Supervisado: Regresión Lineal

Aprendizaje Automático

Víctor de la Cueva
vcueva@itesm.mx

Supervisado

- Aprendizaje inductivo (basado en ejemplos) donde cada uno de los ejemplos de entrenamiento está formado por un par ordenado:

  (entrada, salida)

- Basado en los ejemplo se trata de encontrar la función que más se ajusta a dichos ejemplos, es decir, la que provoca menos error.
- La salida deseada, que es la que se proporciona con el ejemplo, se pueden ver como una etiqueta del ejemplo.

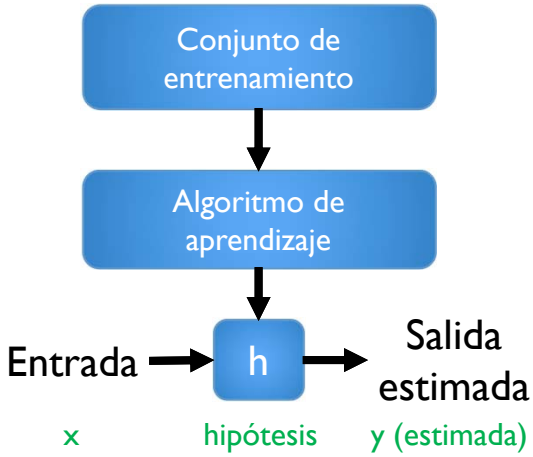
Objetivo del ML supervisado

- Las técnicas de ML supervisado se suelen agrupar en dos grandes grupos dependiendo del objetivo de la función que se quiere encontrar:
  - Regresión o pronóstico: Se encuentra una función de salida continua por medio de la cual se trata de predecir la salida para una entrada dada.
  - Clasificación: Se encuentra una función de salida discreta por medio de la cual, para una entrada dada, se dice a qué clase, de un conjunto finito de clases, pertenece.

Técnicas de ML supervisado

- Existe un gran número de técnicas para hacer ML supervisado, las que vamos a tratar son:
  - Regresión lineal
    - De una variable
    - Múltiple o multivariable
  - Regresión logística
  - Árboles de decisión
  - Redes neuronales

Gráficamente



¿Por qué regresión?

- Regresión viene de la palabra Regresar (*regresses to the mean*)
- La razón es que, cuando se iniciaron los estudios para hacer pronósticos, se pensaba que los datos siempre regresaban a la media.
- Ejemplo de estatura de personas y billar.
- Posteriormente, la gente utilizó esta idea para usar formas funcionales para aproximar un conjunto de datos, olvidando el concepto original de regresar a la media.
- La palabra no significa lo que hace ahora pero se quedó.

Supervisado

REGRESIÓN LINEAL DE UNA VARIABLE

Análisis de Regresión

- Es una técnica estadística para investigar y modelar la relación entre variables.
- Hay aplicaciones en muchos campos y la ingeniería no es la excepción.
- Posiblemente, el análisis de regresión sea la técnica estadística más usada.

Regresión lineal

- Dado un conjunto de puntos *(x,y)*, donde *x* es la entrada y *y* es la salida, encontrar la función *f* (donde *y = f(x)*) que mejor se ajusta a ellos.

NOTA: Existe una confusión al aplicar este método debido a que la función f puede no ser lineal con respecto a la variable x pero sí lo es con respecto a los parámetros que la definen (lo aclaramos más adelante al ver regresión lineal múltiple).

Ejemplo

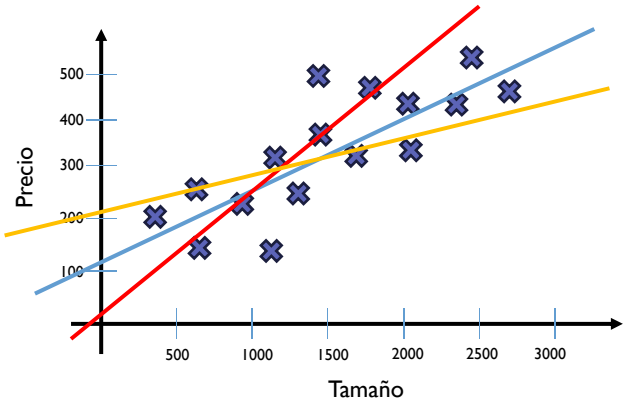
- Se quiere establecer la relación que existe entre el precio de venta de una casa y su tamaño.
- Supongamos que tenemos los siguientes datos:

X	Y
Tamaño (ft^2)	Precio (USD 1000's)
2104	460
1416	232
1534	315
852	178
…	…

Training Set

Gráfica de Dispersión

- Estos datos se pueden graficar en un plano:



¿Cuál es la **función** que mejor se ajusta a los puntos? (no necesariamente es una recta)

Función

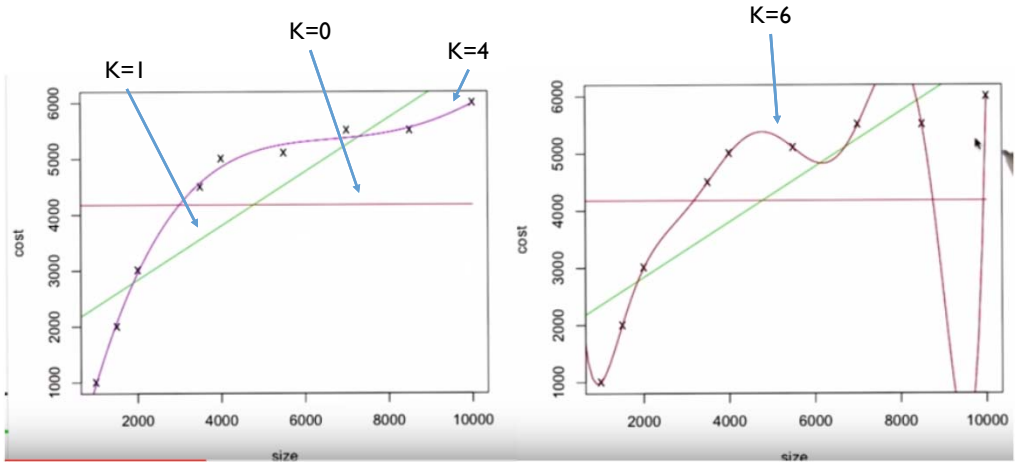
- Se desea aprender cuál es la función que mejor se aproxima (o ajusta) a los datos.
- El algoritmo del análisis de regresión requiere que se le de una posible función como entrada.
- Así que lo primero que debemos hacer es seleccionar una posible función.
  - Hay muchas familias muy utilizadas pero la más común es la polinomial
  - Debemos decidir el grado de función polinomial a usar
  - Lo más simple es usar una recta (recta de regresión)

Ejemplo: ¿qué función es mejor?

- Usando funciones polinomiales:
  $f(x) = c_0+c_1x+c_3x^2+\ldots+c_kx^k$
- ¿cuál grado es el mejor?:
  - k=0 (constante)
  - k=1 (recta)
  - k=2 (parábola), etc.
- El grado no puede pasar el número de puntos.

i	Tamaño	Costo
1	500	1000
2	1300	2000
3	2000	3000
4	3800	4500
5	4000	5000
6	5500	5100
7	6800	5300
8	7400	5400
9	10000	6000

Gráfica



Fuente: [6]

Usando una recta

- Algunas veces da la impresión que los datos caen, en general, pero no exactamente, en una línea recta.
- Si y representa el precio y x el tamaño, la ecuación de la recta que relaciona las dos variables sería:

$$y = \theta_0 + \theta_1 x$$

Donde θ_0, es la ordenada al origen y θ_1 es la pendiente.

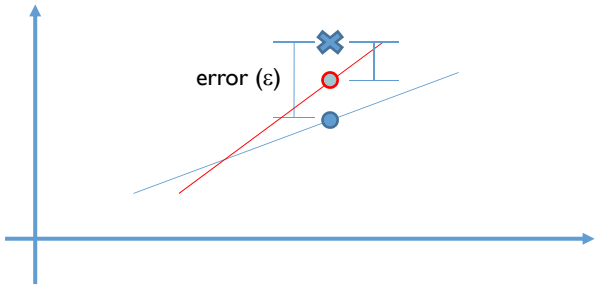
- Surgen varias preguntas:
  - ¿El mejor ajuste es una recta? (hipótesis)
  - ¿Cuáles son los valores de θ_0 y θ_1? (parámetros de la hipótesis)
  - Si hay varias opciones, ¿cómo las selecciono?
  - ¿Para qué me sirve este ajuste?

Regresión Lineal Simple (o de una variable)

- θ_0 y θ_1, son desconocidos y se llaman parámetros.
- Cuando el modelo usa una línea recta como ajuste y la variable de salida depende sólo de una variable de entrada, se le llama: modelo de regresión lineal simple.
- Desde luego que los parámetros pueden tomar un número muy gran de valores:
  - ¿Cómo los selecciono?
  - ¿Cuál es el criterio de selección?
- Aunque hay muchos criterios, por lo general, la forma más usada es la de minimización del error.

Error

- El error es la distancia entre el punto real y el punto dado por la recta.



- El problema es que se debe seleccionar la que minimice el error pero en todas las observaciones al mismo tiempo.
  - ¿Cómo incluyo todas?

Suma de los errores

- Seguro que se les está ocurriendo seleccionar aquella que minimice la suma de los errores.
- La función más común para esto es usar el método de mínimos cuadrados.
  - ¿Cómo se hace una función del error que dependa de todos los errores?
- Si la logramos encontrar, lo único que tenemos que hacer es minimizarla (es un problema de optimización) y listo.

Estimación de θ_0 y θ_1

- La forma más común para estimar θ_0 y θ_1 es usar mínimos cuadrados.
- Si definimos el error de un punto x_i como $\varepsilon_i = (y_i - d_i)$, donde y_i es la salida obtenida (por la recta) y d_i es la salida deseada (de acuerdo al ejemplo de entrenamiento).
- Podríamos definir un error total como la suma de estos errores, pero como algunos son + y otros -, se eliminarían.

Error total

- Otra opción es usar el valor absoluto y definirlo así: $E = \sum \varepsilon_i = \sum |y_i - d_i|$, el problema es que el valor absoluto no es derivable.
- Otra opción es usar las diferencias al cuadrado y es lo que le da el nombre al método (mínimos cuadrados):

$$E = \sum (y_i - d_i)^2$$

$E(\theta_0, \theta_1) = \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - d_i)^2$ (función de costo)

- Lo que se desea hacer es obtener el menor E y eso se logra modificando θ_0 y θ_1, es decir:

$$\min E(\theta_0, \theta_1)$$

Solución

- La minimización se logra DERIVANDO.

$$\frac{\partial E}{\partial \theta_0} = 2\sum_{i=1}^{m}(\theta_0 + \theta_1 x_i - d_i) = 0$$

$$\frac{\partial E}{\partial \theta_1} = 2\sum_{i=1}^{m}(\theta_0 + \theta_1 x_i - d_i)x_i = 0$$

$$m\theta_0 + \theta_1 \sum x_i - \sum d_i = 0$$

$$\theta_0 \sum x_i + \theta_1 \sum x_i^2 - \sum d_i\, x_i = 0$$

- Y su solución: DESDE LUEGO QUE EXISTE, y es un mínimo.

LMS

- En ML se acostumbra usar una variante de Mínimos Cuadrados llamada Error Cuadrático Medio (LMS, por sus siglas en inglés), en donde el error se define así:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- Donde $h_\theta(x) = \theta_0 + \theta_1 x$ (Hipótesis).
- A $J(\theta_0, \theta_1)$ se le conoce como Función de Costo, y es lo que se trata de minimizar:

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

Resumen

- Hipótesis: $h_\theta(x) = \theta_0 + \theta_1 x$
- Parámetros: θ_0, θ_1
- Función de Costo:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- Meta: $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Función de Costo

- Esta función de costo $J(\theta_0, \theta_1)$ es buena por varias razones pero la principal es que si se grafica E con respecto a θ_0 y θ_1, da un paraboloide, el cual tiene un solo mínimo.
- Además, es derivable.

Minimizando la función de costo

- Si procedemos igual que el Mínimos Cuadrados para obtener la solución sería:

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$= \begin{cases} \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) & para\ j = 0 \\ \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x^{(i)} & para\ j = 1 \end{cases}$$

Archivo de entrenamiento

- El archivo de entrenamiento tiene m ejemplos:

$(x^{(1)}, y^{(1)})$
$(x^{(2)}, y^{(2)})$
…
$(x^{(m)}, y^{(m)})$

- El cual tiene n *features* (variables), en este caso n = 1.

Usando matrices

- Podemos usar matrices para facilitar su manejo:

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \dots \\ x_n^{(i)} \end{bmatrix} \epsilon \mathbb{R}$$

$$\boldsymbol{X} = \begin{bmatrix} --- (\boldsymbol{x}^{(1)})^T --- \\ --- (\boldsymbol{x}^{(2)})^T --- \\ \dots \\ --- (\boldsymbol{x}^{(m)})^T --- \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

Facilitando el manejo de matrices

- Si se le agrega a cada ejemplo un 1 al inicio:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x^{(1)} \\ & \dots \\ 1 & x^{(m)} \end{bmatrix} \text{ y } \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

- Entonces podemos escribir: $h_\theta(\boldsymbol{X}) = \boldsymbol{X\theta}$ o $\boldsymbol{\theta}^T \boldsymbol{X}^T$ es decir, La función de costo es: $\boldsymbol{J} = (\boldsymbol{X\theta} - \boldsymbol{y})^2$
- Si derivamos, igualamos a **0** y despejamos θ, directamente obtenemos la ecuación normal para θ:

$$\boldsymbol{\theta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

- Sin embargo, no se utiliza este método por el cálculo de la inversa.
- El que se usa es un método iterativo llamado Gradiente Descendente.

Gradiente Descendente

- Repetir hasta convergencia
  - $\theta_j = \theta_j - \alpha \left[\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \right]$ para j=0 y j=1

Tasa de Aprendizaje (*learning rate*)

- La actualización de las θ debe hacerse en forma simultánea:

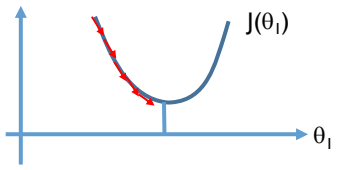
Correcto	Incorrecto
$temp0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ $temp1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ $\theta_0 = temp0$ $\theta_1 = temp1$	$temp0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$ $\theta_0 = temp0$ $temp1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$ $\theta_1 = temp1$

Idea de Gradiente Descendente

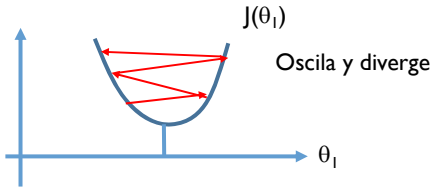
- La idea es ir disminuyendo el valor del error y eso se logra caminando en la superficie de la función de error, en la dirección donde disminuya dicho error E, la cual es indicada por el gradiente negativo de E.
- Gradiente Descendente sólo garantiza llegar a un óptimo local (*gready*) pero debido a la forma de E propuesta, garantizamos el óptimo global (sólo tiene 1).

Tasa de Aprendizaje

- La selección de α (indica el tamaño del paso) es vital para el buen funcionamiento del algoritmo, pero se hace a prueba y error:
- Si α es muy pequeño:



- Si α es muy grande:



Algoritmo Final

- Repetir hasta convergencia:
  - $\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$
  - $\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$
- Hay variantes del algoritmo dependiendo de cómo se toman los ejemplos:
  - Batch: Cada paso de gradiente descendente usa TODOS los ejemplos de entrenamiento.
  - On-Line: Cada paso toma un ejemplo.
  - Semi-Batch: Cada paso toma s ejemplos.

Supervisado

VECTORIZACIÓN

Vectorización

- Cuando el lenguaje de programación cuanta con operaciones con vectores directas, ya sea porque el lenguaje fue hecho para ello (e.g. Matlab) o porque existe una librería que se puede utilizar (e.g. Numpy para Python), es mejor hacer todas las operaciones que se pueden por medio de vectores (vectorización).
- La razón es que estas operaciones están optimizadas para tardarse menos que los ciclos con los que programamos nosotros.

Ejemplo

- Sea el vector X: el Y el θ

(agregando un 1)

Uno	Tamaño
1	500
1	1300
1	2000
1	3800
1	4000
1	5500
1	6800
1	7400
1	10000

Costo
1000
2000
3000
4500
5000
5100
5300
5400
6000

Theta
1848.24
0.5

Operaciones

- Para obtener las Ys aproximadas a partir de ese vector Theta, en lugar de hacer un par de FORs, se puede simplemente hacer: **X***θ
- A esto se le llama vectorización, la cual evita usar ciclos.

Supervisado

REGRESIÓN LINEAL CON POLINOMIO

Usando una NO recta

- En algunas ocasiones, la mejor forma de aproximar un conjunto de datos es por medio de una función no lineal de una sola variable.
- La familia de funciones no lineales más usada es un polinomio de grado mayor a 1.
- Por ejemplo, para el caso de grado 2 (una parábola), la función sería: $f(x) = \theta_0 + \theta_1 x + \theta_2 x^2$:
  - Es cuadrática con respecto a x pero lineal con respecto a los parámetros $(\theta_0, \theta_1, \theta_2)$.
- Este caso se puede ver como uno de regresión lineal múltiple (con múltiples variables, dos en este caso: x y x^2).

Supervisado

REGRESIÓN LINEAL MÚLTIPLE (MÚLTIPLES *FEATURES*)

Varias variables (*features*)

- A veces una variable no es suficiente para modelar el comportamiento.
- Ejemplo: precio de casa:

Tamaño (feet^2)	No. De cuartos	No. De pisos	Edad (años)	Precio ($1000)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

m

Notación

- n = número de features.
- $x^{(i)}$ = entrada (features) del i-ésimo ejemplo de entrenamiento.
- $x_j^{(i)}$ = valor de la feature j en el i-ésimo ejemplo de entrenamiento.
- Ejemplo:

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \qquad x_3^{(2)} = 2$$

Más notación

- Hipótesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

- Por conveniencia $x_0 = 1$ ($x_0^{(i)} = 1$)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

Gradiente descendente para variables múltiples

- Hipótesis: $h_\theta(x) = \theta^T X = \theta_0 x_0 \ldots + \theta_n x_n$
- Función de costo:

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

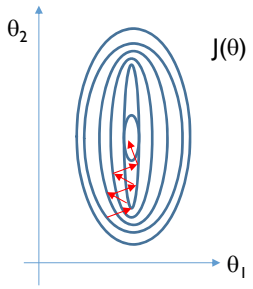
- Gradiente descendente:
  - Repetir
    - $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$
    - Con actualización simultánea para j = 0, …, n

Nuevo Algoritmo

- Repetir hasta convergencia
  - $\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$
- Esto es:
  - $\theta_0 = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_0^{(i)}$
  - $\theta_1 = \theta_1 - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_1^{(i)}$
  - …

Escalamiento

- Muchas veces, la diferencia en magnitud de una variable (feature) con respecto a otras es muy grande.
- Esto ocasiona que el gradiente descendente tarde mucho en una de las dimensiones.
- Ejemplo:
  - x_1 = tamaño (0-2000 ft^2)
  - x_2 = No. De cuartos (1-5)



  - Una variación en x_2 provoca mucho cambio en el costo J.

Escalas similares

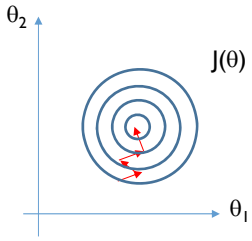
- Se debe procurar que todas las variables tengas escalas similares.
- Esto se logra haciendo escalamiento, por ejemplo, dividiendo entre el valor mayor de los datos:

$x_1 = \frac{tamaño(feet^2)}{2000}$ y $x_2 = \frac{No.de\ Cuartos}{5}$, $0 \leq x_1 \leq 1$ y $0 \leq x_2 \leq 1$

- Esto hace que llegue más rápido:
- Se recomienda llevar cada feature a

Un rango entre $-1 \leq x_i \leq 1$

Excepto x_0 que siempre es 1.



Normalización Media

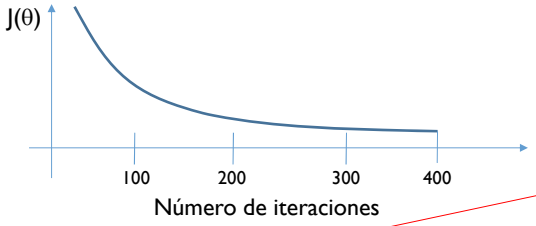
- La mejor forma de escalamiento es llevar todo a media cero (μ=0).
- Esto se logra restando la media a cada dato y dividiendo el resultado entre su rango:
  - $x_i = \frac{x_i - \mu}{rango}$, donde *rango = (máximo – mínimo) o rango =* σ
- Ejemplos:
  - $x_1 = \frac{tamaño\ -1000}{2000}$ y $x_2 = \frac{No.de\ cuartos - 2}{4\ (o\ 5)}$

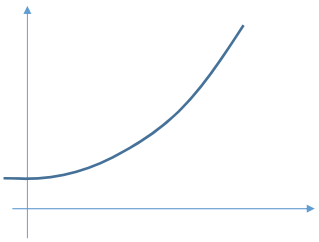
Tasa de Aprendizaje

- Debugging: ¿cómo estoy seguro que el GD está funcionando correctamente?
  - J(θ) debería decrecer en cada iteración.
  - Declarar convergencia si J(θ) decrece por lo menos 10^{-3} en cada iteración.
- ¿Cómo selecciono la tasa de aprendizaje α?
  - Para una α suficientemente pequeña, J(θ) debería decrecer en cada iteración, pero si es muy pequeña, GD es muy lento.
  - Si α es muy grande, J(θ) puede oscilar o crecer.

Gráficamente

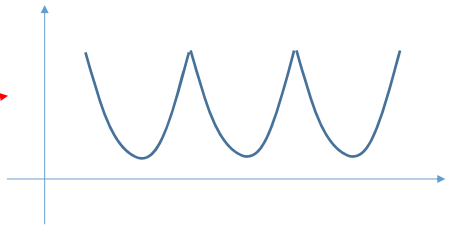
- Normal





- Creciente

- Oscilante



Recomendación para α

- Probar aumentos de 3x en 3x
- Por ejemplo:

0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1…

Supervisado

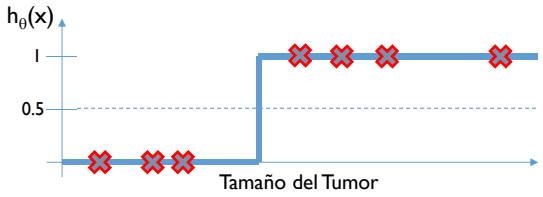
REGRESIÓN LOGÍSTICA

Clasificación

- Consiste en separar un conjunto de datos en 2 grupos, los que cumplen una condición y los que no.
- Es otro uso que se le da a la regresión.
- Ejemplos:
  - Email: Spam/No Spam
  - Online: Transacciones - Fraudulentas/Correctas
  - Tumor: Maligno/Benigno
- Por esta razón, la salida de los ejemplos de entrenamiento siempre es {1,0}
  - $y \in \{1,0\}$, 0: clase "negativa" y 1: clase "positiva".

Límite de clasificación

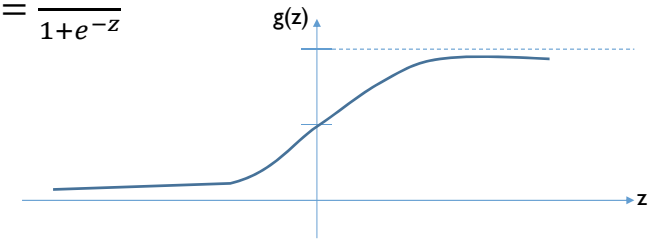
- El límite (threshold) de clasificación será entonces 0.5:
  - Si $h_\theta(x) \geq 0.5$, predice "y=1"
  - Si $h_\theta(x) < 0.5$, predice "y=0"
- La función ideal sería una Escalón.



- En este caso se usa la Regresión Logística (*Logistic Regression*), en la cual la hipótesis siempre debe dar valores entre 0 y 1:
  - $0 \leq h_\theta(x) \leq 1$.

Sigmoidal

- Queremos que $0 \leq h_\theta(x) \leq 1$
- Esto se logra de varias formas pero una de las más comunes, por la forma adecuada que tiene, ya que se parece a la escalón, es la Sigmoidal (*sigmoid function*).
  - $g(z) = \frac{1}{1+e^{-z}}$



Hipótesis

- Si hacemos:
  - $h_\theta(x) = g(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$
  - Lograríamos lo que se buscaba.
- La $h_\theta(x)$ se podría interpretar entonces como la probabilidad de que y = 1:
  - $h_\theta(x) = p(y = 1|x; \theta)$, probabilidad de que y=1 dado X y parametrizada por θ.

Función de costo

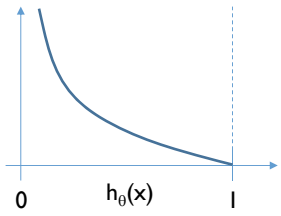
- Como en todos los métodos de regresión, se requiere una función de costo para poder encontrar el valor óptimo de θ que minimice el costo (e.g. usando gradiente descendente).
- Una buena función de costo sería aquella que diera un valor alto si la salida se acerca a 0 y y=1, y un valor bajo si la salida se acerca a 1 y y=1. Además, un valor alto si la salida se acerca a 1 y y=0 y un valor bajo se la salida se acerca a 0 y y=0.

Función Logaritmo

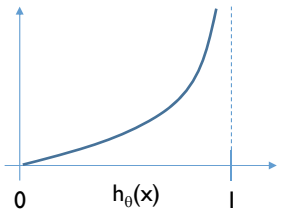
- La opción más usada es la función logaritmo en una función seccionada:

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

Si y=1



Si y=0



Función de costo simplificada

- Debido a que y siempre es {0,1}, se puede usar como una variable binaria para simplificar la función en una sola expresión:

$$Costo(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x))$$

- Ahora bien, el costo para todos los ejemplos es:

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

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

Optimización y Predicción

- Entonces, para encontrar los parámetros θ:

$$\min_{\theta} J(\theta)$$

- Para dar una predicción (salida) dada una nueva x:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T X}}$$

- Que es equivalente a la $p(y = 1|x; \theta)$

Gradiente Descendente

Repetir{

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Actualización simultánea de todas las θ_j

}

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Resulta que la derivada es igual que en el caso anterior pero ahora cambia la h a $h_\theta(x) = \frac{1}{1+e^{\theta^T X}}$, y en el anterior era $h_\theta(x) = \theta^T X$.

Optimización

- Existen muchos algoritmos para hacer la optimización de la función de costo:
  - Gradiente Descendente
  - Gradiente Conjugado
  - BFGS
  - L-BFGS

  Ventajas:
  - No selección de α a mano
  - Normalmente, más rápidos que GD

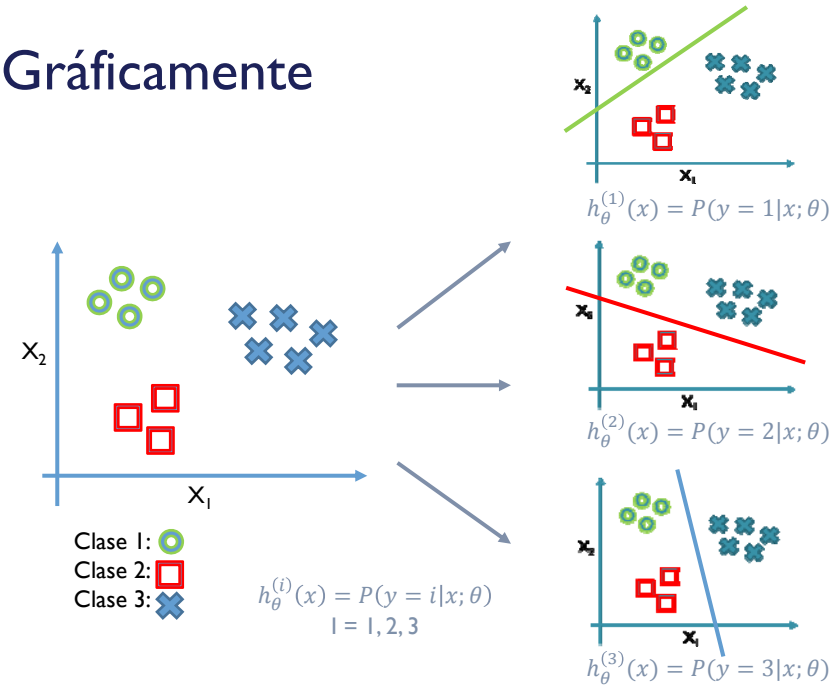
  Desventajas:
  - Más complicados

  Y algunos heurísticos:
  - Algoritmos Genéticos
  - Recocido Simulado, etc.
- Normalmente, existen librerías que contiene estos algoritmos ya programados en una forma muy eficiente.

Clasificación Multiclase

- En algunas ocasiones es necesario clasificar en una de varias clases:
  - Email foldering / tagging:
    - Work (y=1)
    - Friends (y=2)
    - Family (y=3)
    - Hobby (y=4)
  - Diagnóstico Médico:
    - Sano (y=1)
    - Gripa (*flu*) (y=2)
    - Resfriado (*cold*) (y=3)

Gráficamente



One vs All

- Entrenar un clasificador de regresión logística $h_\theta^{(i)}(x)$ para cada clase i, para predecir la probabilidad de que y=i.
- Para hacer la predicción de una nueva entrada X, seleccione la clase i que maximiza:

$$\max_i h_\theta^{(i)}(x)$$

Supervisado

REGULARIZACIÓN

Algunos puntos importantes al hacer ML

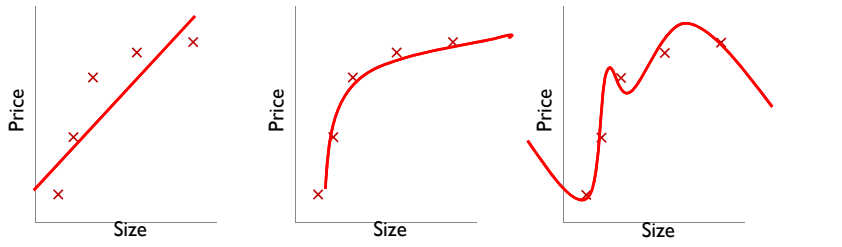
- Los métodos de ML han dado buenos resultados pero no son infalibles.
- Un bajo error en el entrenamiento no garantiza un excelente desempeño en el modo de producción.
- Muchos de los métodos de ML dependen de la correcta selección de sus parámetros (número y valor).

Corrección de bajo desempeño

Es muy común que el bajo desempeño se de por 2 razones:

- Underfitting.
  - Corrección: Aumentar características.
- Overfitting.
  - Corrección 1: Reducir el número de características
    - A mano.
    - Por selección de modelo (más adelante).
  - Correción 2: Mantener todas las características pero reducir la magnitud/valor de los parámetros θ_j (**Regularización**).

Ejemplo: Regresión Lineal



$\theta_0 + \theta_1 x$ Underfit

$\theta_0 + \theta_1 x + \theta_2 x^2$ Correcto

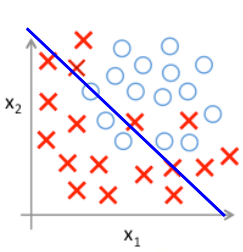
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ Overfit

Overfitting: Generalmente se produce cuando tenemos muchas características.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$$

Tomado de ML Course Porf. Ng.

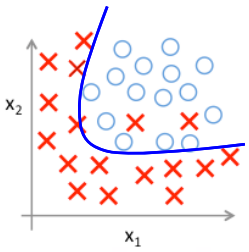
Ejemplo: Regresión Logística



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

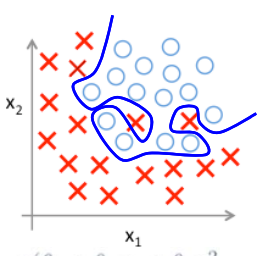
(g = sigmoid function)

Underfit



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

Correcto



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

Overfit

Tomado de ML Course Porf. Ng.

Regularización

- Se le agrega a la función de costo J un término de regularización controlado por un parámetro de regularización definido por nosotros:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

Parámetro de regularización

- Se puede hacer tanto en regresión lineal como en logística.

NOTA: No se regulariza θ_0 porque no tiene que ver con las características.

Valor de λ

- ¿Qué valor le damos a λ?
- Valor muy grande, e.g. $\lambda=10^{10}$, es como si quitáramos todas las θs, excepto θ_0.
- Valor muy pequeño, no quita el overfitting.
- No hay otra más que hacerlo a prueba y error (existen formas sistemáticas como la que vimos para α).
- Lo que sí es un hecho es que si se modifica J se modifica su gradiente.

Gradiente Descendente con regularización

Repetir {

$$\theta_0 = \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_0^{(i)} + \overbrace{\frac{\partial}{\partial\theta_0}\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}^{\text{Es 0}}$$

$$\theta_j = \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} - \frac{\lambda}{m}\theta_j\right] \quad j = 1, \dots, n$$

}

La θ_j se puede escribir como:

$$\theta_j = \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \qquad 1 - \alpha\frac{\lambda}{m} < 1$$

Regresión Logística Regularizada

- Función de costo:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

- Gradiente Descendente:

Repetir {

$$\theta_0 = \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j = \theta_j - \alpha\underbrace{\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} - \frac{\lambda}{m}\theta_j\right]}_{\frac{\partial}{\partial\theta_j}J(\theta)} \quad j = 1, \dots, n$$

}

$$h_\theta(x) = \frac{1}{1 + e^{\theta^T X}}$$

Referencias

1. T. Mitchell. Machine Learning. McGraw-Hill (1997).
2. Andrew Ng. Machine Learning. Stanford-Coursera (2011)
   - https://www.coursera.org/course/ml
3. S. Russel and P. Norvig. Inteligencia Artificial un enfoque moderno. 2ª edición, Pearson, España (2004).
4. B. Sierra. Aprendizaje Automático conceptos básicos y avanzados. Pearson (2006).
5. Wikipedia. http://en.wikipedia.org/wiki/Machine_learning
6. Michael Littman and Charles Isbell. Machine Learning. Georgia Tech and Coursera (2017)
   - https://www.udacity.com/course/machine-learning--ud262