Universidad Nacional Autónoma de México

Facultad de Ingeniería

Programa de Tecnología en Cómputo

Manual de apoyo para el curso de Machine Learning

División de Ingeniería Eléctrica

Rivera Negrete Manuel Armando

Trabajo de investigación:
Sección de interfaces electrónicas

12 de agosto de 2019

Índice general

Índice de figuras

Índice de tablas

Acerca de este manual

Este manual fue hecho para guiar a las personas hispano hablantes que quieren introducirse al mundo del Machine Learning, este manual es una recopilación del material escrito y en video del curso disponible en la plataforma Coursea por la Universidad de Stanford impartido por Andrew Ng a quien se le da el crédito por el material proporcionado sin el cual este proyecto no hubiera sido posible.

Capítulo 1

Introducción

El aprendizaje automático es una de las nuevas tecnologías más emocionantes. Cada vez que usas un buscador web como Google o Bing para hacer una búsqueda en internet, una de las razones por las que funciona tan bien es porque un algoritmo de aprendizaje, implementado por Google o Microsoft, ya aprendió a clasificar las páginas web. Cada vez que usas Facebook o o que la aplicación de fotos de Apple reconoce las fotos de tus amigos, eso es aprendizaje automático.

Cada vez que lees tu correo electrónico o que tu filtro de spam te salva de tener toneladas de correo basura, eso también es un algoritmo de aprendizaje.
¿Por qué predomina el aprendizaje automático hoy en día? Resulta que el aprendizaje automático es un campo que nació a partir de la IA o inteligencia artificial. Queríamos construir máquinas inteligentes y resulta que hay pocas cosas básicas que podemos programar para que haga una máquina como encontrar la ruta más corta de A a B. Pero en gran medida no sabíamos cómo hacer programas de IA para hacer cosas más interesantes como buscar en la web, etiquetar fotos o antispam. Se comprendió que la única forma de hacer estas cosas era que una máquina aprendiera a hacerlas por sí misma. Así que el aprendizaje automático se desarrolló como una nueva capacidad para las computadoras y hoy está en muchos sectores de industria y ciencia básica.

Veamos otros ejemplos de aprendizaje automático. Existe la minería de datos. Una de las razones por las que el aprendizaje automático se ha difundido es el crecimiento de la web y de la automatización, es decir, que ahora tenemos más datos que nunca. Por ejemplo, hoy en día muchas empresas en Silicon Valley recolectan datos de clics, también llamados datos de tráfico, y tratan de usar algoritmos de aprendizaje electrónico para explorar estos datos y entender mejor a los usuarios y darles mejor servicio. Actualmente es un gran segmento de Silicon Valley. Expedientes médicos. Ahora con la llegada de la automatización, tenemos expedientes médicos electrónicos, entonces si convertimos los expedientes médicos en conocimientos médicos, empezaremos a entender mejor la enfermedad. Biología informática. También con la automatización los biólogos recolectan muchos datos sobre la secuencia genética, secuencias de ADN, etc. y los algoritmos de aprendizaje automático nos ayudan a entender mejor el genoma humano y qué significa ser humano. También en ingeniería, en todos las áreas de ingeniería, tenemos conjuntos de datos cada vez más y más grandes, que tratamos de entender mediante algoritmos de aprendizaje.

Actualmente, la mayoría del procesamiento de lenguaje natural y de visión por computadora aplica aprendizaje automático. Los algoritmos de aprendizaje se utilizan mucho en los programas de autopersonalización. Cada vez que entras a Amazon, Netflix o iTunes Genius y te recomienda películas, productos y música, eso también es un algoritmo de aprendizaje. Si lo piensas, tienen millones de usuarios; no hay forma de crear un millón de programas distintos para esos usuarios. La única manera de que el software genere estas recomendaciones personalizadas es que pueda aprender por sí mismo tus preferencias. Por último, los algoritmos de aprendizaje se usan hoy en día para entender el aprendizaje humano y para entender el cerebro.
Hace algunos meses, un estudiante me mostró un artículo.

1.1. Qué es Machine Learning

Entre los practicantes del aprendizaje automático no existe una definición bien aceptada sobre lo que es y lo que no es aprendizaje automático.
Pero vamos a ver un par de ejemplos sobre las maneras en que la gente ha intentado definirlo. Esta es la definición sobre qué es aprendizaje automático para **Arthur Samuel**. Definió el aprendizaje automático como *el campo de estudio que le da a los ordenadores la habilidad de aprender algo sobre lo que no han sido explícitamente programados.*

La fama de Samuel se remonta a la década de 1950, cuando escribió un programa para jugar a las damas. Y lo asombroso sobre este programa que juega a las damas era que el propio Arthur Samuel no era un buen jugador de damas. Pero lo que él hizo fue, tener un programa para jugar decenas de miles de partidas contra si mismo. Y observando que tipo de posiciones en el tablero tienden a conducir a la victoria y que tipo de posiciones en el tablero tienden a perder.
El programa de juego de damas aprende con el tiempo que posiciones son buenas en el tablero y cuales son malas posiciones. Y, finalmente, aprender a jugar a las damas mejor de lo que el propio Arthur Samuel era capaz. Esto fue un resultado notable. Aunque el propio Samuel resultó no ser un jugador de damas muy bueno. Pero debido a que el ordenador tiene la paciencia para jugar decenas de miles de partidas contra él mismo. Ningún humano tiene la paciencia para jugar esa cantidad de partidas. De esta manera el ordenador fue capaz de obtener una gran experiencia jugando a las damas que, finalmente llegó a ser mejor jugador que el propio Arthur Samuel.

Esta es una definición un tanto informal, y algo más antigua. Aquí está una definición un poco más reciente de **Tom Mitchell**, Entonces Tom define el aprendiza automático diciendo que, *Un programa de ordenador se dice que aprende de la experiencia **E**, con respecto a **T**, y alguna medida de rendimiento **P**. Y si esta actuación en **T**, medida por **P** mejora la experiencia **E**.*

Para el ejemplo de jugadores de damas, la experiencia **E**, será la experiencia de tener el programa jugando decenas de miles de partidas reiteradas contra él mismo. La tarea **T**, será la tarea de jugar partidas. Y la medida de mejora **P**, será la probabilidad que lo haga ganar la siguiente partida de damas contra un nuevo oponente.

1.2. Aprendizaje Supervisado

Supongamos que quieres predecir precios de casas. Y digamos que trazas un conjunto de datos y se ve así. En la figura 1.1 tenemos en el eje horizontal, el tamaño de distintas casas en pies cuadrados, y en el eje vertical, el precio de distintas casas en miles de dólares. Entonces, con estos datos, digamos que tienes un amigo que tiene una casa de 750 pies cuadrados y desea venderla y quiere saber cuánto dinero puede obtener por ella. Entonces, ¿cómo puede ayudarte un algoritmo de aprendizaje?

Algo que podría hacer el algoritmo de aprendizaje es trazar una línea recta a través de los datos o insertar una línea recta en los datos y, con base en eso, resulta que tal vez la casa pueda venderse en aproximadamente $150,000. Pero tal vez este no es el único algoritmo de aprendizaje que puedes usar. Puede haber uno mejor. Por ejemplo, en lugar de trazar una línea recta en los datos, podríamos decidir que es mejor insertar una función cuadrática o un polinomio de segundo grado a estos datos. Y si haces eso y haces una predicción aquí, entonces resulta que tal vez puede vender la casa en casi $200,000.

el término aprendizaje supervisado se refiere al hecho de que le dimos al algoritmo un conjunto de datos donde se daban las "respuestas correctas". Es decir, le dimos un conjunto de datos de casas en los que para cada ejemplo del conjunto de datos, se dijo cuál era el precio correcto, el precio real al que se vendió esa casa y la tarea del algoritmo sólo fue generar más "respuestas correctasçomo para esta casa nueva que tu amigo está queriendo vender. Para definir un poco más la terminología, esto también se denomina un problema de regresión y con esto quiero decir que

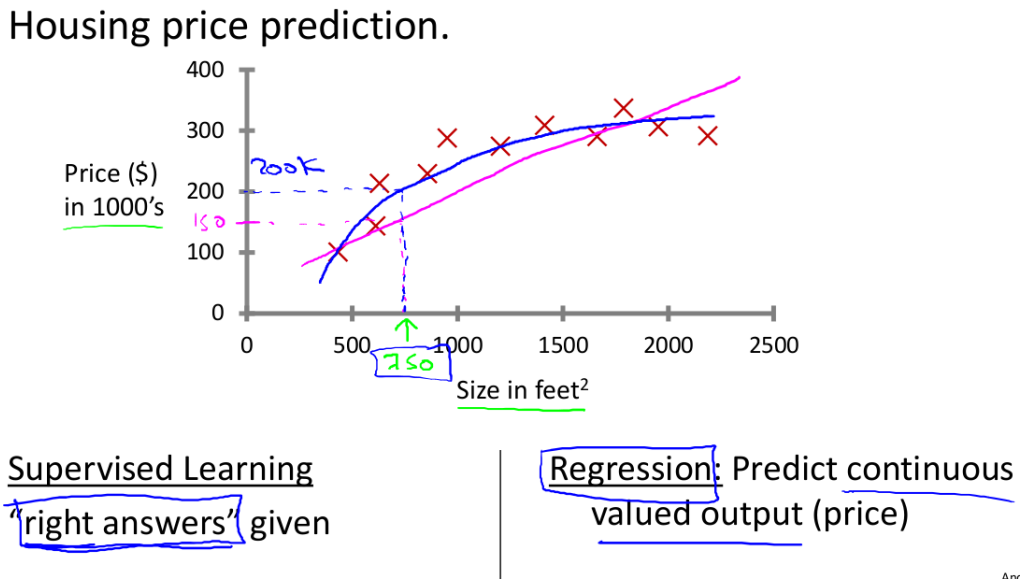


Figura 1.1: Ejemplo de aprendizaje supervisado

queremos predecir un resultado de valor continuo. Es decir, el precio. Así que, técnicamente, creo que puede redondearse el precio al centavo más cercano. Tal vez los precios en realidad son valores discretos, pero en general pensamos en el precio de una casa como un número real, un valor escalar, un número de valor continuo, y el término regresión se refiere al hecho de que tratamos de predecir el tipo de atributo valuado de forma continua.

Digamos que quieres revisar expediente médicos y tratar de predecir cáncer de mama como maligno o benigno.
Un tumor maligno es un tumor dañino y peligroso y un tumor benigno es un tumor inofensivo. Veamos en la Figura 1.2 un conjunto de datos recolectados y supongamos que en estos datos, en el eje horizontal tienes el tamaño del tumor y en el vertical, sí o no, si o no son ejemplos de tumores, el uno si es maligno o el cero si no es maligno.

Supongamos que nuestro conjunto de datos se ve así donde vemos algunos tumores de varios tamaños que resultaron ser benignos.
Por desgracia, también vemos unos tumores malignos. Así que en este ejemplo tengo cinco ejemplos de tumores benignos que se muestran abajo y cinco de tumores malignos que se muestran con un valor del eje vertical de uno.
Y digamos que tenemos una amiga que lamentablemente tiene un tumor mamario y supongamos que su tamaño es de aproximadamente del valor mostrado con la flecha magenta en la figura 1.2. La pregunta de aprendizaje automático es: ¿puedes calcular cuál es la probabilidad, de que un tumor sea maligno versus benigno?

Para mostrar un poco de terminología, este es un ejemplo de un problema de clasificación. El término "clasificación"se refiere al hecho de que intentamos predecir un resultado de valor discreto: cero o uno, maligno o benigno. Y resulta que algunas veces en problemas de clasificación, puedes tener más de dos valores para dos posibles valores para el resultado.
Como ejemplo concreto tal vez haya tres tipos de cáncer de mama, por lo que puedes intentar predecir el valor discreto de cero, uno, dos o tres, en el que cero es benigno. Tumor benigno, es decir sin cáncer. Y uno puede significar un tipo de cáncer, si tienes tres tipos de cáncer, lo que sea que signifique el tipo uno. Y dos puede significar un segundo tipo de cáncer y tres pude ser un tercer tipo de cáncer. Pero esto también sería un problema de clasificación, debido al otro conjunto

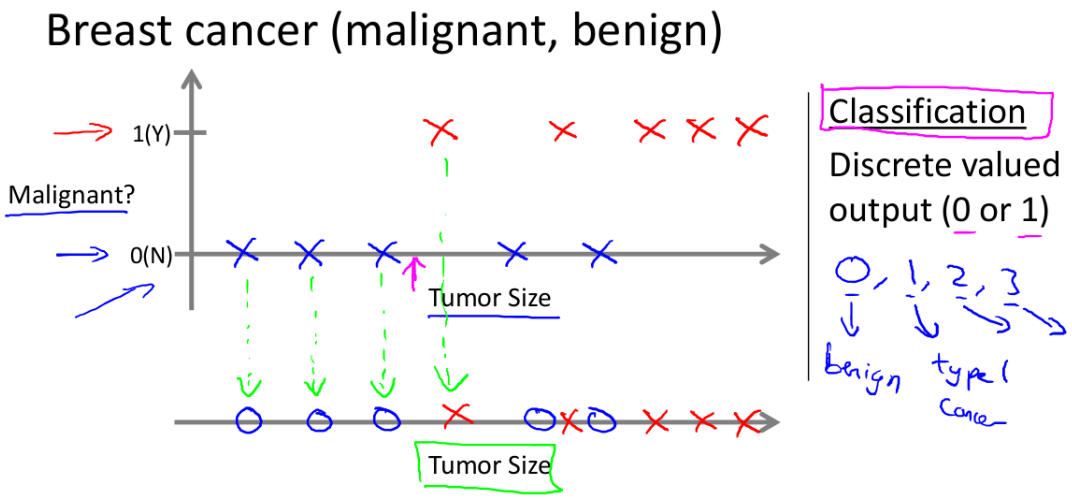


Figura 1.2: Ejemplo de clasificación

de resultados del valor discreto que corresponde a: sin cáncer o cáncer tipo uno, tipo dos, o cáncer tipo tres.

Existe otra forma de trazar estos datos en los problemas de clasificación. Así que si el tamaño del tumor será el atributo que se usará para predecir si es maligno o benigno, En lugar de dibujar cruces,se dibujarán círculos para los tumores benignos. Y se usarán X para denotar los tumores malignos. Todo lo se hizo fue tomar el conjunto de datos de arriba en la Figura 1.2 y se dibujaron abajo.

En este ejemplo sólo usamos una característica o un atributo, principalmente, el tamaño del tumor para predecir si el tumor es maligno o es benigno.

Digamos que en lugar de sólo saber el tamaño del tumor, también conocemos la edad y el tamaño de tumor de los pacientes. En ese caso, tal vez tu conjunto de datos se verá como en la figura 1.3 y tenemos un conjunto de pacientes cuyos tumores resultaron malignos, como lo indican las cruces.

Así que digamos que tienes un amigo que por desgracia tiene un tumor. Y tal vez, el tamaño del tumor y la edad están más o menos donde indica el punto color magenta. Con un conjunto de datos como este, lo que puede hacer el algoritmo de aprendizaje es trazar la línea recta en los datos para tratar de separar los tumores malignos de los benignos y entonces el algoritmo de aprendizaje puede decidir trazar la línea recta para separar las dos clases de tumores. Y con esto, es posible que puedas decidir que es más probable.

En este ejemplo, tuvimos dos variables, la edad del paciente y el tamaño del tumor. En otros problemas de aprendizaje automático con frecuencia tendremos más características como el grosor de acúmulo del tumor mamario, uniformidad del tamaño celular del tumor, uniformidad de la forma celular del tumor, etc., así como otras características. Resulta que para algunos problemas de aprendizaje, lo que realmente no quieres es usar tres o cinco características, sino usar un número infinito de características, un número infinito de atributos, de manera que tu algoritmo de aprendizaje tenga muchos atributos o características o señales para hacer las predicciones. Entonces, ¿cómo manejas un número infinito de características? ¿Cómo almacenas un número infinito de cosas en la computadora cuando sabes que se va a llenar la memoria? Resulta que cuando nos referimos a un algoritmo llamado Máquina de Soporte Vectorial, habrá un truco matemático genial que permitirá

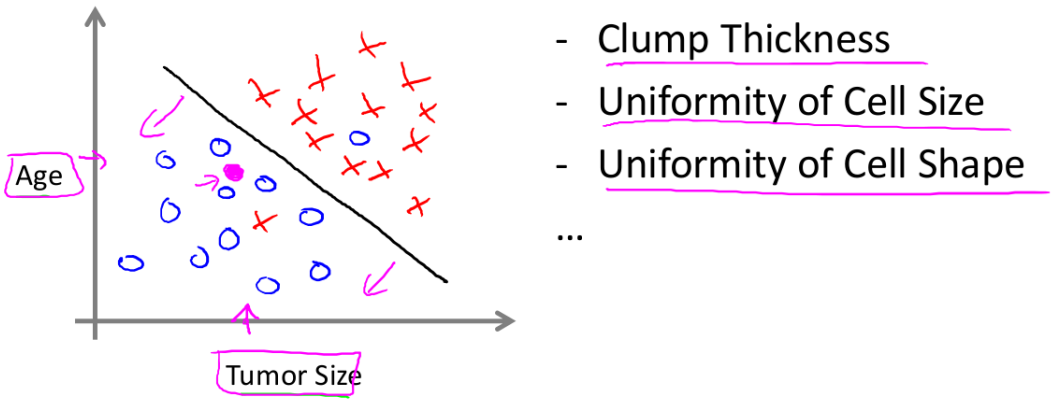


Figura 1.3: Problema de clasificación con dos variables

que la computadora maneje un número infinito de características.

En el aprendizaje supervisado, cada ejemplo de nuestro conjunto de datos, nos dice que la "respuesta correcta"que nos habría gustado, la predijeron los algoritmos en ese ejemplo. Como el precio de la casa o si el tumor es maligno o benigno. También hablamos sobre el problema de regresión. Y regresión significa que nuestro objetivo es predecir un resultado de valor continuo. Y hablamos sobre el problema de clasificación, donde el objetivo es predecir el resultado de valor discreto. Solo una pregunta para recapitular: Supongamos que diriges una empresa y quieres desarrollar algoritmos de aprendizaje para enfrentar los dos problemas. En el primer problema, tienes un gran inventario de artículos idénticos. Entonces, imagina que tienes miles de copias de algunos artículos idénticos para vender y quieres saber cuántos de estos artículos venderás en los próximos tres meses. En el segundo problema, el problema dos, tendrás muchos usuarios y querrás diseñar un software para analizar cada una de las cuentas individuales de tus clientes, cada una de las cuentas de tus clientes; y para cada cuenta decidirás si la cuenta fue o no hackeada o estuvo en peligro. Entonces, ¿cada uno de estos problemas debe tratarse como un problema de clasificación o como un problema de regresión?

Por problema uno lo trataría como un problema de regresión, ya que si tuviera miles de artículos, bueno, probablemente lo trataría como un valor real, como un valor continuo. Y por lo tanto, trataría el número de artículos que vendo como un valor continuo. Y para el segundo problema, lo trataría como un problema de clasificación, ya que podría establecer el valor que quiero predecir con cero para denotar que no se hackeo la cuenta. Y establecer el valor en uno para denotar una cuenta que sí fue hackeada. Así como para el cáncer de mama cero es benigno y uno es maligno. Por ello podría establecerlo en cero o uno dependiendo de si se hackeó y hacer que un algoritmo trate de predecir cada uno de estos dos valores discretos. Y debido a que existe un número reducido de valores discretos, entonces lo manejaría como un problema de clasificación.

1.3. Aprendizaje no supervisado

En el aprendizaje no supervisado nos dan el conjunto de datos y no nos dicen qué hacer con eso y no nos dicen cuál es cada punto de datos. En lugar de eso, nos dicen aquí está el conjunto de datos. ¿Puedes encontrar alguna estructura en los datos? Con estos datos, un algoritmo de aprendizaje no supervisado podría decidir que lo datos están en dos grupos diferentes. Así que hay un grupo y un grupo diferente. A esto se le llama algoritmo de agrupamiento.

El aprendizaje no supervisado nos sirve para hacer el agrupamiento de datos o clustering, también nos sirve para la reducción de dimensionalidad, por ejemplo si tenemos muchos datos y queremos trabajar únicamente con un sector. Cluster: Se puede trabajar con segmentación de mercado. Sistemas de recomendación. Reducir dimensionalidad: Visualización Big data. Descubrimiento de estructuras. Algoritmos: k-means: k es el numero de grupos, cada grupo tendrá un centro (Centroide) y alrededor de este centro se ubicaran datos con caracterísicas similares.

El aprendizaje no supervisado nos permite abordar problemas con poca o ninguna idea de cómo deberían ser nuestros resultados. Podemos derivar la estructura de datos donde no necesariamente sabemos el efecto de las variables.

Podemos derivar esta estructura agrupando los datos en función de las relaciones entre las variables en los datos.

Con el aprendizaje no supervisado no hay retroalimentación basada en los resultados de la predicción.

Ejemplo:

Agrupación: tome una colección de 1,000,000 de genes diferentes y encuentre una manera de agrupar automáticamente estos genes en grupos que de alguna manera sean similares o estén relacionados por diferentes variables, como la duración de la vida, la ubicación, los roles, etc.

Sin agrupación: el “algoritmo de cóctel”, le permite encontrar la estructura en un entorno caótico. (es decir, identificar voces individuales y música de una malla de sonidos en un cóctel).

Capítulo 2

Regresión Lineal con una variable

La regresión lineal predice una salida de valor real basada en un valor de entrada. Discutimos la aplicación de la regresión lineal a la predicción del precio de la vivienda, presentamos la noción de una función de costo e introducimos el método de pendiente de gradiente para el aprendizaje.

2.1. Modelo y función de costo

2.1.1. Modelo y representación

En la figura 2.1 tenemos datos de los precios de distintas casas con sus repectivas áreas. Supongamos que queremos obtener un dato que no está en la gráfica, por ejemplo, queremos comprar una casa de 1250 ($feet^2$) con una aproximación sabremos que nos costará \$220k. La regresión lineal nos permite predecir un valor a partir de otros. La notación que usaremos será la siguiente.

- **m**: Número de datos de entrenamiento.
- $x^{(i)}$: Variables de entrada.
- **y**: Variables de salida

En nuestro ejemplo de las casas nuestras variables de entrada (**x**) serían el área de las casas. Las variables de salida son los precios de dichas casas. Guiándonos en la tabla 2.1 diremos que $X^{(1)} = 2104$ Cabe destacar que el 1 no es un exponente sino un índice.
Para obtener la función que mas se adapte a nuestros datos de entrenamiento tenemos que usar la función de hipótesis que se muestra en la ecuación 2.1 donde θ_0 es el punto sobre el eje y donde

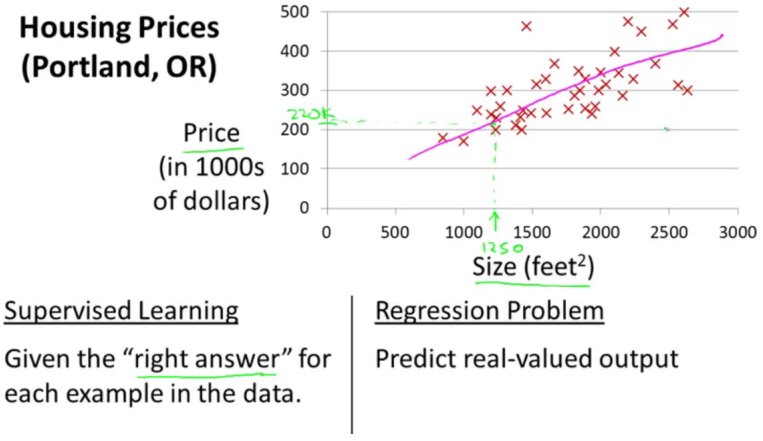


Figura 2.1: Precios y tamaño de casas

Tamaño en $feet^2$ (x)	Precio en 1000 (y)
2104	460
1416	232
1534	315
852	178

Tabla 2.1: Relación precio-área

pasa la recta y θ_1 es la pendiente de la recta. si $\theta_0 = 0$ se dice que tenemos una regresión lineal con una variable, también es llamado regresión lineal univariable.

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{2.1}$$

2.1.2. Función de costo

Lo que se busca es que nuestra hipótesis sea la mejor, esto es que la distancia que hay de cada dato de entrenamiento y la recta que está definida por la función de hipótesis sea la menor posible. Cuando minimizamos estas distancias decimos que estamos minimizando el error. La función de costo nos regresa éste error, como su nombre lo indica nos está regresando el costo, por lo tanto lo que queremos hacer es minimizar esta función. La función de costo se indica en la ecuación 2.2. En este caso estamos trabajando con una sola variable por lo que $\theta_0 = 0$ y para este caso $(h_\theta(x^{(i)})$ es equivalente a $\theta_1 x^{(i)}$. Nuestro objetivo siempre será minimizar la función de costo $J(\theta_0, \theta_1)$.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{2.2}$$

Vamos ver como se comporta la función de costo en a diferentes valores de θ_1, para esto nos ayudaremos de la gráfica que se encuentra en la figura 2.2.
Si $\theta_1 = 1$

$$J(1) = \frac{1}{2m}(1 * 1 - 1)^2 + (1 * 2 - 2)^2 + (1 * 3 - 3)^2 = 0$$

Nuestra hipótesis presenta un costo cero por lo que es una buena hipótesis y posiblemente la mejor.
Si hacemos $\theta_1 = 0{,}5$ tenemos:

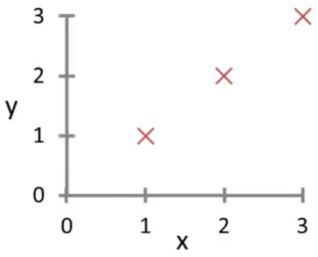


Figura 2.2: Gráfica x-y

$$J(0{,}5) = \frac{1}{2m}(0{,}5 * 1 - 1)^2 + (0{,}5 * 2 - 2)^2 + (0{,}5 * 3 - 3)^2 = \frac{3{,}5}{6} \approx 0{,}58$$

Si hacemos $\theta_1 = 0$

$$J(0) = \frac{1}{2m}(-1)^2 + (-2)^2 + (-3)^2 = \frac{14}{6}$$

Si seguimos dando valores a θ_1 y graficamos los valores de la función de costo veremos que la función de costo es una paráblola donde el punto más bajo son las coordenadas (1,0).

2.2. Parámetro de aprendizaje

2.2.1. Descenso de gradiente

Como vimos anteriormente lo que buscamos en minimizar la función de costos, en esta sección veremos un algoritmo llamado el descenso de gradiente. Dicho algoritmo no sólo se utiliza en la regresión lineal sino en todo el aprendizaje automático. Con el descenso de gradiente podemos minimizar cualquier función no sólo la función de costos.
Supongamos que tenemos una función $J(\theta_0, \theta_1)$ y queremos minimizarla, lo que hacemos es darle un valor aleatorio a θ_0 y θ_1, lo que haremos es cambiar estos valores de manera que que en cada cambio que hagamos $J(\theta_0, \theta_1)$ se cada vez menor.

Supongamos que la superficie que se muestra en la figura 2.3 es la representación gráfica de una función de costos dada una función de hipótesis y aleatoriamente elegimos valores para θ_0 y θ_1 éste punto se muestra en la imagen como la primera cruz negra que esta en el punto más alto. Imaginemos que tu estas parado en una colina que se representa por la función $J(\theta_0, \theta_1)$ vista en la imagen, y tu posición es el punto θ_0 y θ_1, ahora lo que quieres es dar un paso de tal forma que vayas hacia abajo, das este paso y después vuelves a ver en que dirección debes dar el siguiente paso para ir cada vez más abajo, estos pasos se representan en la figura con las estrellas negras.
Como podemos ver después de una serie de repeticiones llegamos al punto más bajo de toda la función, los valores θ_0 y θ_1 son valores bastante aceptables para trabajar nuestra hipótesis. el algoritmo de descenso de gradiente se muestra en la ecuación 2.3 este algoritmo se tiene que repetir hasta que converga.

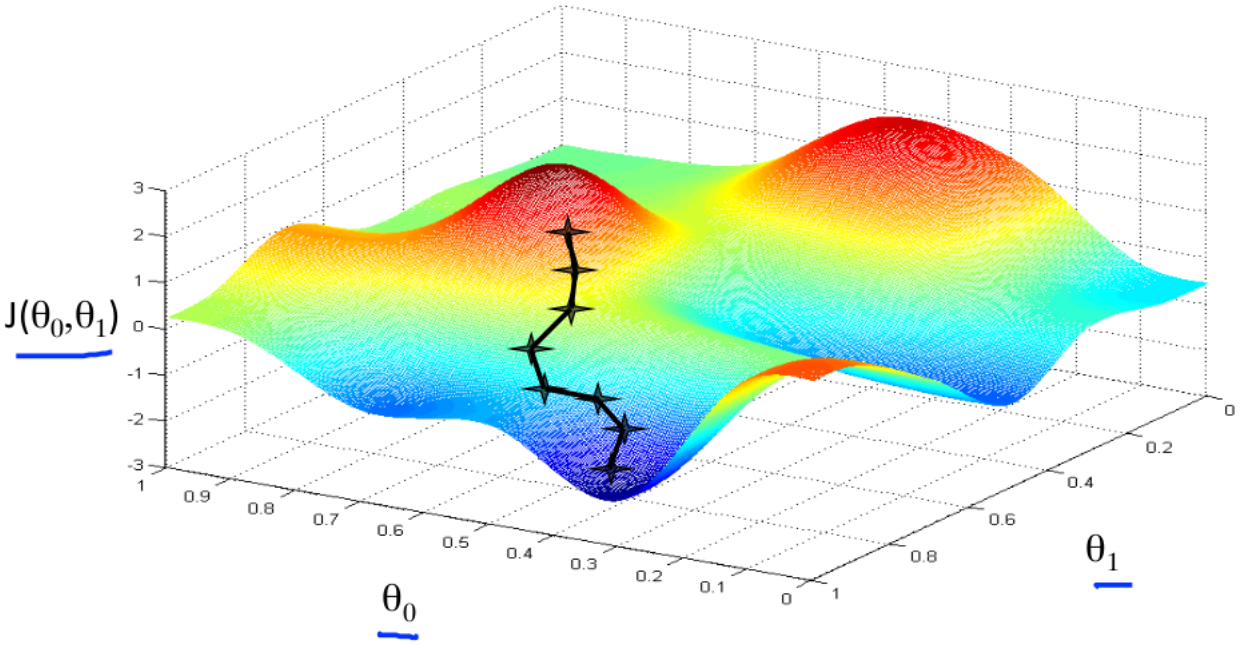


Figura 2.3: Minimización de $J(\theta_0, \theta_1)$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad Para \quad j = 0 \quad y \quad j = 1 \tag{2.3}$$

Lo que le estamos diciendo es que con cada iteración el valor de θ_j se tiene que actualizar a un nuevo valor que tiene que ser menor al anterior.
α es el índice de aprendizaje y lo que hace es controlar qué tan grande el un paso que estamos dando con el gradiente de descenso, si α es muy grande estamos dando pasos muy grandes hacia abajo, si α es pequeño estaremos dando pasos muy pequeños. Uno de nuestros objetivos es encontrar el mejor

valor de α ya que al dar pasos muy grandes puede que oscile todo el tiempo entre valores bajos y altos entrando en un bucle infinito y puede que no converga. Si α da pasos muy pequeños tardará demasiado nuestro algoritmo en encontrar una solución aparte de consumir muchos recursos. Mas adelante veremos como encontrar el mejor valor para α.
Algo que tenemos que tomar en cuenta para este algoritmo es que tiene que haber una actualización simultanea entre los términos θ_0 y θ_1 como se muestra en las ecuaciones desde la 2.4 a 2.7.

$$temp_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \tag{2.4}$$

$$temp_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \tag{2.5}$$

$$\theta_0 := temp_0 \tag{2.6}$$

$$\theta_1 := temp_1 \tag{2.7}$$

Lo que vamos a hacer a continuación es usar el gradiente de descenso para minimizar la función de costo, para esto nos vamos a centrar en las derivadas parciales de dicha función.

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=0}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2 \tag{2.8}$$

A partir de la ecuación 2.8 podemos escribir las derivadas parciales tanto para θ_0 como para θ_1 como se muestra a continuación.

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \tag{2.9}$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)} \tag{2.10}$$

Ahora lo que hacemos es sustituir las ecuaciones 2.9 y 2.10 en 2.4 y 2.5 respectivamente. De igual forma tenemos que actualizar θ_0 y θ_1 hasta que convergan.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \tag{2.11}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \tag{2.12}$$

$$\tag{2.13}$$

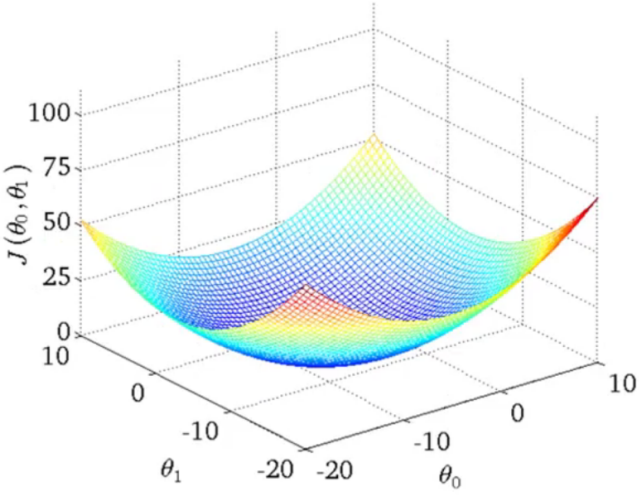


Figura 2.4: Función de costo para una regresión lineal

La función de costos para la regresión lineal siempre será convexa y en forma de arco tal y como se ve en la figura 2.4. Cuando obtenemos el mínimo de la función siempre será un mínimo global ya que es el único punto más bajo que tiene, a diferencia de la función que se muestra en la figura 2.3 que también poseé mínimos locales.
Entonces si encontramos los valores θ_0 y θ_1 del punto más bajo de la función $j(\theta_0, \theta_1)$ y sustituimos dichos valores en la función de hipótesis diremos que dicha función puede predecir un valor con una exactitud aceptable.

Capítulo 3

Regresión lineal con múltiples variables

3.1. Múltiples variables

Hasta este punto habíamos puesto como ejemplo predecir el precio de una casa dado su tamaño, que pasaría si queremos predecir el precio de una casa pero no sólo tenemos su tamaño como variable si también contamos con otras como el numero de habitaciones, pisos, años que tiene la casa de ser construida, etc. Dichas variables se encuentran en la tabla 3.1.

Tamaño ($feet^2$)	Número de habitaciones	Número de pisos	Años de la casa	Precio($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	4178
...	...	...	...	...

Tabla 3.1: Múltiples variables

Ahora vamos a hablar sobre la notación que usaremos. El tamaño sera nuestra variable x_1 el número de habitaciones será x_2 y así sucesivamente, el valor del precio será nuestra variable y.

n: Número de variables.
x^i: variables del i-ésimo dato de entrenamiento (Filas).
x^i_j: Valor de la variable j en el i-ésimo dato de entrenamiento.

En el ejemplo mostrado en la tabla $n = 4$ ya que tenemos 4 variables, $m = 4$ ya que tenemos 4 datos de entrenamiento. El valor que corresponde a $x^{(2)}_3 = 2$ y $x^{(2)}$ se representa como una matriz de $4X1$ cuyos valores son 1416, 3, 2, 40.

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

Al trabajar con mas variables tenemos que cambiar nuestra función de hipótesis como se muestra en la ecuación 3.1.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{3.1}$$

En nuestro ejemplo θ_0 es el precio base de la casa, θ_1 es el precio por $feet^2$, θ_2 es el precio por una habitación, θ_3 es el precio por un piso y θ_4 es el precio por cada año de la casa, podemos considerar la casa pierde su valor en función de los años por lo tanto θ_4 sería un valor negativo.

x_1 es el número de $feet^2$ que tiene la casa, x_2 es el número de habitaciones, x_3 es el número de pisos que tiene la casa, x_4 son los años que tiene la casa. Por convención $x_0^{(i)} = 1$. Lo que haremos a continuación será vectorizar las variables x y θ como se muestra en la ecuación 3.2.

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \dots & \theta_n \end{bmatrix} \tag{3.2}$$

Entonces tenemos que:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T x \tag{3.3}$$

Lo que tenemos que hacer es de igual forma como en el caso de una variable, actualizar los valores de θ con un algoritmo iterativo hasta que estos convergan.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \tag{3.4}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \tag{3.5}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \tag{3.6}$$

$\dots$

Visto de otra forma nuestro algoritmo para múltiples variables queda resumido en la ecuación 3.7

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad Para \quad j := 0 \dots n \tag{3.7}$$

3.2. Escalamiento de variables

Al trabajar con múltiples características tenemos que tomar en cuenta el rango de valores que estas toman, por ejemplo tenemos una característica que es el tamaño de la casa cuyos valores oscilan entre 0-2000 y tenemos otra característica que es el número de habitaciones cuyos valores oscilan entre 1-5.

Al ejecutar el gradiente de descenso puede oscilar dando pasos muy grandes o muy pequeños, este comportamiento se sale de nuestro control y puede que el valor nunca converga.
Lo que hacemos para evitar esto es escalar los valores de nuestras características, es decir que todas tomen un rango similar idealmente entre $-1 \leq x_i \leq 1$ es aceptable un rango un poco más grande o más pequeño.

Por ejemplo que los rangos de valores de nuestras características se encuentren en $0 \leq x_1 \leq 3$ o $-2 \leq x_2 \leq 0{,}5$, lo que no está permitido es tomar rangos muy grandes o muy pequeños como $-100 \leq x_3 \leq 100$ o $-0{,}0001 \leq x_4 \leq 0{,}0001$.
La ecuación que usaremos para hacer este escalamiento se muestra en la ecuación 3.8.

$$x_i := \frac{x_i - \mu_i}{s_i} \tag{3.8}$$

Donde:
x_i es la característica que queremos escalar.
μ_i es el promedio de los valores de dicha característica.
s_i Es el rango de valores (Valor maximo - Valor mínimo). o también puede ser una derivación

estandar

Por ejemplo si queremos escalar la característica de número de habitaciones haríamos lo siguiente.

$$x_2 = \frac{\#Habitaciones - 2}{5}$$

3.3. Índice de aprendizaje

Ahora vamos a hablar del parámetro de aprendizaje α. Como ya habíamos dicho, uno de nuestros objetivos principales es minimizar la función $J(\theta)$ por lo que a cada iteración ésta debe ser menor, si la graficamos en el eje Y y el número de iteraciones en el eje X, debemos de observar una gráfica en la que si $x \to \infty$, $y \to 0$. Decimos que $J(\theta)$ converge si decrece menos de 10^{-3} en cada iteración.

Si al graficar $J(\theta)$ en el eje Y y el número de iteraciones en el eje X y vemos una curva que conforme $x \to \infty$, $y \to \infty$ vemos que no converge, en este caso tenemos que hacer uso de una α más pequeña. El número de iteraciones necesarias para que nuestra función converga es variable, para una aplicación nos puede tomar 30 iteraciones mientras que para otra distinta nos puede tomar 30,000 y para otra aplicación 30,000,000.

Es muy difícil saber de antemano cuantas iteraciones necesitará nuestro algoritmo, podemos dar con un aproximado trazando las gráficas de $J(\theta)$ - Iteraciones y $J(\theta)$ - θ. Otra cosa que hay que tener en cuenta es que para un valor de α lo suficientemente pequeño, $J(\theta)$ debe decrecer en cada iteración, pero si α es muy pequeña, el gradiente de descenso puede converger de una forma mucho más lenta.

Resumiendo:
Si α es muy pequeño converge muy lento.
Si α es muy grande $J(\theta)$ puede no decrecer con cada iteración y posiblemente no converger.
Para escoger valores de α podemos elegir entre 0.001, 0.01, 0.1, 1.

3.4. Características y regresión polinomial

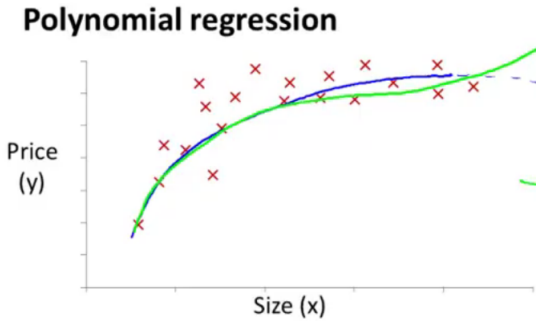


Figura 3.1: Regresión polinomial

En ocasiones tenemos datos que al graficarlos no parece que una linea recta se ajuste de la mejor manera sino que tenemos que hacer uso de una función polinomial.
Veamos el ejemplo de una casa, al graficar el precio y el tamaño vemos que no parece una línea recta, si usamos un polinomio de la forma $\theta_0 + \theta_1 x + \theta_2 x^2$ vemos como se muestra con la línea azul en la figura 3.1 que se ajusta un poco mejor en cierto rango de valores, sin embargo este polinomio de grado dos tiende a decender con forme el eje x tiende a ser mayor, y no tiene sentido que una casa a mayor tamaño cueste menos.
Veamos con un polinomio de grado 3 $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_2 x^3$ vemos que se parece en cierta forma a nuestro polinomio de grado 2, sin embargo cuando éste desciende, el de grado 3 incrementa como lo muestra la línea verde en la figura 3.1, tiene más sentido que una casa a mayo tamaño el precio sea mayor. Ahora cómo hacemos para ajustar el modelo cúbico que tiene tres características a nuestro problema de las casas que en este caso sólo tenemos la característica del tamaño y con base en ello obtenemos el precio. Lo que podemos hacer es tomar el tamaño y reemplazarlo por la variable x como vemos en la ecuación 3.9.También podemos combinar nuestras características con el fin de mejorar nuestra hipótesis, por ejemplo, si

tengo dos características (Largo y Ancho de mi casa), las puedo combinar haciendo una sola (Área de mi casa = LargoXAncho).

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 \theta_3 x_3 \\ h_\theta(x) &= \theta_0 + \theta_1(tamaño) + \theta_2(tamaño)^2 \theta_3(tamaño)^3 \\ x_1 &= (tamaño) \\ x_2 &= (tamaño)^2 \\ x_3 &= (tamaño)^2 \end{aligned} \tag{3.9}$$

Al trabajar con regresión polinomial resulta de gran importancia el escalamiento ya que por ejemplo si el tamaño de la casa va de 1-1000, al elevarlo al cuadrado va de 1-1,000,000 y al elevarlo al cubo es de 1-10^9.

3.5. Ecuación Normal

La ecuación normal para algunos problemas de regresión lineal nos dará una mejor manera de resolver los valores de θ para el valor óptimo. Hasta este punto el algoritmo que hemos usado para minimizar nuestra función de costo es el gradiente de descenso que involucra muchas iteraciones. En contraste la ecuación normal nos da un método para despejar θ de forma analítica, esto para evitar usar un algoritmo iterativo y calcular en un sólo paso el valor óptimo de θ. Lo que hacemos para encontrar el valor mínimo de θ es igualar la derivada de la función $J(\theta)$ a cero y encontrar los valores de θ.

Tomemos el ejemplo de las casas con múltiples características que tenemos en la tabla 3.1 lo que haremos será añadir x_0 que como vimos anteriormente $x_0^j = 1$ quedando como se muestra en la tabla 3.2.

x_0	Tamaño ($feet^2$)	Número de habitaciones	Número de pisos	Años de la casa	Precio($1000)
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	4178

Tabla 3.2: Datos de casas

Ahora vamos a construir una matriz que contiene todos mis datos de entrenamiento, así como el vector y.

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

A partir de estos vectores podemos calcular θ como se muestra en la ecuación 3.10.

$$\theta = (X^T X)^{-1} X^T y \tag{3.10}$$

Las ventajas de usar la ecuación normal son:

- No se necesita elegir un valor para α
- No se necesita iterar.

Las ventajas del gradiente de descenso son:

- Funciona bien aunque n sea demasiado grande.

Las desventajas de la ecuación normal son:

- Se necesita calcular $(X^T X)^{-1}$
- Es muy lento si n es muy grande.

Conviene optar por la ecuación normal cuando $n < 10,000$. Cuando queremos resolver un problema por ejemplo de clasificación como regresión logística nos damos cuenta que el algoritmo de la ecuación normal no funciona por lo que tendremos que optar por el gradiente de descenso.

Capítulo 4

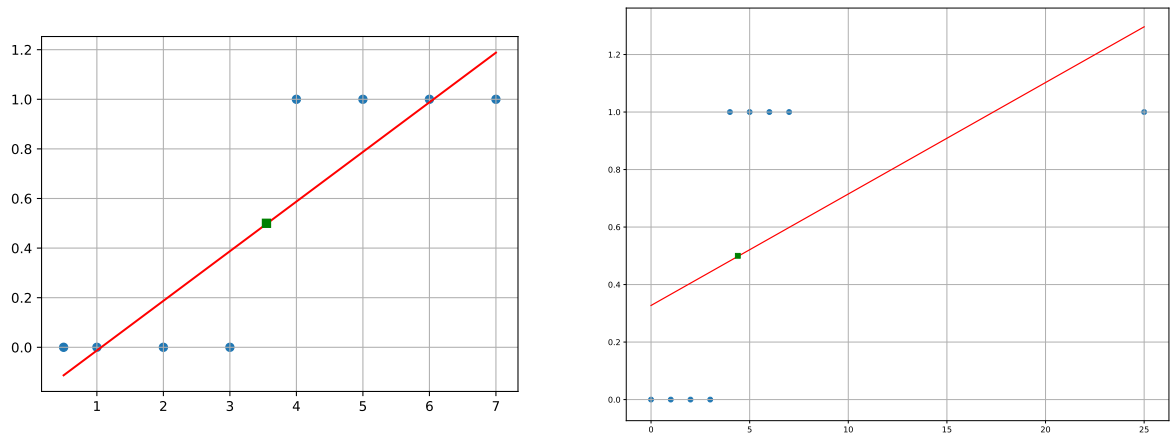
Regresión logística

4.1. Clasificación

Usamos algoritmos de clasificación cuando la variable y el valor que queremos predecir tienen valores discretos, el algoritmo que usamos se llama regresión logística que es de los más utilizados. Algunos ejemplos donde utilizamos clasificación son:
Predecir si un correo es spam o no, si tenemos un paciente con un tumos y queremos predecir si este tumor es maligno o benigno. En este último ejemplo si nuestro algoritmo nos devuelve un 0 esto quiere decir que se trata de un tumor benigno, si nos devuelve un 1, se trata de un tumor maligno. Si graficamos estos valores donde el eje x representa el tamaño del tumor y en el eje Y si es 1 o 0, y aplicamos nuestra función de hipótesis para el caso de regresión lineal $h_\theta(x) = \theta^T x$ diremos que:

$$Si \quad h_\theta(x) \geq 0{,}5 \quad y = 1 \qquad Si \quad h_\theta(x) < 0{,}5 \quad y = 0$$

Como se ve en la figura 4.1a la predicción es buena. Sin embargo si agregamos muchos valores puede que nuestra recta no se ajuste perfectamente y nos de muchos errores como se ve en la figura 4.1b. Otro problema al usar regresión lineal en este caso es que la función de hipótesis nos regresa un valor que en la mayoría de los casos no es un valor 0 o 1 que es lo que nos interesa.
En conclusión, usar un algoritmo de regresión lineal en un problema de clasificación no es para nada una buena idea y no se debe usar nunca.



(a) (b)

Figura 4.1: Regresión lineal aplicada a un problema de regresión logística

4.2. Representación de la hipótesis

Vamos a modificar nuestra función de hipótesis que teníamos en la regresión lineal y la vamos a adaptar para un problema de clasificación. Nuestra función de hipótesis estará dada por:

$$h_\theta(x) = g(\theta^T x) \tag{4.1}$$

Donde $g(z)$ es:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4.2}$$

La ecuación 4.2 recibe el nombre de función sigmoide o función logística. Por lo tanto nuestra función de hipótesis para la regresión logística es:

$$h_\theta(x) = g\Big(\frac{1}{1 + e^{(-\theta^T x)}}\Big) \tag{4.3}$$

Podemos ver en la ecuación 4.3 que:
Si $\theta^T x$ tiende a $\infty h_\theta(x)$ tiende a 1.
Si $\theta^T x$ tiende a $-\infty$ $h_\theta(x)$ tiende a 0.
La función sigmoide se puede ver en la figura 4.2.

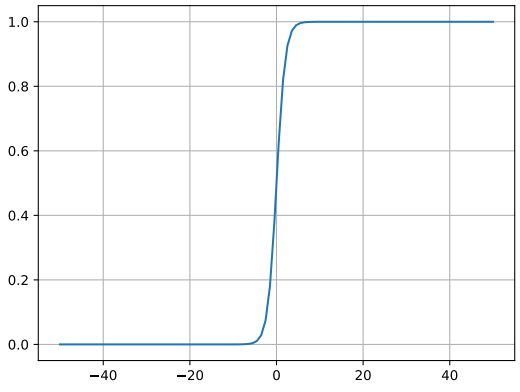


Figura 4.2: Función sigmoide

4.2.1. Interpretación del resultado en nuestra hipótesis

Si por ejemplo $h_\theta(x) = 0{,}7$ diremos que $y = 1$ es decir que el paciente tiene el 70 % de probabilidad de que el tumor sea maligno.

La probabilidad de que y=1 se representa así $h_\theta(x) = P(y = 1|x; \theta)$. La probabilidad de que y=1 + la probabilidad de que y=0 debe ser 1. Por lo tanto diremos que la probabilidad de que y=0 = 1-La probabilidad de que y=1.

$$P(y = 0|x; \theta) = 1 - P(y = 1|x; \theta) \tag{4.4}$$

4.3. Límite de decisión

El límite de decisión es la frontera que separa nuestros datos discretos, por ejemplo si en una gráfica tenemos los datos de tumores malignos y benignos agrupados, podemos trazar una curva que los separe, esta curva recibe el nombre de límite de decisión o Decision boundery.
Supongamos que tenemos la siguiente función de hipótesis.

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \tag{4.5}$$

Donde el vector θ esta dado por:

$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

Por lo tanto la función quedará expresada de la siguiente forma.

$$h_\theta(x) = g(-3 + x_1 + x_2)$$

Ahora nuestra predicción será y=1 si y solo si $-3 + x_1 + x_2 \geq 0$, entonces tenemos la ecuación $x_1 + x_2 = 3$ esta línea sera nuestro límite de decisión, es decir:

$$y = 1 \quad Si \quad x_1 + x_2 \geq 3; \qquad y = 0 \quad Si \quad x_1 + x_2 < 3$$

La gráfica que muestra lo que hemos analizado se encuentra en la figura 4.3 Nos podemos encontrar

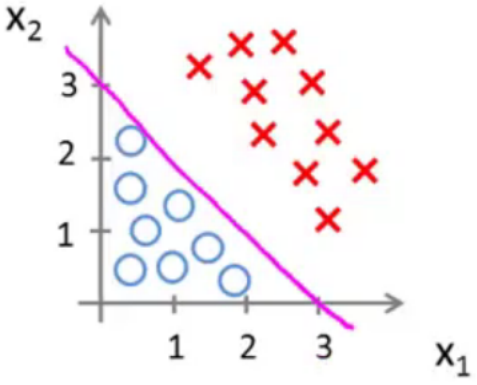


Figura 4.3: Límite de decisión

con casos en los que una línea recta no soluciona el problema y no pude ser nuestro límite de decisión. Lo que podemos hacer es ajustar nuestra curva de tal forma que sea un polinomio de grado n. Pongamos como ejemplo los datos que se encuentran graficados en la figura 4.4. Nuestra

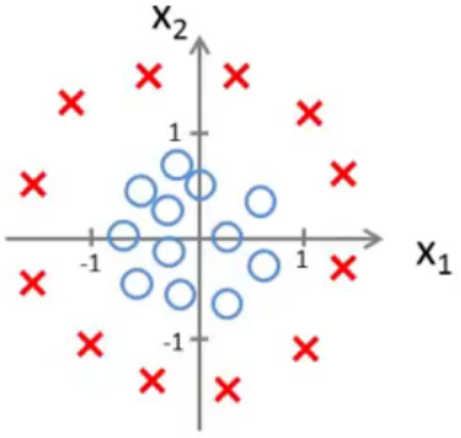


Figura 4.4: Datos no lineales

hipótesis será para este caso:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Donde:

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Por lo tanto nuestra predicción sera y=1 si $-1 + x_1^2 + x_2^2 \geq 0$ por lo que tenemos:

$$x_1^2 + x_2^2 \geq 1$$

Que es la ecuación de una circunferencia con centro en el origen y radio 1. Como vemos en la figura esta curva se adapta muy bien a nuestros datos. Modificando los valores del vector θ y el grado del polinomio, podemos trazar la curva que mejor se adapte a nuestros datos.

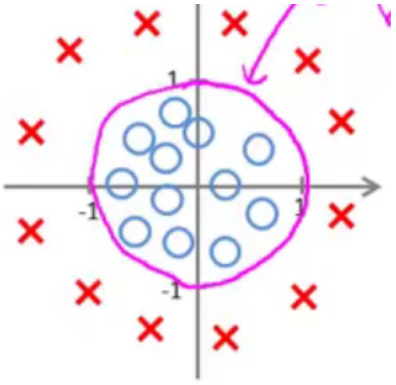
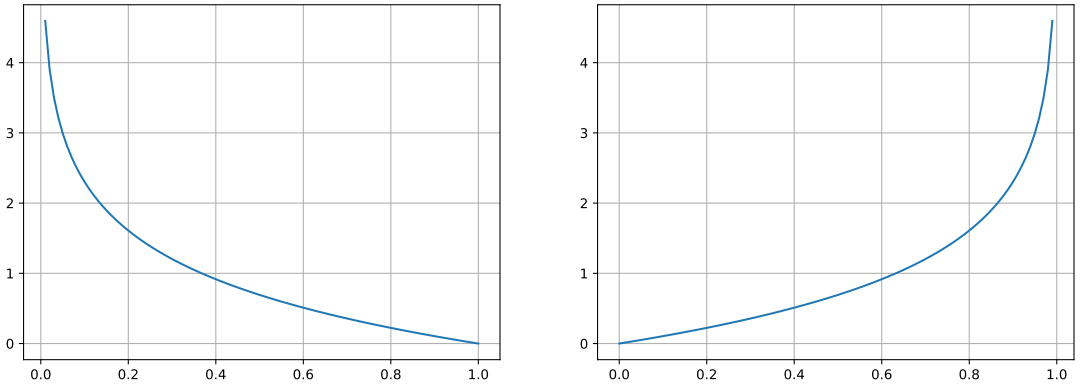


Figura 4.5: Límite de decisión no lineal

4.4. Función de costo

Hasta ahora hemos visto la definición de un problema de clasificación y hemos dicho que para resolverlo tenemos que hacer uso de la regresión logística pero no hemos visto como escoger el mejor parámetro para θ. Antes que nada vamos a definir nuestra función de costos que viene dada por la expresión que se muestra en la ecuación 4.6.



(a) $h_\theta(X)$ dado que y=1 (b) $h_\theta(X)$ dado que y=0

Figura 4.6: Gráficas de la función de costo para regresión logística

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & si \quad y = 1 \\ -log(1 - h\theta(x)) & si \quad y = 0 \end{cases} \tag{4.6}$$

La gráfica que muestra el comportamiento para la función de costo si y=1 se muestra en la figura 4.6a. Como podemos ver si y=1 y nuestra función de hipótesis nos dice que y=0 (Predicción errónea), se penaliza el algoritmo con un costo muy alto. Nótese que si nuestro algoritmo predice y=0 (Predicción correcta) el error se minimiza. Para este caso:

$$Costo \Rightarrow \infty \quad Si \quad h_\theta(x) \Rightarrow 0$$

Ahora para el caso en el que y=0 sucede algo similar como se puede observar en la figura 4.6b. Para este caso:

$$Costo \Rightarrow \infty \quad Si \quad h_\theta(x) \Rightarrow 1$$

Podemos simplificar la ecuación 4.6 como se muestra a continuación.

$$Cost(h_\theta(x), y) = -y(h_\theta(x)) - (1 - y)log(1 - h_\theta(x)) \tag{4.7}$$

Ahora ya podemos definir una función de costos definitiva con la que estaremos trabajando a lo largo del curso y la más utilizada para problemas de clasificación, dicha expresión se muestra en la ecuación 4.8.

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \\ J(\theta) &= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)})log(1 - h_\theta(x^{(i)}))\right] \end{aligned} \tag{4.8}$$

Podemos implementarla de forma vectorial como se expresa en la ecuación **??**.

$$\begin{aligned} h &= g(X\theta) \\ J(\theta) &= \frac{1}{m}(-y^T log(h) - (1 - y)^T log(1 - h) \end{aligned} \tag{4.9}$$

4.5. Descenso de gradiente

En esta sección nos ocuparemos de encontrar el parámetro θ que minimiza la función $J(\theta)$. El algoritmo que se usará es muy similar al que usamos en la regresión lineal y es el siguiente.

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta((x^{(i)}) - y^{(i)})x_j^{(i)}) \tag{4.10}$$

Repetiremos el algoritmo que se expresa en la ecuación 4.10 hasta que converga, es decir, hasta que encontremos un valor tal que $J(\theta)$ sea mínimo.
Como podemos ver este algoritmo es muy parecido al que ocupamos en la regresión lineal con la excepción de que en este caso $h_\theta(x)$ es la función sigmoide (Ecuación 4.3).
Podemos implementarla en forma vectorial como se muestra en la ecuación 4.11.

$$\theta := \theta - \alpha\frac{1}{m}X^T(g(X\theta) - \bar{y}) \tag{4.11}$$

4.6. Algoritmos avanzados de optimización

Hasta ahora el único algoritmo de optimización era el que se muestra en la ecuación 4.10 que es el algoritmo de descenso de gradiente, pero también existen otros como lo son:

- Gradiente conjugado BFGS
- Gradiente conjugado L-BFGS

Estos algoritmos son más avanzados y requieren de cálculo numérico avanzado para obtener las derivadas, una de las ventajas de estos algoritmos es que no necesitas una tasa de aprendizaje α porque emplea un algoritmo de búsqueda lineal que automáticamente prueba distintos valores para α de forma que incluso se puede seleccionar una tasa de aprendizaje diferente para cada iteración. Estos algoritmos generalmente terminan convergiendo más rápido que el gradiente de descenso. Debido a su complejidad, no es conveniente programar por nosotros mismos estos algoritmos sino usar bibliotecas que alguien más ya programó.
Lo primero que tenemos que hacer es definir las funciones $J(\theta)$ y $\frac{\partial}{\partial\theta_j}J(\theta)$. Lo podemos hacer escribiendo una función que regrese estos dos valores.

```
function [jVal, gradient] = costFunction(theta)
jVal = [...code to compute J(theta)...];
gradient = [...code to compute derivative of J(theta)...];
end
```

Ahora tenemos que usar la función $fminunc()$ que optimiza el algoritmo usando la función $optimset()$, este código se puede implementar de la siguiente forma.

```
options = optimset('GradObj', 'on', 'MaxIter', 100);
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] = fminunc(@costFunction, ...
initialTheta, options);
```

Entonces le damos pasamos a la función $fminunc()$, la función de costo, un vector inicial de los valores de theta, y las opciones, éste último es el vector que ya hemos creado.

4.7. Clasificación multiclase

Hemos estudiado el caso en el que la variable y puede tomar dos valores, esto es $y \in 0, 1$, pero que pasa si queremos clasificar más de dos valores, como ya vimos un problema de clasificación involucra valores discretos, esto nos dice que no sólo pueden ser dos valores, como hemos visto hasta ahora.
Si tenemos un problema en el cual $y \in 0, 1, \ldots, n$, lo que debemos hacer es dividir nuestro problema en $n+1$ problemas de clasificación, en cada uno de estos sub-problemas y predecimos la probabilidad de que ocurra y. Esto es:

$$\begin{aligned} y &\in 0, 1, \ldots, n \\ h_\theta^{(0)}(x) &= P(y = 0|x;\theta) \\ h_\theta^{(1)}(x) &= P(y = 1|x;\theta) \\ &\cdots \\ h_\theta^{(n)}(x) &= P(y = n|x;\theta) \\ prediccion &= max(h_\theta^{(i)}(x)) \end{aligned}$$

Lo que haremos será obtener las probabilidades de cada caso y diremos que nuestra predicción será el valor de y para el cual la probabilidad $h_\theta^{(i)}(x)$ es máxima.

Capítulo 5

Regularización

5.1. Sobre Ajuste (Overfitting)

El sobre ajuste es un problema de machine learning que se presenta cuando nuestra predicción se ajusta exactamente a nuestros datos, esto podría parecer bueno, sin embargo es un gran problema ya que para datos con los que entrenamos el algoritmo, éste funciona muy bien y para datos nuevos que nuestro algoritmo nunca antes ha visto, funciona muy mal, podríamos decir que nuestro algoritmo memorizó los datos con los que lo estábamos entrenando, pero no aprendió nada de ciertos datos, por lo tanto al querer obtener una predicción, ésta no se acerca nada al valor real.

El *Underfitting* es otro problema, éste ocurre cuando nuestro algoritmo no se ajusta ni a los datos dados ni a nuevos datos. Podemos hacer una analogía con tres tipos de estudiantes: El estudiante **A** es aquel que se presenta a un examen sin estudiar nada, el estudiante **B** es aquel que se presenta a un examen para el cual estudió, por último tenemos al estudiante **C** el cual consiguió las preguntas que vendrían en el examen y memorizó las respuestas. El estudiante **A** es un ejemplo de *Underfitting* podrá responder el examen pero lo hará sin acertar ninguna respuesta. El estudiante **C** es un ejemplo de *Overfitting*, obtendrá la calificación más alta en el examen, sin embargo cuando tenga que resolver problemas parecidos pero no exactos a los del examen, no podrá resolverlos, ni siquiera acercarse un poco al resultado correcto. Por último el estudiante **B** es un ejemplo del ajuste que queremos conseguir, es el que más nos conviene, el estudiante tal vez no obtenga la calificación más alta como el estudiante **C** pero cuando se enfrente a problemas similares a los del examen podrá enfrentarlos obteniendo resultados que se acercan mucho al valor deseado.

Overfitting: Si tenemos muchas variables, la hipótesis puede ajustarse correctamente a nuestro training set pero fallará con nuevos datos, esto se puede ver en la ecuación 5.1. Si $m \to \infty$, $J(\theta) \to 0$. En la figura 5.1 se muestra un ejemplo gráfico para el caso de la regresión lineal y en la figura 5.2 se muestra el ejemplo para el caso de regresión logística.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i - y^i)^2 \tag{5.1}$$

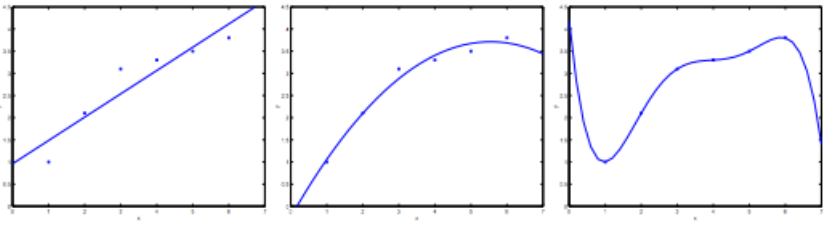


Figura 5.1: Regresión Lineal

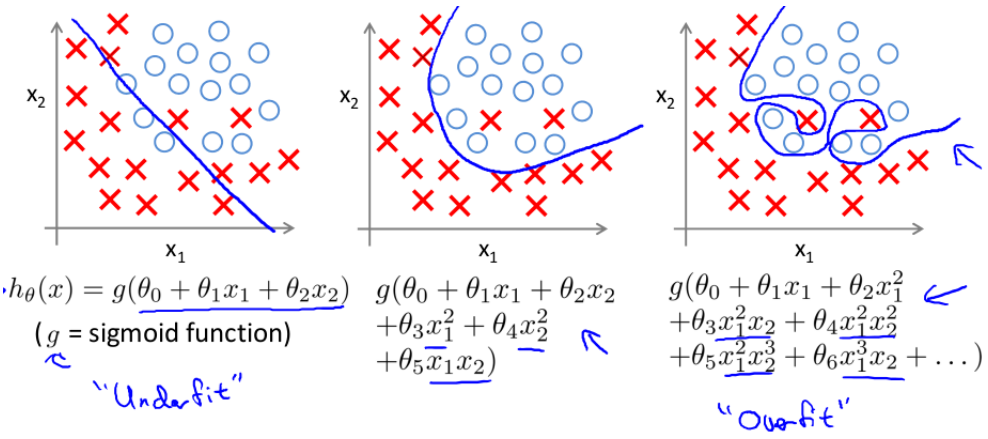


Figura 5.2: Regresión Logística

5.1.1. Evitar el overfitting

El overfitting ocurre principalmente cuando tenemos pocos datos de entrenamiento y muchas variables, en este caso podemos recurrir a dos opciones:

1. Reducir el número de variables.
   - Selección manual de cuales variables mantener. En este caso debemos revisar cuales son las más importantes y eliminar las demás.
   - Selección por medio de un algoritmo (Se tratará en capítulos posteriores).
2. Regularización
   - Mantener las variables pero reducir la magnitud de los parámetros θ_j
   - La regularización funciona bien cuando tenemos muchas variables útiles.

5.2. Función de costo

En esta sección lo que buscamos es encontrar una forma de la función de costos que vamos a utilizar al momento de implementar la regularización. Imaginemos que tenemos cuatro variables para el problema de el precio de una casa, con cuatro variables nuestra función será un polinomio de grado 4 y se verá más parecida al tercer recuadro de la figura 5.1 que al segundo, entonces lo que buscamos es obtener un polinomio de grado 2, en la ecuación 5.2 podemos ver un ejemplo de nuestra función de costos con las variables 3 y 4.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 + 1000\ \theta_3^2 + 1000\ \theta_4^2 \tag{5.2}$$

El número 1000 se puso para simbolizar que la variables x^3 y x^4 respectivamente, pueden ocupan números muy grandes, vemos que la única forma de que la función de costos sea pequeña es que tanto los parámetros θ_3 y θ_4 se aproximen a 0.
Cuando tenemos muchas variables, por ejemplo 100 y no sabemos cuál eliminar, podemos conservar todas modificando la función de costos de tal manera que cada parámetro θ_j^2 lo reduzcamos a un valor pequeño, la función de costos quedará como en la ecuación 5.3.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m} h_\theta(x^i - y^i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2 \tag{5.3}$$

Donde λ es el parámetro de regularización. Si elegimos λ muy pequeño en la hipótesis se eliminarán todos los parámetros θ a excepción de θ_0 de tal forma que obtendremos una línea recta $h_\theta(x) = \theta_0$.

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Por otro lado si λ es muy grande, tenemos el caso de *underfitting*.

5.3. Regresión lineal regularizada

Como se trató en capítulos anteriores, usamos un algoritmo iterativo para encontrar los valores θ_j que mejor se ajustan a nuestra hipótesis, este algoritmo es el del `gradiente de descenso`, tomando en cuenta la taza de aprendizaje α y el parámetro de regularización λ, nuestro algoritmo queda definido como se ve en la ecuación 5.4. El parámetro λ no lo ocuparemos en θ_0.

$$\begin{aligned}
\theta_j :=& \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i + \frac{\lambda}{m}\theta_j\right] \\
\theta_j :=& \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i \qquad j \in \{1, 2, \ldots, n\} \qquad (5.4) \\
\theta_0 :=& \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_0^i \qquad (5.5)
\end{aligned}$$

El segundo algoritmo que vimos para ajustar un problema de regresión lineal fue el de la `ecuación normal` en el cual creamos una matriz $\mathbf{X}$ y un vector $\mathbf{y}$. La ecuación normal queda como se muestra en la ecuación 5.6, cabe destacar que la dimensión de la matriz que acompaña al parámetro λ es de (n+1)*(n+1).

$$\theta = \left(X^T X\lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}\right)^{-1} X^T y \qquad (5.6)$$

La función de costo queda definida como

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

5.4. Regresión logística regularizada

El proceso de regularización para la regresión logística, es similar al de regresión lineal quedando de la siguiente manera.

$$\begin{aligned}
\theta_0 :=& \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_0^i \\
\theta_j :=& \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i + \frac{\lambda}{m}\theta_j\right]
\end{aligned}$$

Al ver las similitudes entre las ecuaciones para la regresión logística y para la regresión lineal podemos pensar que se trata de exactamente lo mismo, sin embargo no es así ya que para la regresión logística la función de hipótesis es diferente, como recordaremos:

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

La función de costo queda definida como:

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

Podemos usar algoritmos avanzados de optimización

```
function [jVal, gradient] = costFunction(theta)
jVal = [...code to compute J(theta)...];
gradient = [...code to compute derivative of J(theta)...];
end
```

Capítulo 6

Redes neuronales

6.1. Hipótesis no lineal

Hasta el momento hemos conocido dos algoritmos, uno de clasificación y otro de regresión lineal, en este capítulo nos ocuparemos de un tercer algoritmo llamado redes neuronales, éste existe desde hace mucho tiempo, sin embargo hoy en día ha tomado bastante fuerza con la revolución tecnológica ya que tiene bastantes aplicaciones.

La importancia de usar redes neuronales es que para muchos problemas de aprendizaje automático, el número de variables es bastante grande y como ya vimos en capítulos anteriores, esto puede ocasionar un problema de *overfitting* a parte de que se vuelve complicado manejar tantas variables.
Un ejemplo en el que podemos observar la gran cantidades de variables que se usan en un problema de Machine Learning, es la visión computacional. Cuando una computadora "veüna imagen, lo que hace internamente es trabajar sobre una matriz de números que representan la intensidad de cada pixel de dicha imagen.

Si tratamos de resolver este problema con el algoritmo de regresión logística, separando las imágenes que son carros de las que no, y suponiendo que tratamos con imágenes pequeñas de 50x50 pixeles, tendríamos una matriz de 2500 variables, esto considerando que las imágenes están en escala de grises, si las trabajamos con colores RGB, tendríamos 7500 variables, hasta este punto podríamos pensar que una computadora como las que tenemos hoy en día pueden trabajar cómodamente con esta cantidad de variables, sin embargo para que nuestra función de hipótesis esté completa tenemos que añadir los terminos cuadráticos y en este punto es donde la cantidad de variables que operamos se dispara a 3 millones, con esta cantidad es claro que el costo para hacer trabajar nuestro algoritmo es enorme, no hay que olvidar que estamos trabajando con imágenes pequeñas, de 50 x 50 pixeles, ahora si trabajamos con imágenes de 100 x 100 pixeles estaríamos operando aproximadamente con 50 millones de variables.

6.2. Neuronas y el cerebro

Las redes neuronales son un algoritmo bastante antiguo, surgió con el objetivo de hacer máquinas que pudieran simular el cerebro. La máquina de aprendizaje más asombrosa que conocemos es el cerebro y de ahí el interés de crear algoritmos que lo imitaran. Las redes neuronales fueron utilizadas en los 80's y 90's, su popularidad disminuyó, a pesar de eso, recientemente han tenido un resurgimiento.

Una de las razones de este resurgimiento es la creación de computadoras rápidas y potentes, aunque las redes neuronales es una buena opción sobre regresión logística y regresión lineal, es muy costoso, hoy en día este costo es despreciable en comparación con los recursos computacionales que están al alcance de nuestras manos, esto ha permitido operar redes neuronales de gran escala y, por

esto y otras técnicas, las redes neuronales modernas son la técnica más vanguardista para muchas aplicaciones.

El cerebro humano puede ver y procesar imágenes, escuchar, aprender a solucionar operaciones matemáticas y muchas otras cosas sorprendentes, pareciera que para poder imitar completamente al cerebro se necesitan diferentes partes de software, es decir un programa que procese imágenes, otro que procese audio, otro que aprenda a resolver operaciones matemáticas, etc. Sin embargo existe la hipótesis de que sólo se requiere de un único algoritmo de aprendizaje para poder imitar completamente al cerebro.

Esta hipótesis se basa en que si cortamos la conexión de la corteza auditiva con el oído y la conectamos con el nervio óptico, la corteza que en un principio sólo sabía escuchar, ahora aprenderá a ver. Lo mismo sucede si cortamos la conexión de de la corteza somatosensorial (que se encarga del sentido del tacto) y la conectamos al ojo, aprenderá a ver. Estos comportamientos del cerebro han dado paso a la idea de que si una corteza puede aprender a procesar algo nuevo, también se puede crear un algoritmo que aprenda por sí sólo al estimularlo con nuevos datos diferentes para lo que fue creado en un principio.