatrix}$$

Lo que queremos es encontrar la matriz X, por lo que despejamos:

1. $A \cdot X = Y$
2. Se multiplica por el inverso de A, en ambos lados de la igualdad.
3. $(A^{-1})A \cdot X = (A^{-1})Y$
4. Sin embargo, no es posible obtener inversa de A, dado que A no es una matriz cuadrada; por lo que es necesario usar un truco matemático que nos permita realizar lo anterior.
5. Entonces, partiendo de lo siguiente: $A \cdot X = Y$
6. Podemos multiplicar ambos lados de la igualdad por un factor común.
7. $(A^T \cdot A) \cdot X = A^T \cdot Y$
8. Volvemos a multiplicar ambos lados de la igualdad por un factor común.
9. $(A^T \cdot A)^{-1} \cdot (A^T \cdot A) \cdot X = (A^T \cdot A)^{-1} \cdot A^T \cdot Y$
10. Lo que nos queda.
11. $I \cdot X = (A^T \cdot A)^{-1} \cdot A^T \cdot Y$
12. Lo que nos queda.
13. $X = (A^T \cdot A)^{-1} \cdot A^T \cdot Y$

Lo anterior se conoce como método de ***Ecuación Normal***, y es igual a:
$X = (A^T \cdot A)^{-1} \cdot A^T \cdot Y$
En octave es:
X= pinv(A'*A)*A'*Y

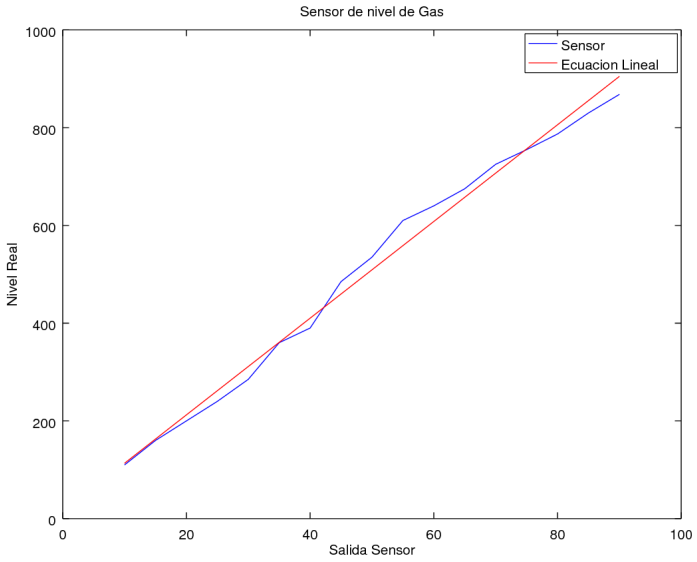


Figura 7.2: Gráfica Obtenida del Sensor contrastada con los valores reales y el modelado con la Ecuación Lineal

Como vemos hay errores, pero es una buena aproximación a la realidad, en ejemplos posteriores aproximaremos mejor la ecuación a los datos reales.

7.3. Error medio de las diferencias al cuadrado (J)

El error medio, nos da $\frac{1}{2}$ de las diferencias al cuadrado promedio entre los datos reales (entrenamiento) y los valores calculados con los parámetros obtenidos con el método de la ecuación normal (hipótesis).

$$J = \frac{1}{2m} \sum_{i=1}^{m} \left((y'^{(i)} - y^{(i)} \right)^2$$

7.3.1. Ejemplo

Del ejemplo que hemos estado usando, al hacer los cálculos, $X = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$, nos da $X = \begin{pmatrix} -0.96835 \\ 0.10011 \end{pmatrix}$

Cuando hacemos los cálculos con esta hipotética ecuación, obtenemos lo siguiente:

Salida Sensor	Nivel Real de Nivel en %	Valores Calculados
A	Y	Y'
868	90	85.928
830	85	82.124
787	80	77.819
755	75	74.616
725	70	71.612
675	65	66.607
640	60	63.103
610	55	60.099
535	50	52.591
485	45	47.586
390	40	38.075
360	35	35.072
285	30	27.563
240	25	23.058
200	20	19.054
160	15	15.049
110	10	10.044

Tabla 7.2: Valores obtenidos del Sensor de Nivel, los valores reales de Nivel y los valores calculados

Como vemos hay un error entre lo real y lo calculado, para medir el error se suele usar el cuadrado del error, para que al momento su sumar los errores no se cancelen entre sí, se puede hacer como muestra la siguiente tabla:

Salida Sensor	Nivel Real de Nivel en %	Valores Calculados	Error	$Error^2$
A	Y	Y'	Y'-Y	$(Y'-Y)^2$
868.00	90.00	85.93	-4.07	16.58
830.00	85.00	82.12	-2.88	8.27
787.00	80.00	77.82	-2.18	4.76
755.00	75.00	74.62	-0.38	0.15
725.00	70.00	71.61	1.61	2.60
675.00	65.00	66.61	1.61	2.58
640.00	60.00	63.10	3.10	9.63
610.00	55.00	60.10	5.10	26.00
535.00	50.00	52.59	2.59	6.71
485.00	45.00	47.59	2.59	6.69
390.00	40.00	38.07	-1.93	3.71
360.00	35.00	35.07	0.07	0.01
285.00	30.00	27.56	-2.44	5.94
240.00	25.00	23.06	-1.94	3.77
200.00	20.00	19.05	-0.95	0.90
160.00	15.00	15.05	0.05	0.00
110.00	10.00	10.04	0.04	0.00

Tabla 7.3: Valores obtenidos del Sensor de Nivel, los valores reales de Nivel y los valores calculados

Por lo que el promedio del error cuadrático medio es $\frac{98.285}{2 \cdot 17} = 2.8907$

7.4. Ecuaciones de segundo grado o de mayor grado

Como vemos en la gráfica, la ecuación lineal (de primer grado), no se adapta bien a los datos de entrenamiento, por lo que tenemos que incrementar el grado de la ecuación para ver si podemos modelar mejor a los datos de entrenamiento.

De esta manera, por ejemplo para un tercer grado, quedaría así:

Salida Sensor	Salida Sensor^2	Salida Sensor^3	Nivel Real de Nivel en %
A	A^2	A^3	Y
868.00	753424.00	653972032.00	90.00
830.00	688900.00	571787000.00	85.00
787.00	619369.00	487443403.00	80.00
755.00	570025.00	430368875.00	75.00
725.00	525625.00	381078125.00	70.00
675.00	455625.00	307546875.00	65.00
640.00	409600.00	262144000.00	60.00
610.00	372100.00	226981000.00	55.00
535.00	286225.00	153130375.00	50.00
485.00	235225.00	114084125.00	45.00
390.00	152100.00	59319000.00	40.00
360.00	129600.00	46656000.00	35.00
285.00	81225.00	23149125.00	30.00
240.00	57600.00	13824000.00	25.00
200.00	40000.00	8000000.00	20.00
160.00	25600.00	4096000.00	15.00
110.00	12100.00	1331000.00	10.00

Tabla 7.4: Valores obtenidos del Sensor de Nivel y elevados a la segunda y tercera potencia

Ahora la ecuación esperada es: $x_0 + x_1 \cdot A + x_2 \cdot A^2 + x_3 \cdot A^3 = Y$

$$A = \begin{pmatrix} 1.00 & 868.00 & 753424.00 & 653972032.00 \\ 1.00 & 830.00 & 688900.00 & 571787000.00 \\ 1.00 & 787.00 & 619369.00 & 487443403.00 \\ 1.00 & 755.00 & 570025.00 & 430368875.00 \\ 1.00 & 725.00 & 525625.00 & 381078125.00 \\ 1.00 & 675.00 & 455625.00 & 307546875.00 \\ 1.00 & 640.00 & 409600.00 & 262144000.00 \\ 1.00 & 610.00 & 372100.00 & 226981000.00 \\ 1.00 & 535.00 & 286225.00 & 153130375.00 \\ 1.00 & 485.00 & 235225.00 & 114084125.00 \\ 1.00 & 390.00 & 152100.00 & 59319000.00 \\ 1.00 & 360.00 & 129600.00 & 46656000.00 \\ 1.00 & 285.00 & 81225.00 & 23149125.00 \\ 1.00 & 240.00 & 57600.00 & 13824000.00 \\ 1.00 & 200.00 & 40000.00 & 8000000.00 \\ 1.00 & 160.00 & 25600.00 & 4096000.00 \\ 1.00 & 110.00 & 12100.00 & 1331000.00 \end{pmatrix}, X = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}, Y = \begin{pmatrix} 90 \\ 85 \\ 80 \\ 75 \\ 70 \\ 65 \\ 60 \\ 55 \\ 50 \\ 45 \\ 40 \\ 35 \\ 30 \\ 25 \\ 20 \\ 15 \\ 10 \end{pmatrix}$$

Calculando $X = (A^T \cdot A)^{-1} \cdot A^T \cdot Y$

$$X = \begin{pmatrix} 9.1182e-04 \\ 1.1938e-01 \\ -9.7328e-05 \\ 9.2200e-08 \end{pmatrix}$$

Graficando nuevamente:

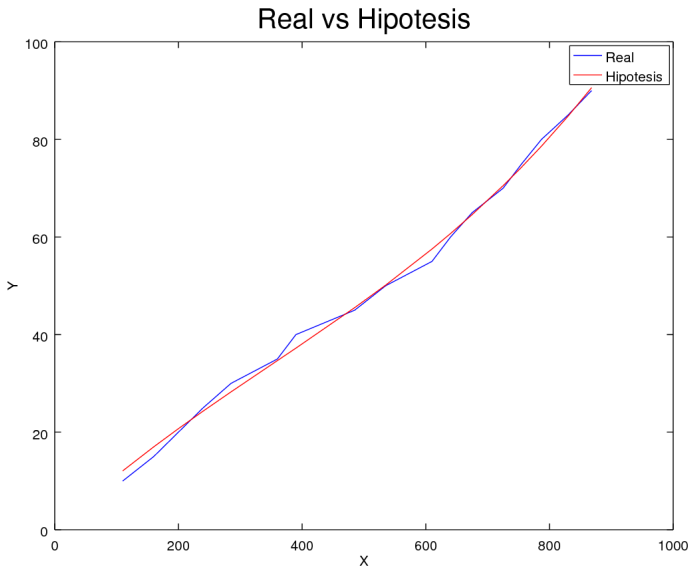


Figura 7.3: Gráfica Obtenida del Sensor contrastada con los valores reales y el modelado de la Ecuación de tercer grado

Como vemos en la gráfica, la Ecuación de tercer grado es una mejor aproximación que la ecuación lineal.

7.5. Ejemplos en Octave

Un programa que reciba como argumento de entrada: la matriz A, Y, y el grado mayor que se quiera investigar. Como argumento de salida: una matriz de todos los errores, el índice del mejor error de la matriz, la matriz de resultados X.

```
function [errores errorIndex X]=tarea1(B, Y, m)
n= size(B,1);
A=[ones(n,1) B];
errores=zeros(m,1);
errorIndex=1;
for i=1:m,
  X= pinv(A'*A)*A'*Y;
  tmp=(A*X-Y).^2;
  errores(i)=sum(tmp)/2/n;
  A=[A A(:,2).^(i+1)];
  if(errores(errorIndex)>errores(i)),
    errorIndex=i;
  end;%if
end;%for
A=A(:,1:(errorIndex+1));
X= pinv(A'*A)*A'*Y;
close all;
plot(A(:,2),Y);
hold on
plot(A(:,2),A*X,'r');
xlabel('X');
ylabel('Y');
legend('Real','Hipotesis');
title('Real vs Hipotesis','fontsize',20);
print -dpng 'tarea1.png'
end;%function
```

Listing 7.1: Descriptive Caption Text

7.6. Salida dependiente de multivariables

ejemplo

Concepto	casa 1	casa 2	casa 3	casa 4	casa 5	casa 6	casa 7	casa 8	casa 9	casa 10
baño	1	1	3	2.5	3	3	3	5	1	1.5
recamara	2	1	3	3	4	4	3	4	1	2
sala	0	0	1	1	1	1	1	1	0	0.5
cocina	1	1	1	1	1	1	1	2	1	1
comedor	1	0	1	1	1	1	1	1	0	0.5
cochera	0	0	1	0	1	1	1	2	0	0
estudio	0	0	2	0	0	1	0	2	0	0
lavado	0	0	0	0	1	1	1	1	0	0
area	90	90	300	150	288	225	300	400	72	90
plusvalia	1	1	1	1.2	1.3	1.3	1.5	1.6	0.8	1
Costo	280000	210000	765000	567000	972400	942500	1072500	1744000	153600	305000

Tabla 7.5: Tabla de Precios de Casas

Capítulo 8

Regresión Lineal con una variable

En estadística la regresión lineal o ajuste lineal, es un método matemático que modela la relación entre una variable dependiente Y, y variables independientes X_i, y un término aleatorio ε. [4]

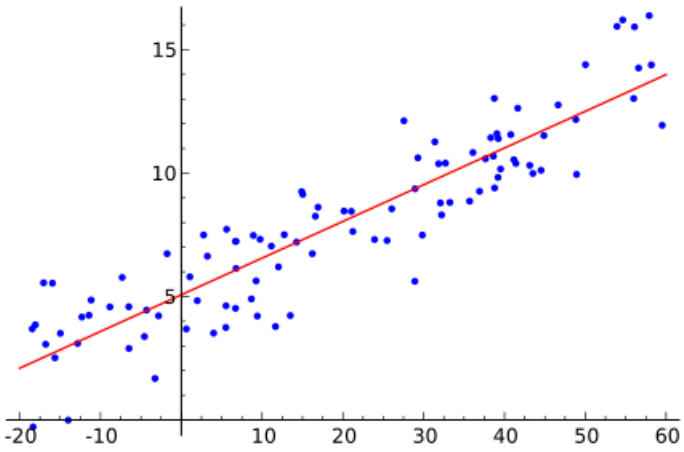


Figura 8.1: Ejemplo de regresión lineal de una variable independiente y una variable dependiente.

Ejemplo de base de datos de entrenamiento para precios de casa

m^2	Precio
195	460
131	232
142	315
80	178
⋮	⋮

Que nos daría una gráfica

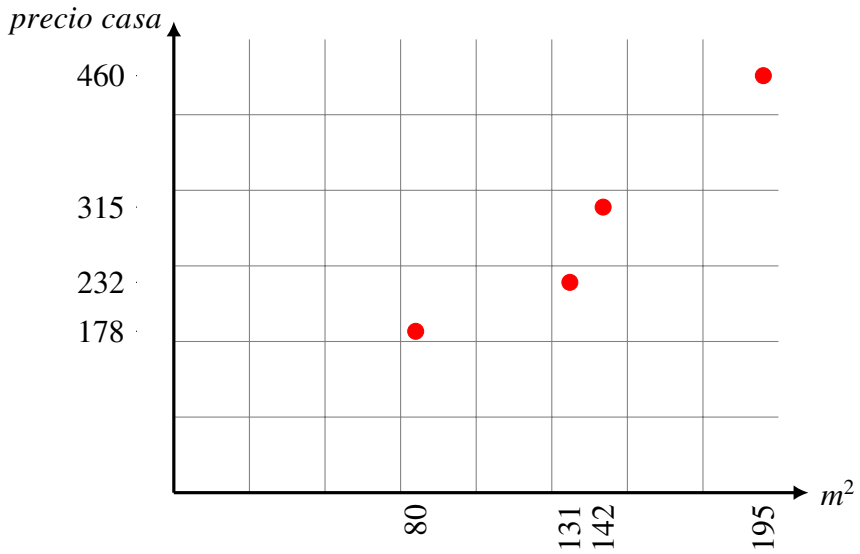


Figura 8.2: m^2-Precio casa

De lo anterior podemos establecer la siguiente notación:

m número de ejemplos de entrenamiento

$x's$ Variable de entrada ó también característica

$y's$ Variable de salida

(x, y) Un ejemplo de entrenamiento

$(x^{(i)}, y^{(i)})$ El í-esimo entrenamiento

8.1. Función Hipótesis

Podemos decir entonces que el tamaño de la casa es x, y que el precio estimado es y. Entonces nuestro algoritmo de aprendizaje deberá obtener un función que se conoce como hipótesis, que mapea los datos de $x \Rightarrow y$.
Desglosando los pasos:

- Tenemos una base de m datos de entrenamiento que usaremos para alimentar nuestro algoritmo de aprendizaje.
- Nuestro algoritmo de aprendizaje nos proporcionará como resultado de su análisis, una hipótesis.
- Nuestra hipótesis nos servirá para pronosticar un valor de salida (en este caso: precio de la casa), para cualquier posible entrada (área de la casa).

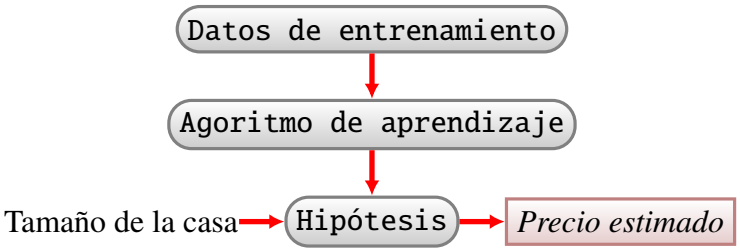


Figura 8.3: Objetivo del algoritmo de aprendizaje

La función hipótesis ($\hat{y}$) para este caso donde solo hay una variable de entrada, tiene la forma siguiente:
$\hat{y} = b + \Theta_1 x_1$
donde b y Θ_1 se conocen como *"parametros del modelo"* y entonces nuestro problema se convierte en encontrar los valores adecuados para los parametros del modelo.
Ejemplos de valores para los parametros de la hipótesis:

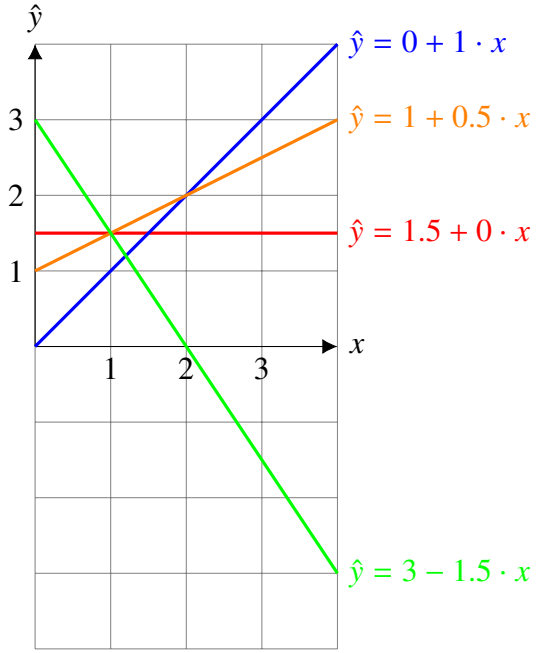


Figura 8.4: Grafica

La idea es escoger los valores de b y Θ_1, de tal forma que $\hat{y}$ esté lo más cerca a y para los entrenamientos (x, y). De esta forma, la idea general entonces es minimizar la sumatoria de las diferencias entre el valor de $\hat{y}^{(i)}$ y $y^{(i)}$; por lo que tenemos entonces que minimizar la sumatoria de las diferencias al cuadrado (al cuadrado para tener valores absolutos y magnificados los errores).

$$\sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2 \Longrightarrow 0 \Longrightarrow \textit{minimizar } b \text{ y } \Theta_1$$

pero como queremos el error promedio: $\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$

y para facilitar algunos calculos futuros, y sin afectar los resultados finales, ya que solamente reduces el error a la mitad:

$$\frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)}-y^{(i)}\right)^2$$

por convención, usaremos una función de costos J, cuya función es minimizar theta cero y theta uno:

$$J(\Theta)=\frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)}-y^{(i)}\right)^2$$

La función de costo J también es llamada función del error cuadrado (squared error function).

Resumiendo lo anterior:

Hipótesis $\hat{y}=b+\Theta_1 x_1$

Parámetros $b\ ,\Theta_1$

Función de error(loss) $\mathscr{L}(\hat{y},y)=\frac{1}{2}(\hat{y}-y)^2$

Función de costo $J(\Theta)=\frac{1}{m}\sum_{i=1}^{m}\mathscr{L}^i$

Objetivo minimizar $J(\Theta)$

NOTA: La función de error calcula el error para un solo ejemplo de entrenamiento, mientras que la función de costo es el promedio de las funciones de error de todo el conjunto de entrenamiento.

8.2. Función de Costo

Función de Costo

La función de costo, nos da $\frac{1}{2}$ de la diferencia promedio entre los entrenamientos y la hipótesis calculada.

$$J(\Theta)=\frac{1}{m}\sum_{i=1}^{m}\left(\mathscr{L}^i\right)^2$$

Ejemplo, si suponemos que nuestros entrenamientos son:

x	y
1	1
2	2
3	3

Nos daría la gráfica:

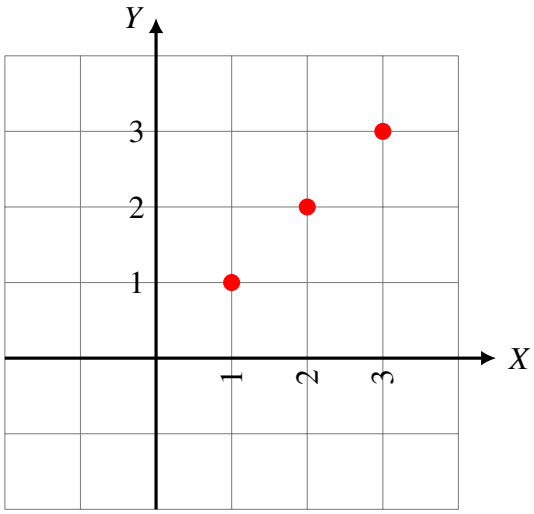


Figura 8.5: *ejemploCosto*

Tenemos la hipótesis $\hat{y} = b + \Theta_1 x_1$, y suponemos que nuestro algoritmo nos dio los valores de los parámetros de la siguiente forma: $b = 0$ y $\Theta_1 = 0.5$; de tal forma que la hipótesis queda: $\hat{y} = 0.5x$, y la evaluamos de la siguiente manera:

x	y	$\hat{y}$	$\mathscr{L}(\hat{y}, y)$
1	1	0.5	0.125
2	2	1	0.5
3	3	1.5	1.125

si la graficamos queda:

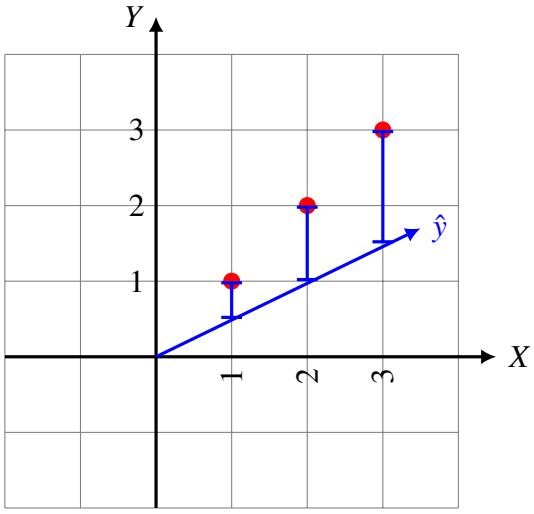


Figura 8.6: *ejemploCosto* $b = 0, \Theta_1 = 0.5$

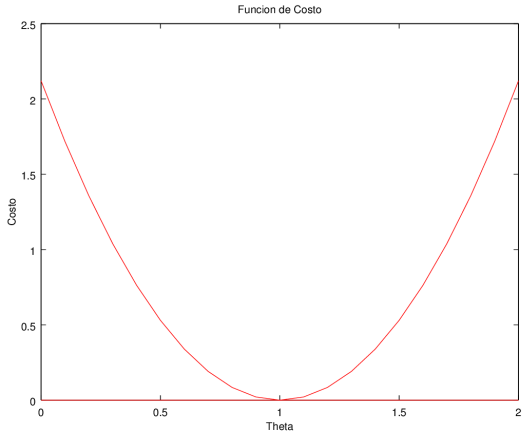


Figura 8.7: Función de Costo, respecto de Θ_1, con $b = 0$

8.3. Algoritmo descendente de la función de costo

Nuestro objetivo es encontrar los valores de los parámetros que minimicen el valor de la función de costo.
El algoritmo comienza con valores arbitrarios para los parámetros, y en cada iteración va ajustando los valores, de tal forma que en la siguiente iteración, la función de costo tenga un valor menor.
Esto se logra calculando la pendiente de la función de costo en ese punto dado, y usando esa pendiente ajustar los nuevos valores para la siguiente iteración.
Pseudocódigo del algoritmo descendente de la función de costo:

- Tenemos cualquier función de costo $J(\Theta)$
- Deseamos $\underset{\Theta}{minimizar}\ J(\Theta)$
- Entonces:
  - Empezamos con cualquier b, Θ_1
  - Cambiamos b, Θ_1 para reducir $J(\Theta)$ hasta alcanzar un mínimo.

Entrada: X, Y
Salida: Valores adecuados para los parámetros b y Θ_1

while No hay convergencia **do**
  for Θ **do**
    $b = b - \alpha \frac{\partial}{\partial b} J(\Theta)$
    $\Theta_1 = \Theta_1 - \alpha \frac{\partial}{\partial \Theta_1} J(\Theta)$
  end for
end while
Desplegar los valores de b y Θ.

Figura 8.8: Algoritmo descendente del gradiente

α se conoce como coeficiente de aprendizaje.

Los valores de las derivadas parciales son:

$$b\colon \frac{\partial}{\partial b}J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)$$

$$\Theta_1\colon \frac{\partial}{\partial \Theta_1}J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)x^{(i)}$$

como sabemos que $\hat{y}^{(i)} = b + \Theta_1 x_1^{(i)}$, entonces:

$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(b + \Theta_1 x_1^{(i)} - y^{(i)}\right)^2$$

Entonces los valores de las derivadas parciales son:

$$b\colon \frac{\partial}{\partial b}J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(b + \Theta_1 x_1^{(i)} - y^{(i)}\right)$$

$$\Theta_1\colon \frac{\partial}{\partial \Theta_1}J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(b + \Theta_1 x_1^{(i)} - y^{(i)}\right)x^{(i)}$$

Por lo que el algoritmo puede quedar:

Entrada: m duplas de entrenamientos
Salida: Valores adecuados para los parámetros b y Θ_1

1: $b = 0$
2: $\Theta_1 = 0$
3: **while** No hay convergencia **do**
4: $Acum_0 = 0$
5: $Acum_1 = 0$
6: **for** i = 1 hasta m **do**
7: $Acum_0 = Acum_0 + \left(b + \Theta_1 x_1^{(i)} - y^{(i)}\right)$
8: $Acum_1 = Acum_1 + \left(b + \Theta_1 x_1^{(i)} - y^{(i)}\right)x_1^{(i)}$
9: **end for**
10: $Acum_0 = \frac{Acum_0}{m} * \alpha$
11: $Acum_1 = \frac{Acum_1}{m} * \alpha$
12: $b = b - Acum_0$
13: $\Theta_1 = \Theta_1 - Acum_1$
14: **end while**
15: Desplegar los valores de b y Θ_1.

Figura 8.9: Algoritmo descendente del gradiente

Capítulo 9

Regresión Lineal con múltiples variables

9.1. Notación

La regresión lineal con una sola variable contempla una variable independiente X y una dependiente Y, como en el caso del ejemplo de la casa, donde el precio de la casa solo depende de los m^2 de área.

m^2	Precio
195	460
131	232
142	315
80	178
⋮	⋮

X	Y
195	460
131	232
142	315
80	178
⋮	⋮

Figura 9.1: Ejemplo de Regresion lineal con 1 variable independiente

Sin embargo, solamente en pocas ocasiones encontraremos que una variable dependiente solo dependa de una variable independiente; en la mayoría de las ocasiones, dependerá de varias variables independientes y posiblemente de las relaciones entre ellas.
Siguiendo el ejemplo de la casa; es más probable que el precio de la casa dependa de varios factores, como por ejemplo: área, número de recámaras, número de plantas, plusvalía de la colonia, años de construida, etc. Quedando una tabla más compleja:
A= años
P= plusvalía
F= pisos
R= recámaras

A	P	F	R	m^2	Precio
5	.9	3	4	195	460
6	.4	2	2	131	232
4	.9	2	3	142	315
7	.6	1	1	80	178

Variables Independientes					Salida
X_1	X_2	X_3	X_4	X_5	Y
5	.9	3	4	195	460
6	.4	2	2	131	232
4	.9	2	3	142	315
7	.6	1	1	80	178

Figura 9.2: Ejemplo de Regresion lineal con varias variables independientes

De lo anterior podemos establecer la siguiente notación:

m número de ejemplos de entrenamiento

n número de características (variables independientes)

$x_j^{(i)}$ valor de la característica j del i^{th} entrenamiento

9.2. Función Hipótesis

La función hipótesis (h) para este caso donde solo hay múltiples variables de entrada, tiene la forma siguiente:
$h_\Theta(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_3 + \ldots + \Theta_n x_n$

9.3. Función de Costo

Función de Costo
La función de costo, nos da $\frac{1}{2}$ de la diferencia promedio entre los entrenamientos y la hipótesis calculada.
$J(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\Theta(x^{(i)}) - y^{(i)} \right)^2$

9.4. Algoritmo descendente del gradiente

Nuestro objetivo es encontrar los valores de theta que minimicen el valor de la función de costo; es decir, queremos encontrar los valores óptimos de theta para nuestra función hipótesis.
El algoritmo comienza con unos valores arbitrarios para theta, y en cada iteración va ajustando el valor de theta, de tal forma que en la siguiente iteración, la función de costo tenga un valor menor.
Esto se logra calculando la pendiente de la función de costo en ese punto dado, y usando esa pendiente ajustar el nuevo valor de theta para la siguiente iteración.
Pseudocódigo del algoritmo descendente del gradiente:

- Tenemos cualquier función de costo $J(\Theta_0)$
- Deseamos $\underset{\Theta_0,\Theta_1,\Theta_2,\Theta_3,\ldots,\Theta_n}{minimizar} \; J(\Theta)$
- Entonces:
    - Empezamos con cualquier $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$
    - Cambiamos $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$ para reducir $J(\Theta)$ hasta alcanzar un mínimo.

Algoritmo descendente del gradiente para n características

Entrada: m entrenamientos

Salida: Valores adecuados para los parámetros $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$

1: **while** No hay convergencia **do**
2: **for** $\Theta_0 \rightarrow \Theta_n$ **do**
3: $\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta)$
4: **end for**
5: **end while**
6: Desplegar los valores de Θ_j.

Figura 9.3: Algoritmo descendente del gradiente.

α se conoce como coeficiente de aprendizaje.

Los valores de las derivadas parciales son:

$$j = 0\colon \frac{\partial}{\partial \Theta_0} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\Theta(x^{(i)}) - y^{(i)} \right)$$

$$j = 1\colon \frac{\partial}{\partial \Theta_1} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\Theta(x^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

$$\vdots$$

$$j = n\colon \frac{\partial}{\partial \Theta_n} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\Theta(x^{(i)}) - y^{(i)} \right) x_m^{(i)}$$

como sabemos que $h_\Theta(x^{(i)}) = \Theta_0 + \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)}$**, entonces:**

$$J(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\Theta_0 + \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)} - y^{(i)} \right)^2$$

Si hacemos que $x_0^{(i)} = 1; entonces$:

$$J(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\Theta_0 x_0^{(i)} + \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)} - y^{(i)} \right)^2$$

Entonces los valores de las derivadas parciales son:

$$j = k\colon \frac{\partial}{\partial \Theta_k} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\Theta_0 x_0^{(i)} + \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)} - y^{(i)} \right) x_k^{(i)}$$

Por lo que el algoritmo puede quedar:

Entrada: m duplas de entrenamientos
Salida: Valores adecuados para los parámetros $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$

1: $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n = 0$
2: **while** No hay convergencia **do**
3: $Acum_0, Acum_1, Acum_2, Acum_3, \ldots, Acum_n = 0$
4: **for** i = 1 hasta m **do**
5: **for** j = 0 hasta n **do**
6: $Acum_j = Acum_j + \left(\Theta_0 x_0^{(i)} + \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)} - y^{(i)}\right) x_j^{(i)}$
7: **end for**
8: **end for**
9: **for** j = 0 hasta n **do**
10: $Acum_j = \frac{Acum_j}{m} * \alpha$
11: $\Theta_j = \Theta_j - Acum_j$
12: **end for**
13: **end while**
14: Desplegar los valores de $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$.

Figura 9.4: Algoritmo descendente del gradiente

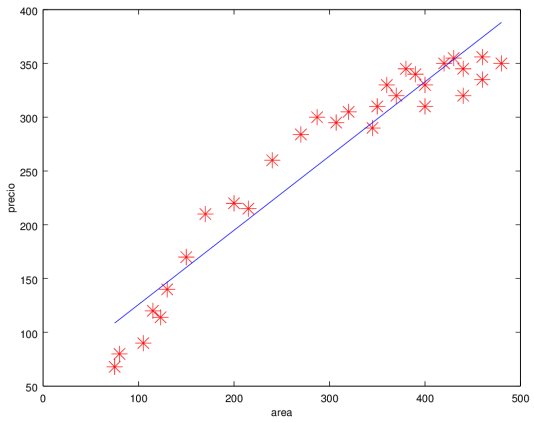
9.5. Usando Características No Lineales

Siguiendo con el mismo ejemplo de predecir el valor de una casa, dependiendo de sus dimensiones, podemos suponer que una función hipótesis queda de la siguiente manera: $h_\Theta(x) = \Theta_0 + \Theta_1 \cdot Frente + \Theta_2 \cdot Fondo$
Pero también podemos ver que en realidad el precio depende del área en sí, no de las medidas individuales de frente y fondo; por lo que es conveniente mejor representar de la siguiente manera: $h_\Theta(x) = \Theta_0 + \Theta_1 \cdot area$
Si tenemos los siguientes entrenamientos:

```
75, 68
80, 80
105, 90
115, 120
130, 140
123, 114
150, 170
170, 210
200, 220
215, 215
240, 260
270, 284
287, 300
307, 295
320, 305
345, 290
350, 310
360, 330
370, 320
380, 345
390, 340
400, 310
400, 330
420, 350
430, 355
440, 320
440, 345
460, 335
460, 356
480, 350
```

Listing 9.1: Entrenamientos (area - precio)

Nos daría la siguiente gráfica usando la ecuación normal:

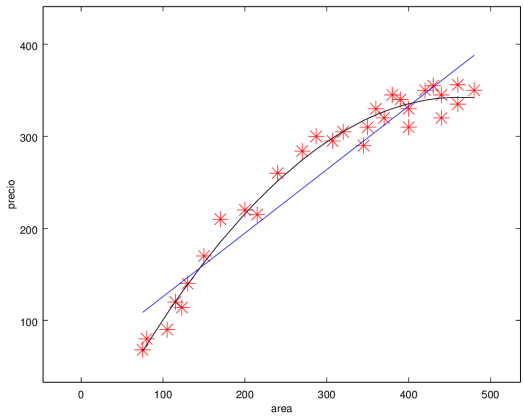


Como vemos en la gráfica anterior, la recta de la hipótesis, no se ajusta muy bien a los datos. En éstos casos conviene agregar una característica adicional, que sería el área^2, para de ésta manera tener una gráfica un poco más parecida al comportamiento de los precios.
Por lo que quedaría la función hipótesis de la siguiente manera: $h_\Theta(x) = \Theta_0 + \Theta_1 \cdot area + \Theta_2 \cdot area^2$
Tenemos entonces los siguientes entrenamientos:

```
 7.50000000e+01 5.62500000e+03 6.80000000e+01
 8.00000000e+01 6.40000000e+03 8.00000000e+01
 1.05000000e+02 1.10250000e+04 9.00000000e+01
 1.15000000e+02 1.32250000e+04 1.20000000e+02
 1.30000000e+02 1.69000000e+04 1.40000000e+02
 1.23000000e+02 1.51290000e+04 1.14000000e+02
 1.50000000e+02 2.25000000e+04 1.70000000e+02
 1.70000000e+02 2.89000000e+04 2.10000000e+02
 2.00000000e+02 4.00000000e+04 2.20000000e+02
 2.15000000e+02 4.62250000e+04 2.15000000e+02
 2.40000000e+02 5.76000000e+04 2.60000000e+02
 2.70000000e+02 7.29000000e+04 2.84000000e+02
 2.87000000e+02 8.23690000e+04 3.00000000e+02
 3.07000000e+02 9.42490000e+04 2.95000000e+02
 3.20000000e+02 1.02400000e+05 3.05000000e+02
 3.45000000e+02 1.19025000e+05 2.90000000e+02
 3.50000000e+02 1.22500000e+05 3.10000000e+02
 3.60000000e+02 1.29600000e+05 3.30000000e+02
 3.70000000e+02 1.36900000e+05 3.20000000e+02
 3.80000000e+02 1.44400000e+05 3.45000000e+02
 3.90000000e+02 1.52100000e+05 3.40000000e+02
 4.00000000e+02 1.60000000e+05 3.10000000e+02
 4.00000000e+02 1.60000000e+05 3.30000000e+02
 4.20000000e+02 1.76400000e+05 3.50000000e+02
 4.30000000e+02 1.84900000e+05 3.55000000e+02
 4.40000000e+02 1.93600000e+05 3.20000000e+02
 4.40000000e+02 1.93600000e+05 3.45000000e+02
 4.60000000e+02 2.11600000e+05 3.35000000e+02
 4.60000000e+02 2.11600000e+05 3.56000000e+02
 4.80000000e+02 2.30400000e+05 3.50000000e+02
```

Listing 9.2: Entrenamientos (area - area*area - precio)

Y nos daría la siguiente gráfica usando la ecuación normal, donde se ve claramente que ya no es más una ecuación lineal, sino una función polinomial cuadrática:



Capítulo 10

Regresión Logística

Actualmente uno de los algoritmos más ampliamente usados en el problema de clasificación es el de regresión logística. Algunos ejemplos del problema de clasificación son:

email ¿spam / no spam ?

transacción en línea ¿fraudulenta (sí / no)?

tumor ¿maligno / benigno?

En todos los problemas de clasificación simple tenemos una variable $y \in \{0, 1\}$,

donde: $y = \begin{cases} 0: & \text{clase negativa, ejemplo tumor benigno,} \\ 1: & \text{clase positiva, ejemplo tumor maligno.} \end{cases}$

También está el problema de clasificación multiclase, donde tenemos varios posibles casos de clasificación, ejemplo clasificar el canto de un pájaro: $y \in \{0, 1, 2, 3, 4, 5\}$,

donde: $y = \begin{cases} 0: & \text{clase negativa, ejemplo no es ningún canto,} \\ 1: & \text{clase 1, canto gorrión,} \\ 2: & \text{clase 2, canto golondrina,} \\ 3: & \text{clase 3, canto jilguero,} \\ 4: & \text{clase 4, canto canario,} \\ 5: & \text{clase 5, canto chachalaca,} \end{cases}$

otro ejemplo, tipo de email: $y \in \{1, 2, 3, 4\}$,

donde: $y = \begin{cases} 1: & \text{clase 1, trabajo,} \\ 2: & \text{clase 2, amigos,} \\ 3: & \text{clase 3, familia,} \\ 4: & \text{clase 4, hobby.} \end{cases}$

otro ejemplo, pronosticar el tiempo: $y \in \{1, 2, 3, 4\}$,

donde: $y = \begin{cases} 1: & \text{clase 1, soleado,} \\ 2: & \text{clase 2, nublado,} \\ 3: & \text{clase 3, lluvia,} \\ 4: & \text{clase 4, nieve.} \end{cases}$

Sin embargo, como veremos más adelante el problema de clasificación múltiple puede construirse a partir del problema de clasificación simple, el cual es: y = 0 ó 1

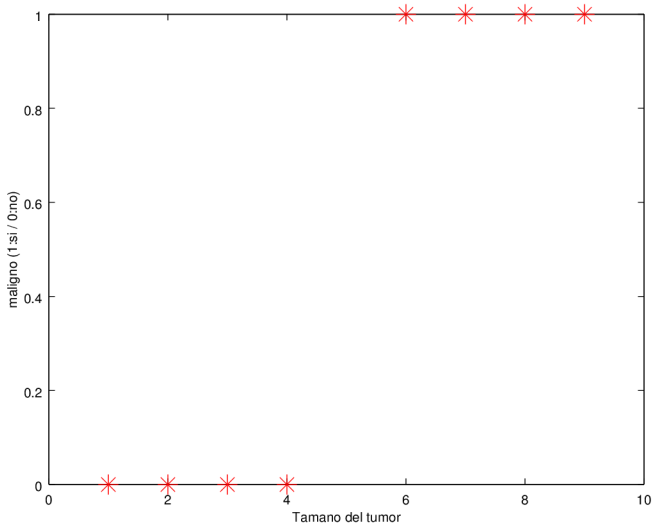
10.1. Clasificador

Tenemos los siguientes entrenamientos:

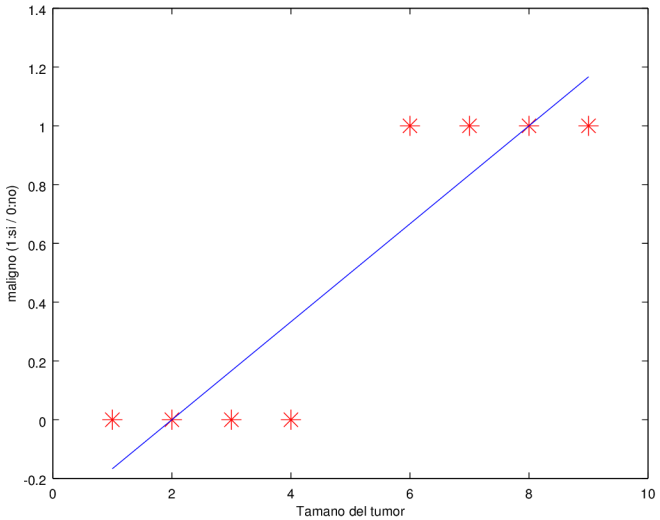
```
1.00000000e+00 0.00000000e+00
2.00000000e+00 0.00000000e+00
3.00000000e+00 0.00000000e+00
4.00000000e+00 0.00000000e+00
6.00000000e+00 1.00000000e+00
7.00000000e+00 1.00000000e+00
8.00000000e+00 1.00000000e+00
9.00000000e+00 1.00000000e+00
```

Listing 10.1: Entrenamientos (tamano del tumor - maligno)

Que nos da la siguiente gráfica:



Buscando nuestra $h_\Theta(x)$ y graficando, quedaría así:



Como acabamos de ver, la regresión lineal como tal, no nos sirve para decidir si el tamaño del tumor es maligno o no.
Debido principalmente a que el problema de clasificación esta acotada a: $y = \{0 \text{ ó } 1\}$
mientras que la $h_\Theta(x)$ es una ecuación que no esta acotada.
Lo que hacemos entonces es usar un algoritmo de clasificación conocido como regresión logística, donde: $0 \leq h_\Theta(x) \leq 1$
básicamente lo que se hace es convertir nuestra $h_\Theta(x) = \Theta^T x$ del modelo de regresión lineal, al nuevo modelo de regresión logística $h_\Theta(x) = g(\Theta^T x)$, donde $g(z) = \dfrac{1}{1 + e^{-z}}$, por lo que quedaría nuestra nueva hipótesis de regresión logística de la siguiente manera: $h_\Theta(x) = \dfrac{1}{1 + e^{-\Theta^T \cdot x}}$, de ésta manera acotamos la hipótesis a los valores intermedios entre 0 y 1; ya que la gráfica de ésta ecuación es asintótica a 0 y 1; como se ve en la siguiente gráfica.

$g(z) = \frac{1}{1+e^{-z}}$

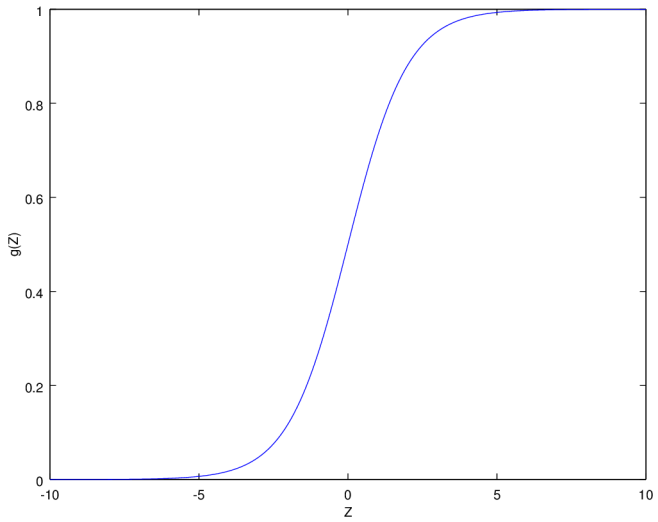


Figura 10.1: g(z)

La interpretación de lo anterior es que nuestra hipótesis $h_\Theta(x)$ nos da la probabilidad estimada de que $Y = 1$ en una entrada X
Ejemplo: si $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tamaño del tumor} \end{bmatrix}$
si $h_\Theta(x) = 0.7$
Indicaría al paciente que tiene un 70 % de que el tumor sea maligno.

$h_\Theta(x) = P(y = 1|x; \Theta)$
La ecuación anterior, se puede leer de la siguiente manera: probabilidad de que Y sea 1, dado X, con la paramétrica por Θ
Sabemos que $P(y = 0|x; \Theta) + P(y = 1|x; \Theta) = 1$
por lo tanto: $P(y = 0|x; \Theta) = 1 - P(y = 1|x; \Theta)$

10.2. Regresión Logística

Sabemos que:

- $h_\Theta(x) = g(\Theta^T x)$
- $g(z) = \frac{1}{1+e^{-z}}$

Supón que queremos predecir que $y = 1$, como vemos en la imagen de g(z), ésto sucede cuando $h_\Theta(x) \geq 0.5$; lo que a su vez sucede cuando $\Theta^T x \geq 0$

Para predecir que $y = 0$, como vemos ésto sucede cuando $h_\Theta(x) < 0.5$; lo que a su vez sucede cuando $\Theta^T x < 0$

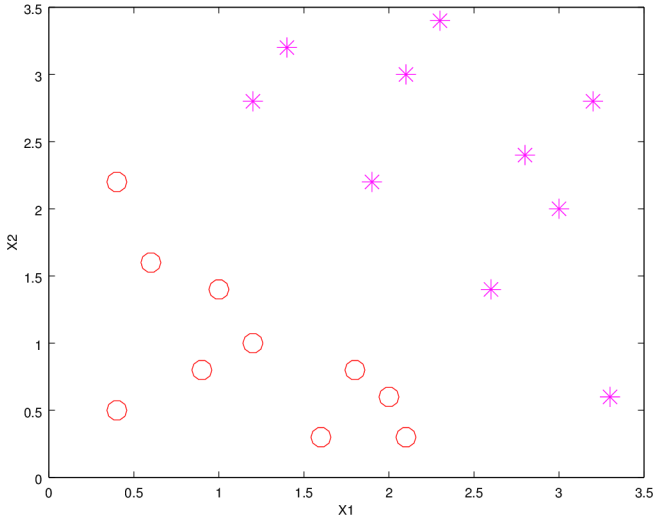
10.3. Frontera Lineal de Decisión

Con los siguientes entrenamientos:

```
0.4 0.5 0
0.4 2.2 0
0.6 1.6 0
0.9 0.8 0
1 1.4 0
1.2 1 0
1.6 0.3 0
1.8 0.8 0
2 0.6 0
2.1 0.3 0
1.2 2.8 1
1.4 3.2 1
1.9 2.2 1
2.1 3 1
2.3 3.4 1
2.6 1.4 1
2.8 2.4 1
3 2 1
3.2 2.8 1
3.3 0.6 1
```

Listing 10.2: Entrenamientos (x1-x2-Y)

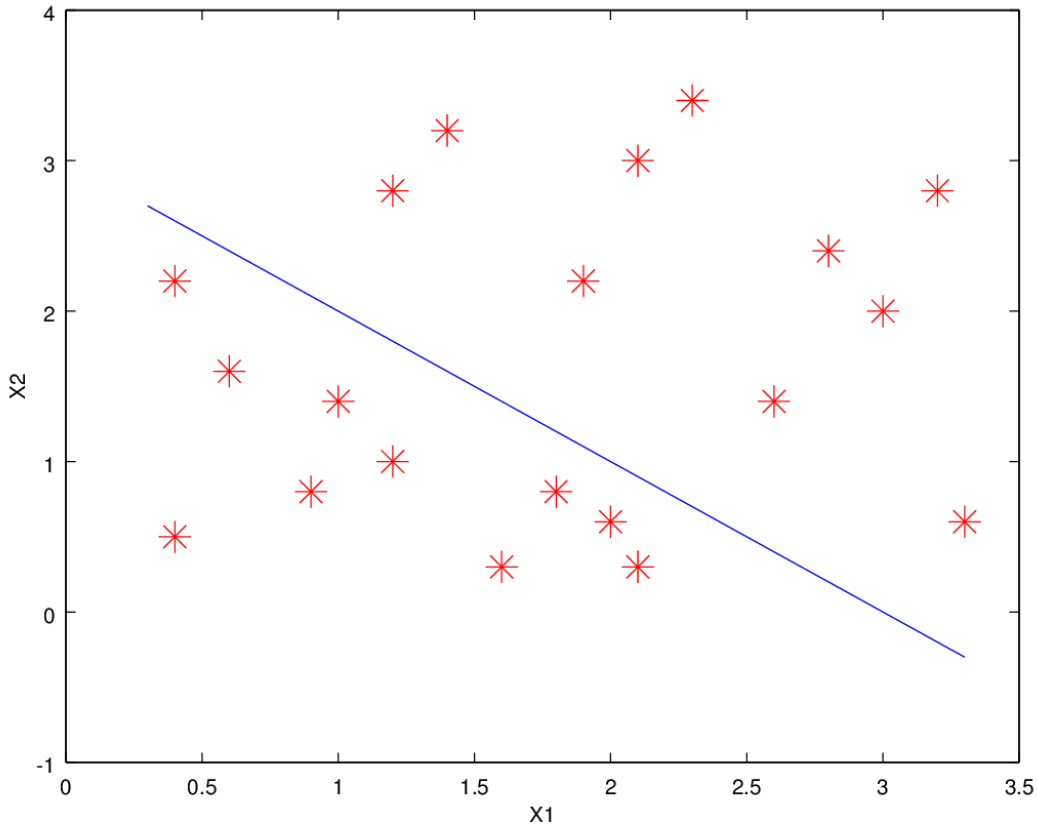
Nos da la siguiente gráfica:



Si suponemos que $h_\Theta(x) = g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2)$
Por lo pronto supondremos también los valores de Θ_i, con los siguientes valores:

- $\Theta_0 = -3$
- $\Theta_1 = 1$
- $\Theta_2 = 1$

entonces tenemos: $h_\Theta(x) = g(-3 + x_1 + x_2)$
Si queremos predecir cuando $y = 1$, entonces como habíamos mencionado ésto sucede cuando $h_\Theta(x) \geq 0.5$; lo que a su vez sucede cuando $\Theta^T x \geq 0$, y sí $\Theta^T x = -3 + x_1 + x_2$, por lo que y=1, cuando: $-3 + x_1 + x_2 \geq 0$; es decir cuando $x_1 + x_2 \geq 3$; que nos da la siguiente frontera lineal de decisión.


4
3
2
1
0
-1
X2
0
0.5
1
1.5
2
2.5
3
3.5
X1

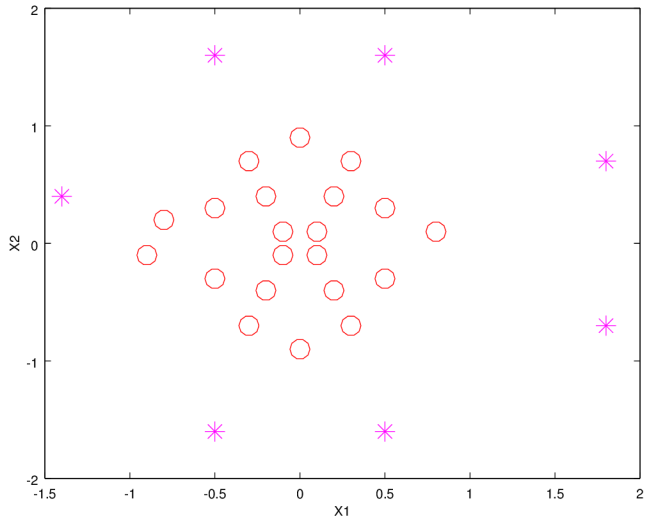
10.4. Frontera No-Lineal de Decisión

Con los siguientes entrenamientos:

```
-0.8 0.2 0
-0.9 -0.1 0
0.8 0.1 0
0.0 0.9 0
0.0 -0.9 0
0.3 0.7 0
0.2 0.4 0
0.3 -0.7 0
0.2 -0.4 0
0.5 0.3 0
0.5 -0.3 0
-0.3 0.7 0
-0.3 -0.7 0
-0.2 0.4 0
-0.2 -0.4 0
-0.5 0.3 0
-0.5 -0.3 0
-0.1 0.1 0
-0.1 -0.1 0
0.1 0.1 0
0.1 -0.1 0
0.5 1.6 1
0.5 -1.6 1
-0.5 -1.6 1
-0.5 1.6 1
1.8 0.7 1
1.8 -0.7 1
-1.4 0.4 1
```

Listing 10.3: Entrenamientos (x1-x2-Y)

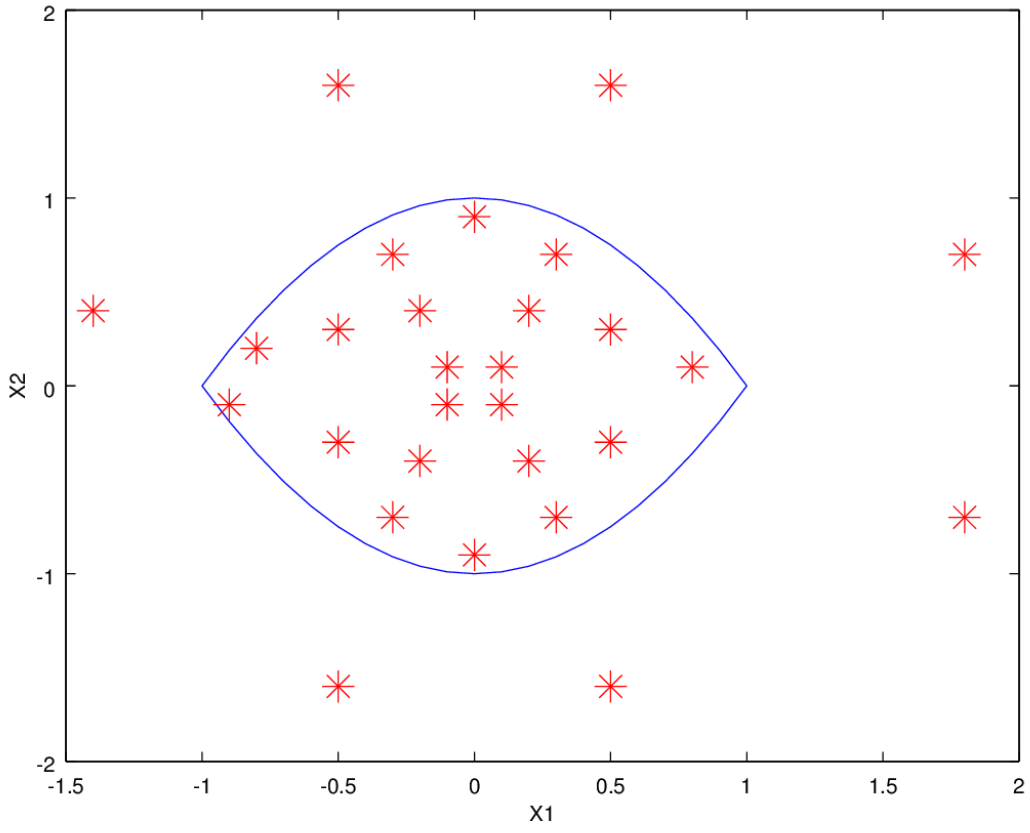
Nos da la siguiente gráfica:



Si suponemos que $h_\Theta(x) = g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1^2 + \Theta_4 x_2^2)$
Por lo pronto supondremos también los valores de Θ_i, con los siguientes valores:

- $\Theta_0 = -1$
- $\Theta_1 = 0$
- $\Theta_2 = 0$
- $\Theta_3 = 1$
- $\Theta_4 = 1$

entonces tenemos: $h_\Theta(x) = g(-1 + x_1^2 + x_2^2)$
Si queremos predecir cuando $y = 1$, entonces como habíamos mencionado ésto sucede cuando $h_\Theta(x) \geq 0.5$; lo que a su vez sucede cuando $\Theta^T x \geq 0$, y sí $\Theta^T x = -1 + x_1^2 + x_2^2$, por lo que y=1, cuando: $-1 + x_1^2 + x_2^2 \geq 0$; es decir cuando $x_1^2 + x_2^2 \geq 1$; que nos da la siguiente frontera lineal de decisión.



10.5. Función de Costo

Hasta este momento hemos visto los ejemplos con los valores del vector Θ resuelto, sin embargo, ¿cómo escogemos los valores correctos de Θ?
Repasando como calculábamos la función de costo en regresión lineal:

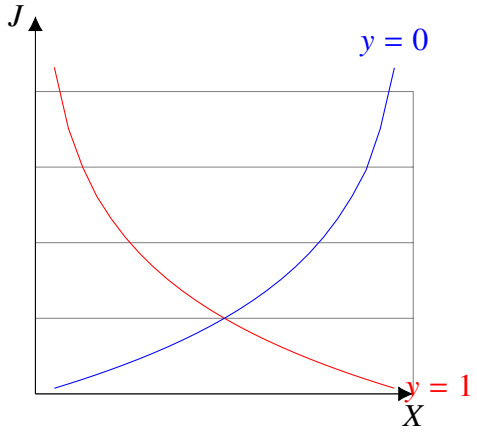
$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta(x^{(i)}) - y^{(i)}\right)^2$$

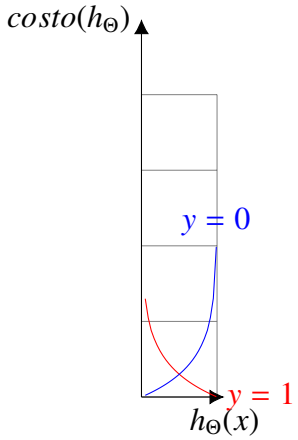
Necesitamos un algoritmo nuevamente para buscar los valores de Θ; y el algoritmo que veremos es similar al de descenso del gradiente que vimos en regresión lineal; con la diferencia de que el error calculado ya no es la diferencia entre la hipótesis y la salida del entrenamiento y; dado que ahora y tiene solamente 2 valores posibles.
Ahora la ecuación que usaremos será:

$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m} costo(h_\Theta(x^{(i)}, y^{(i)})$$

Donde: $costo(h_\Theta(x^{(i)}, y) = \begin{cases} si\ y = 1: & -log(h_\Theta(x)), \\ si\ y = 0: & -log(1 - h_\Theta(x)) \end{cases}$

Donde la idea es que la penalización que dará el algoritmo a las $h_\Theta(x)$ equivocadas será alta.





Resumiendo lo anterior, la función de costo podría quedar así:
$cost(h_\Theta(x), y) = -y \cdot log(h_\Theta(x)) - (1 - y) \cdot log(1 - h_\Theta(x)))$
Por lo tanto, la función de regresión logística podría quedar:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \cdot log(h_\Theta(x^{(i)})) + (1 - y^{(i)}) \cdot log(1 - h_\Theta(x^{(i)}))\right]$$

Función de Costo

La función de costo, es entonces: $J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \cdot log(h_\Theta(x^{(i)})) + (1 - y^{(i)}) \cdot log(1 - h_\Theta(x^{(i)}))\right]$

10.6. Algoritmo descendente de la función de costo

Nuestro objetivo es encontrar los valores de theta que minimicen el valor de la función de costo; es decir, queremos encontrar los valores óptimos de theta para nuestra función hipótesis.
El algoritmo comienza con unos valores arbitrarios para theta, y en cada iteración va ajustando el valor de theta, de tal forma que en la siguiente iteración, la función de costo tenga un valor menor.
Esto se logra calculando la pendiente de la función de costo en ese punto dado, y usando esa pendiente ajustar el nuevo valor de theta para la siguiente iteración.
Pseudocódigo del algoritmo descendente de la función de costo:

- Tenemos cualquier función de costo $J(\Theta_0)$
- Deseamos $\underset{\Theta_0,\Theta_1,\Theta_2,\Theta_3,\ldots,\Theta_n}{minimizar}\ J(\Theta)$
- Entonces:
  - Empezamos con cualquier $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$
  - Cambiamos $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$ para reducir $J(\Theta)$ hasta alcanzar un mínimo.

Algoritmo descendente del gradiente para n características
Entrada: m entrenamientos
Salida: Valores adecuados para los parámetros $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$

1: **while** No hay convergencia **do**
2: **for** $\Theta_0 \rightarrow \Theta_n$ **do**
3: $\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta)$
4: **end for**
5: **end while**
6: Desplegar los valores de Θ_j.

Figura 10.2: Algoritmo descendente del gradiente.

α se conoce como coeficiente de aprendizaje.
Los valores de las derivadas parciales son:

$$j = 0: \ \frac{\partial}{\partial \Theta_0} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$j = 1: \ \frac{\partial}{\partial \Theta_1} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\vdots$$

$$j = n: \ \frac{\partial}{\partial \Theta_n} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_n^{(i)}$$

como sabemos que:
$\Theta^T x = \Theta_0 x_0^{(i)} + \Theta_1 x_1^{(i)} + \Theta_2 x_2^{(i)} + \cdots + \Theta_n x_n^{(i)}$,
y que:

$g(z) = \dfrac{1}{1 + e^{-z}}$
y que:
$h_\Theta(x) = g(\Theta^T x) = \dfrac{1}{1 + e^{-\Theta^T x}}$
Entonces los valores de las derivadas parciales son:
$j = k\colon \dfrac{\partial}{\partial \Theta_k} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_k^{(i)}$

Por lo que el algoritmo puede quedar:

Entrada: m duplas de entrenamientos
Salida: Valores adecuados para los parámetros $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$

1: $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n = 0$
2: **while** No hay convergencia **do**
3: $Acum_0, Acum_1, Acum_2, Acum_3, \ldots, Acum_n = 0$
4: **for** i = 1 hasta m **do**
5: **for** j = 0 hasta n **do**
6: $Acum_j = Acum_j + (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ {sabemos que: $h_\Theta(x) = g(\Theta^T x) = \dfrac{1}{1 + e^{-\Theta^T x}}$}
7: **end for**
8: **end for**
9: **for** j = 0 hasta n **do**
10: $Acum_j = \dfrac{Acum_j}{m} * \alpha$
11: $\Theta_j = \Theta_j - Acum_j$
12: **end for**
13: **end while**
14: Desplegar los valores de $\Theta_0, \Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n$.

Figura 10.3: Algoritmo descendente del gradiente 2

10.7. Usar librerías de Octave en lugar del algoritmo descendente del gradiente

Las opciones son:

- Algoritmo descendente del gradiente
- Gradiente conjugado
- BFGS
- L-BFGS

Ligas interesantes:

- `https://en.wikipedia.org/wiki/Conjugate_gradient_method`
- `http://mmc2.geofisica.unam.mx/cursos/tedcd/GradienteConjugado.pdf`

- https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno
- http://www.chokkan.org/software/liblbfgs/

Capítulo 11

Redes Neuronales

11.1. Introducción

Si tenemos datos como se ve en la gráfica **??** encontrar la combinación de parámetros correctos ya se vuelve una tarea descomunal, pues tendriamos que explorar muchas combinaciones posibles, un ejemplo de como podría quedar sería lo siguiente: $\theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot x_1 \cdot x_2 + \theta_4 \cdot x_1^2 \cdot x_2 + \theta_5 \cdot x_1^3 \cdot x_2 + \theta_6 \cdot x_1 \cdot x_2^2 + \cdots$

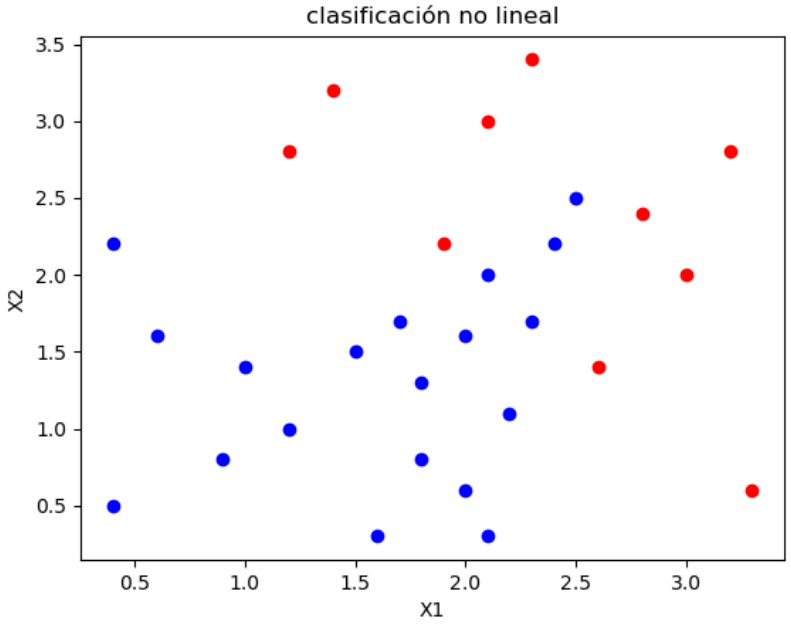


Figura 11.1: Clasificación no-lineal

Es entonces, que ahora recurrimos a las redes neuronales, que desde sus origenes fueron algoritmos que intentaron imitar el cerebro.
Fue muy utilizado en los años 80 y principios de los 90; su popularidad disminuyó a finales de los 90; y tuvieron un resurgimiento reciente, como una técnica de vanguardia para muchas aplicaciones de inteligencia artificial, gracias en parte a los avances que se realizaron en poder de cómputo de las PC.
La clave es que en lugar de crear una sola ecuación que modele todo el comportamiento de los datos, ahora usaremos muchos clasificadores lineales y la solución final es la suma de todos ellos.

11.2. Neurona

La neurona es la unidad del todo en el cerebro, es decir, el cerebro esta formado por millones de neuronas y sus interconexiones entre ellas.

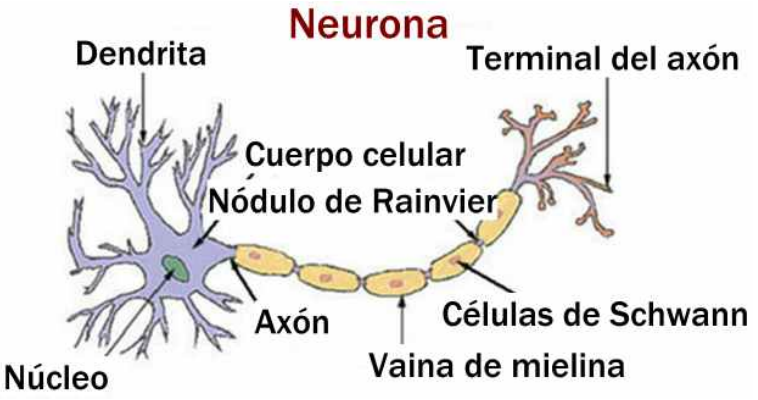


Figura 11.2: Neurona

La imagen 11.3 fue tomada con un microscopio de fluorescencia en la que se observan neuronas del hipocampo cultivadas in vitro.

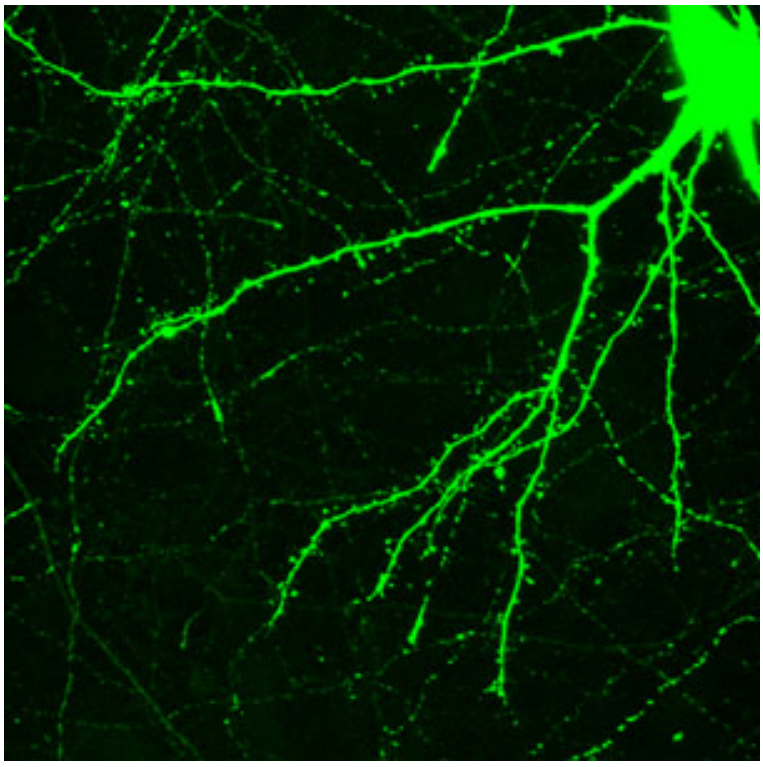
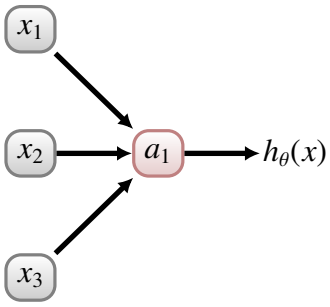
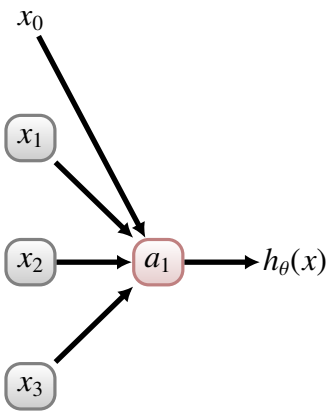


Figura 11.3: Dendritas

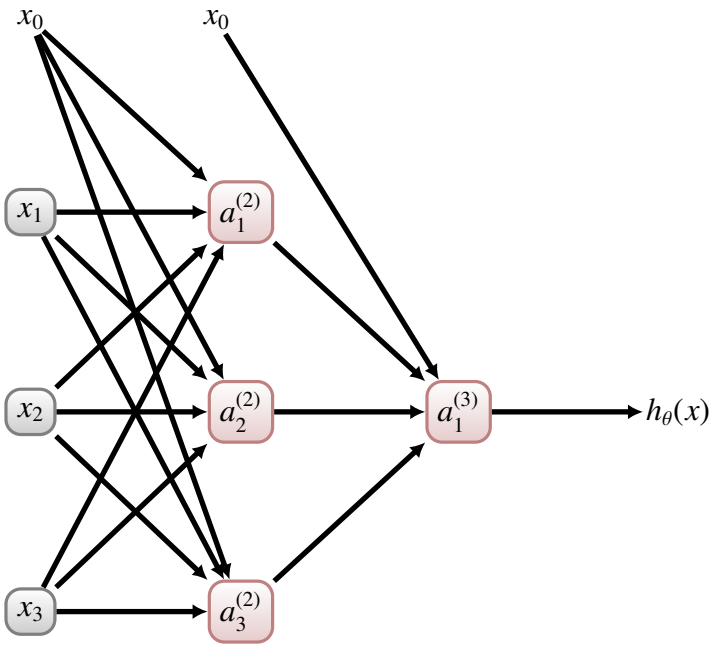
Ahora vemos la imitación a la neurona.



Observamos que una neurona la modelamos como un clasificador logistico,



Observamos que una red neuronal la modelamos como un arreglo de clasificadores logisticos.



Glosario:

$a_i^{(j)}$ = *Activación* de la neurona i en la capa j

$\Theta^{(j)}$ = Matriz de pesos que controla el mapeo de funciones de la capa j a la capa j+1

Neuronas:

$$a_1^2 = g(\Theta_{10}^{(1)} \cdot x_0 + \Theta_{11}^{(1)} \cdot x_1 + \Theta_{12}^{(1)} \cdot x_2 + \Theta_{13}^{(1)} \cdot x_3)$$

$$a_2^2 = g(\Theta_{20}^{(1)} \cdot x_0 + \Theta_{21}^{(1)} \cdot x_1 + \Theta_{22}^{(1)} \cdot x_2 + \Theta_{23}^{(1)} \cdot x_3)$$

$$a_3^2 = g(\Theta_{30}^{(1)} \cdot x_0 + \Theta_{31}^{(1)} \cdot x_1 + \Theta_{32}^{(1)} \cdot x_2 + \Theta_{33}^{(1)} \cdot x_3)$$

$$a_1^3 = g(\Theta_{10}^{(2)} \cdot a_0^{(2)} + \Theta_{11}^{(2)} \cdot a_1^{(2)} + \Theta_{12}^{(2)} \cdot a_2^{(2)} + \Theta_{13}^{(2)} \cdot a_3^{(2)})$$

11.3. Forward Propagation

Queremos crear un clasificador que pueda exitosamente clasificar la función XOR; nos damos cuenta rápidamente que no es posible con solo un clasificador lineal; por lo que necesitamos crear varios clasificadores lineales y combinar sus resultados para lograr obtener nuestro clasificador XOR.

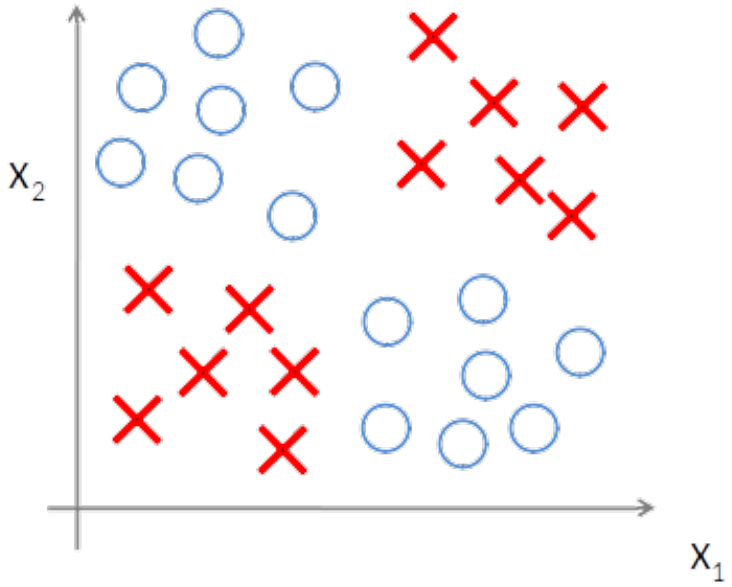
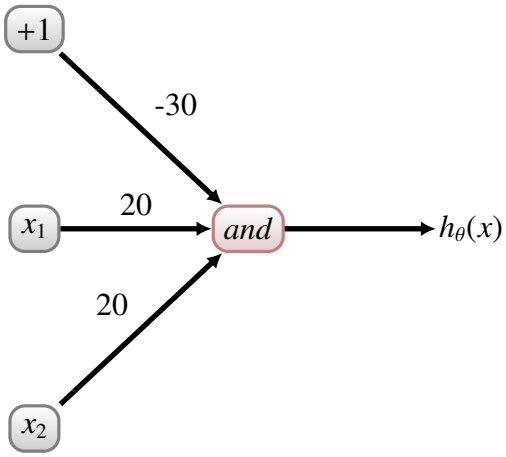


Figura 11.4: XNOR

x_1	x_2	A	B	C	x_1 XNOR x_2
		x_1 AND x_2	x_1 OR x_2	x_1 NOR x_2	A OR C
0	0	0	0	1	1
0	1	0	1	0	0
1	0	0	1	0	0
1	1	1	1	0	1

Tabla 11.1: Tabla de verdad de XOR

11.4. Función AND



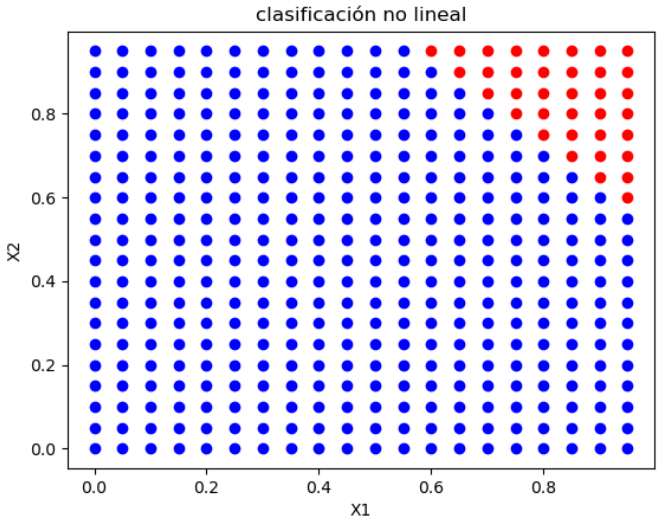
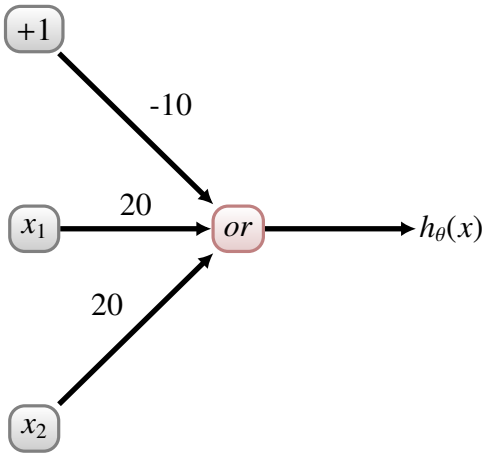


Figura 11.5: tabla AND

11.5. Función OR



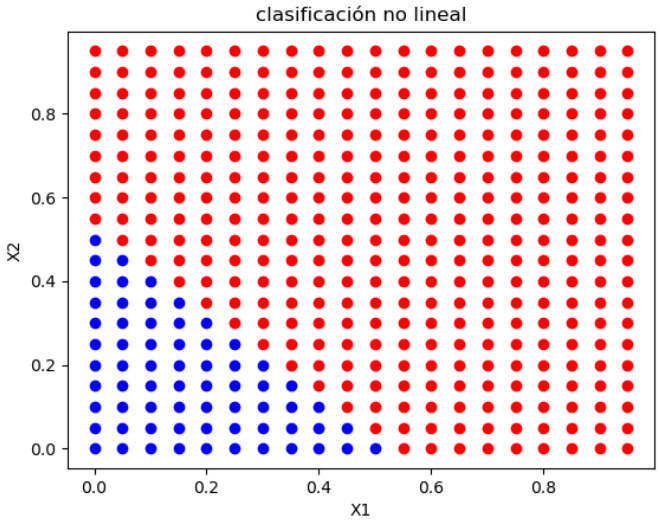
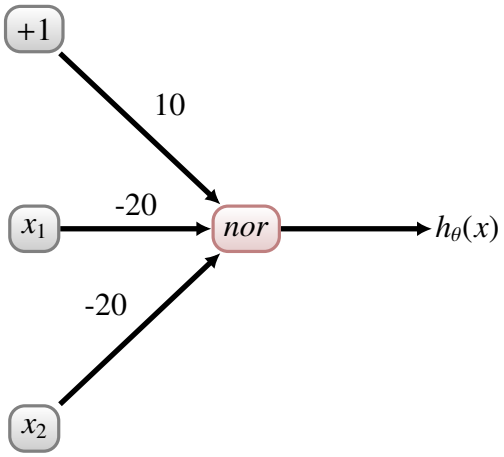


Figura 11.6: tabla OR

11.6. Función NOR



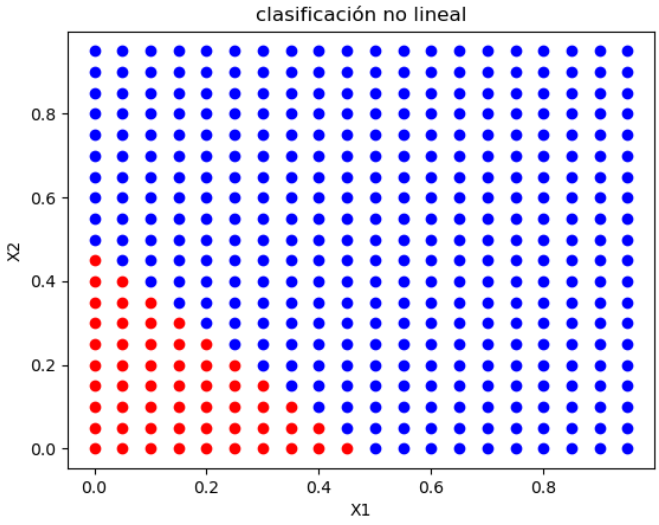
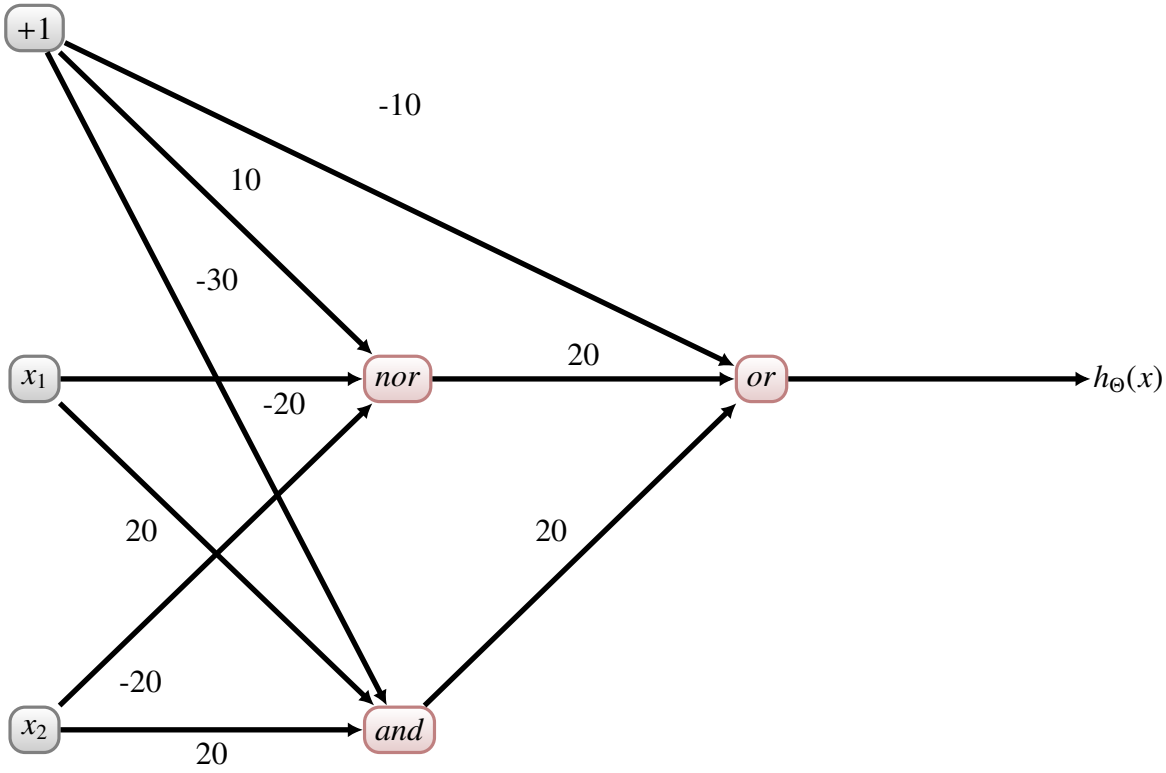


Figura 11.7: tabla NOR

11.7. Función XOR



Bibliografía

[1] Genesereth, M. R., and Nilsson, N. J. *Logical Foundations of Artificial Intelligence*, first edition ed. Morgan Kaufmann, July 1987.

[2] Grossman, S. I. *Algebra Lineal - 5b: Edicion*. McGraw-Hill Companies, July 1999.

[3] Nilsson, N. J. *Artificial Intelligence: A New Synthesis*, 1 ed. Morgan Kaufmann Publishers, Inc., Apr. 1998.

[4] Wikipedia contributors. Regresión lineal, Sept. 2012. Page Version ID: 59583240.