Contenido

[Regresión lineal con múltiples variables 1](#_Toc504300498)

[1.1 Características 1](#_Toc504300499)

[1.2 Descenso del gradiente para múltiples variables 4](#_Toc504300500)

[1.3 El descenso del gradiente en la práctica: 7](#_Toc504300501)

[1.3.1 Escalamiento de características 7](#_Toc504300502)

[Normalización respecto a la media 10](#_Toc504300503)

[1.3.2 Tasa de aprendizaje 11](#_Toc504300504)

[1.4 Características y regresión polinomial 14](#_Toc504300505)

[1.5 Ecuación normal 18](#_Toc504300506)

[Matrices no invertibles 25](#_Toc504300507)

[1.6 (Apéndice) Derivando el modelo matricialmente 26](#_Toc504300508)

# Regresión lineal con múltiples variables

## Características

En la versión original de la regresión lineal que hemos desarrollado, tenemos solo un input de entrada (el tamaño de la casa), y queremos usarla para predecir , el de la casa. Esta fue la formulación de nuestra hipótesis.

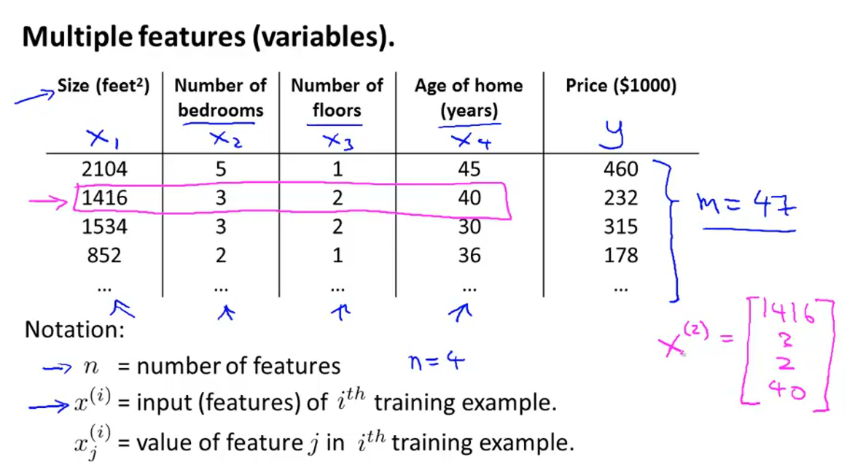
Pero ahora imagina, qué pasa si no solo tuviéramos el tamaño de la casa como característica o como variable para tratar de predecir el precio, sino que también supiéramos el número de habitaciones, el número de pisos y la edad de la casa en años. Parece que esto nos daría mucha más información para predecir el precio.

Para introducir un poco de notación, que en cierto modo ya mencionamos antes, vamos a utilizar las variables "x" subíndice 1, "x" subíndice 2 y así sucesivamente hasta denotar, en este caso, mis cuatro características y voy a seguir utilizando "y" para denotar la variable, el precio variable de salida que estamos tratando de predecir.

Vamos a introducir un poco más de notación. Ahora que tenemos 4 características voy a usar la "n" minúscula para denotar el número de características. Así es que en este ejemplo tenemos n=4 porque tenemos cuatro características. Y "n" es diferente de nuestra notación anterior en donde estábamos usando "m" para denotar el número de ejemplos. Así que si tienes 47 filas, "m" es el número de filas en esta tabla o el número de ejemplos de entrenamiento.

Así es que yo también vamos a utilizar "x" superíndice "i" para denotar las características de entrada del ejemplo de entrenamiento "ith".

Como ejemplo concreto digamos que x2 va a ser un vector de las características para mi segundo ejemplo de entrenamiento. Así que x2 va a ser un vector de [1416, 3, 2, 40], porque esos son las cuatro características que tengo Para tratar de predecir el precio de la segunda casa.



Así que, en esta notación, el superíndice es un índice en mi conjunto de entrenamiento. Esto no es "x" a la potencia 2. Por el contrario, esto es, ya sabes, un índice que te indica mirar la fila 2 de la tabla anterior. Esto se refiere a mi 2 ejemplo de entrenamiento o sujeto 2 de la muestra que tiene sus respectivos inputs y un output.

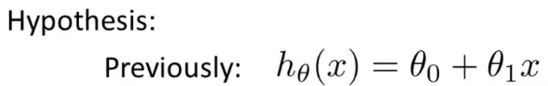
Con esta notación, x2 (por ejemplo) es un vector de cuatro dimensiones. De hecho, en términos más generales, esto es un vector de características en dimensiones.

Con esta notación, x2 es ahora un vector y así, voy a utilizar también x(i) con subíndice "j" para denotar el valor de "j", de la característica número "j" en el ejemplo de entrenamiento "ith".

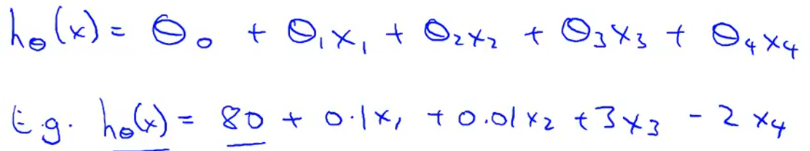
Así que concretamente x2 subíndice 3 (), se referirá a la característica número 3 en el vector o fila que es igual a 2.

|  |  |
| --- | --- |
|  | Numero de sujetos en el conjunto de datos. |
|  | Numero de características de entrada u inputs |
|  | Se usa para denotar las características |
|  | Denota la variable que se usa de output |
|  | Denota al sujeto en la fila i |
|  | Denota la caractérisitca j |

Ahora que tenemos múltiples características, hablemos de cómo debería ser la formulación de nuestra hipótesis. Anteriormente, cuando disponíamos de una sola característica nuestra hipótesis era:

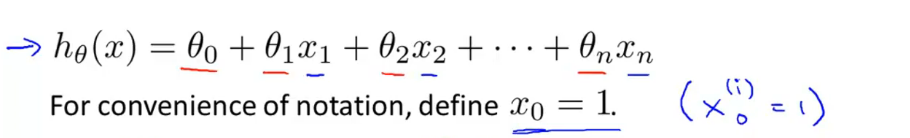


pero ahora que tenemos múltiples características, ya no vamos a utilizar la representación simple. Ahora, la formulación de la hipótesis en la regresión lineal va a ser:



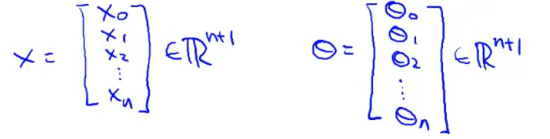
Este sería un ejemplo de una hipótesis y como recordarás, la hipótesis está tratando de predecir el precio de la casa en miles de dólares, diciendo que el precio base de una casa puede ser de 80,000 más cien dólares por pie cuadrado, además el precio sube un poco para cada piso adicional que tiene la casa. x2 es el número de pisos y va en aumento por cada dormitorio adicional que tiene la casa, porque x3 era el número de dormitorios y el precio disminuye un poco con cada año adicional de la edad de la casa.

En una notación más general para características:



Para conveniencia de la notación, "x" subíndice 0 es igual a 1. Concretamente, esto significa que para cada ejemplo "i" tengo un vector de características "x" superíndice "i" y "x" superíndice "i" subíndice 0 va a ser igual a 1. Puedes pensar en esto como la definición de una característica cero adicional. Mientras que anteriormente tuve "n" características debido a x1, x2 hasta xn, ahora estoy definiendo un tipo adicional de vector de características cero que siempre toma el valor de 1. Por lo que ahora mi vector de características "x" se convierte en el vector dimensional n+1 que es el índice cero.

Así es ahora un vector de características dimensionales n+1, pero indexado desde 0 y también vamos a pensar en los parámetros como un vector. Entonces, nuestros parámetros serían nuestros theta0, theta1, theta2 y así hasta llegar hasta theta n, Este es otro vector dimensional n+1.

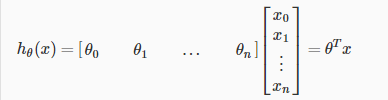


Así es que, ahora mi hipótesis puede escribirse como:



Y esta ecuación es la misma que está en la parte superior porque, como sabes, x0 es igual a 1.

Y lo bueno es que puedo tomar esta formulación de la hipótesis y escribo esto como theta transpuesta de "x":



Esto nos da una manera conveniente para escribir la formulación de la hipótesis como el producto interno entre nuestro vector de parámetro theta y nuestro vector . Sí esta pequeña parte de notación, este pequeño extracto de la convención de notación, la que nos permite escribir esta formulación compacta. De manera que esta es la formulación de una hipótesis cuando tenemos características múltiples. Y, solo para darle otro nombre, también se le llama regresión lineal multivariable. El término multivariable es solamente una forma elegante de decir que tenemos múltiples características, o multivariables para tratar de predecir el valor de

## Descenso del gradiente para múltiples variables

En la sección anterior, hablamos sobre la formulación de la hipótesis para la regresión lineal con múltiples características o con múltiples variables. En esta sección, vamos a hablar de cómo ajustar los parámetros de esa hipótesis. En particular vamos a hablar de cómo utilizar el gradiente de descenso para la regresión lineal con múltiples características.

Para resumir rápidamente nuestra notación, esta es nuestra hipótesis formal en la regresión lineal multivariable:

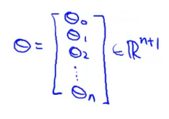


Donde hemos adoptado la convención de que x0 = 1.

Los parámetros de este modelo van de theta0 hasta theta n:

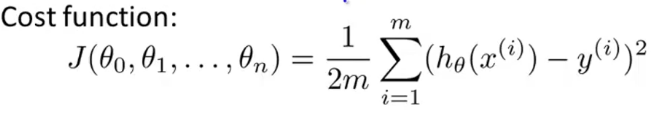


Pero en lugar de pensar en ellos como parámetros separados de "n" (que es válido), en su lugar voy a pensar en los parámetros como theta, en donde theta de aquí es un vector dimensional n+1:



Entonces sólo voy a pensar en los parámetros de este modelo como si fueran un vector.

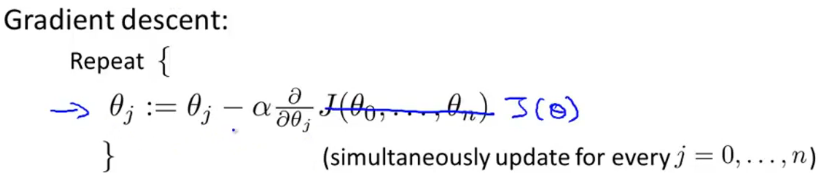
Nuestra función de costo es "J" que va de theta0 a theta n y que está dada por esta habitual suma de cuadrados del término de error:



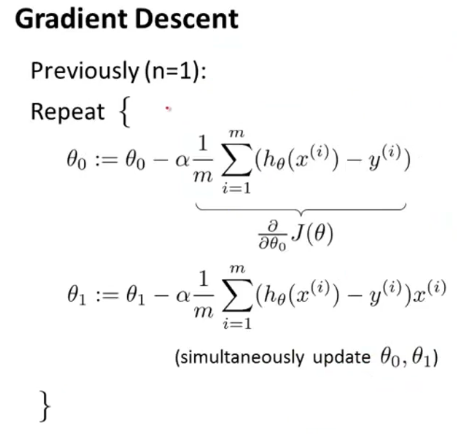


Pero, de nuevo en lugar de pensar en "J" como una función de estos números n+1, voy a escribir más frecuentemente "J" como una simple función del vector de parámetros theta de manera que theta sea un vector.

Y ahora así quedaría gradiente de descenso:



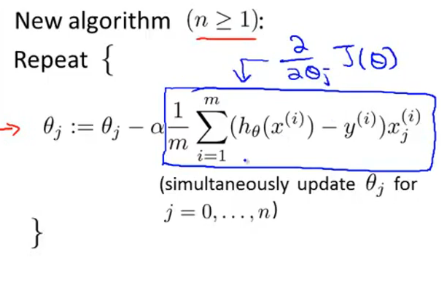
Vamos a actualizar varias veces cada parámetro de acuerdo con menos α multiplicado la derivada. Y de nuevo escribimos esto simplemente como , por lo que se actualiza como menos la tasa de aprendizaje α multiplicada por la derivada, una derivada parcial de la función de costos con respecto al parámetro . Vamos a ver cómo se ve esto cuando implementamos el gradiente de descenso y, en particular, vamos a ver cómo se ve el término derivado parcial. Esto es lo que tenemos para el gradiente de descenso para cuando teníamos la característica n=1:



Teníamos dos reglas de actualización independientes para los parámetros theta0 y theta1. Y la derivada parcial de la función de costos con respecto al parámetro de theta0, y del mismo modo teníamos una regla de actualización diferente para el parámetro theta1.

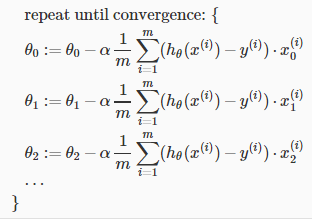
Hay una pequeña diferencia y es que cuando antes teníamos solo una característica, la llamábamos x(i) pero ahora en nuestra nueva notación por supuesto la llamaremos x superíndice i subíndice j para denotar nuestra carácterística.

Entonces eso era para cuando teníamos sólo una característica. Ahora veamos el nuevo algoritmo para el cual tenemos más de una característica, donde el número de características "n" puede ser mucho mayor a 1. Obtenemos esta regla de actualización para el gradiente de descenso y, tal vez para quienes saben cálculo, si se toma la definición de la función de costos y la derivada parcial de la función de costos "J" con respecto al parámetro theta j, encontrarás que la derivada parcial es exactamente ese término al que le puse un recuadro azul alrededor.

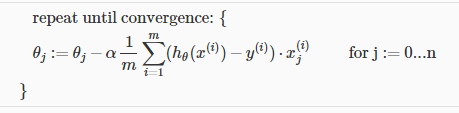


Y si lo implementas obtendrás una implementación funcional del gradiente de descenso para la regresión lineal multivariable. Lo último que vamos a ver en esta sección es explicarte por qué estos algoritmos son más o menos lo mismo o por qué ambos son algoritmos similares o por qué ambos son algoritmos de gradiente de descenso.

Vamos a considerar un caso en donde tenemos dos características o tal vez más de dos características, así que tenemos tres reglas de actualización para los parámetros theta0, theta1, theta2 y tal vez para otros valores de theta también:



En otras palabras:



Si te fijas en la regla de actualización para theta0, puedes encontrar que la regla de actualización de aquí es la misma que la regla de actualización que teníamos antes para el caso de n = 1. Y la razón por la que son equivalentes es, por supuesto, porque en nuestra convención de notación tuvimos la convención de que = 1, por lo que estos dos términos son equivalentes.

Del mismo modo, si ves la regla de actualización para theta1, encontrarás que este término de aquí es equivalente al término que teníamos antes, o a la ecuación o a la regla de actualización que teníamos antes para theta1, en donde, por supuesto, sólo estamos utilizando esta nueva notación para denotar nuestra nueva notación denotando nuestra primera característica y ahora que tenemos más de una característica podemos tener reglas de actualización similares para los otros parámetros como θ2 y así sucesivamente.

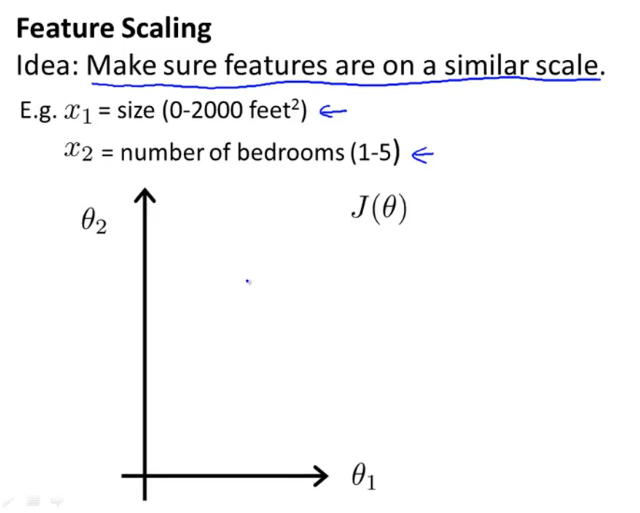
## El descenso del gradiente en la práctica:

## Escalamiento de características

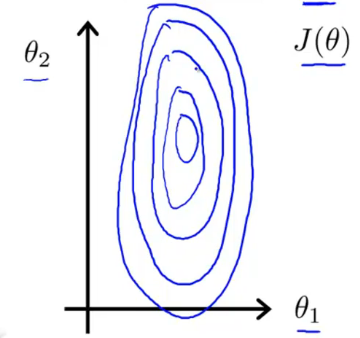
En esta sección, vamos a ver algunos trucos prácticos para lograr que el gradiente de descenso funcione bien. Vamos a hablar acerca de una idea llamada escalamiento de características.

Cuando tienes un problema en donde hay múltiples características, si te aseguras de que las características están en una escala similar, y me refiero a que te asegures que las diferentes características toman rangos de valores similares, entonces los gradientes de descenso pueden converger más rápidamente.

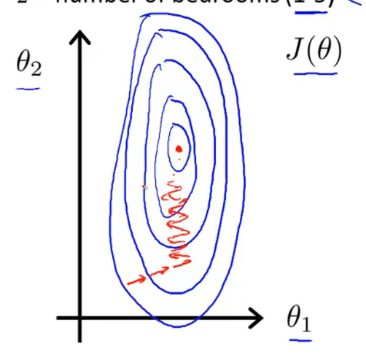
Específicamente, digamos que tienes un problema con dos características en donde x1 es el tamaño de la casa y toma valores entre, digamos, 0 y 2000 y x2 es el número de dormitorios, que tal vez toma valores entre 1 y 5. Si trazas el contorno de la función de coseno "J" de theta, entonces el contorno puede verse así:



En donde, vamos a ver, "J" de theta es una función de parámetros theta 0, theta 1 y theta 2. Vamos a olvidarnos de theta 0 y pretender que es una función de sólo theta 1 y theta 2, pero si x1 puede tomar un rango mucho más amplio de valores y x2, resulta que el contorno de la función de vostes "J" de theta puede una forma elíptica y muy sesgada. Por lo tanto, estas elipses, muy altas y delgadas, o estos óvalos muy altos y delgados, pueden formar los contornos de la función de costes "J" de theta.



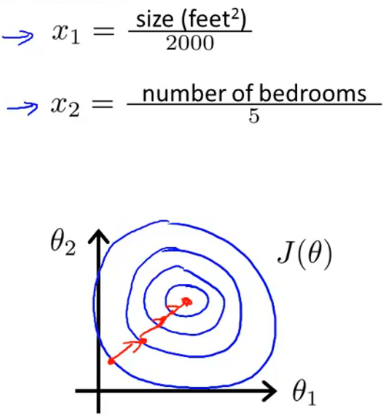
Y si ejecutas el gradiente de descenso en esta función de costes, tus gradientes pueden terminar tomando mucho tiempo y pueden oscilar de ida y vuelta y tardar mucho antes de que finalmente puedan encontrar su camino hacia el mínimo global.



De hecho, puedes imaginar que estos contornos son incluso más exagerados cuando dibujas contornos increíblemente delgados, contornos altos y delgados, y pueden ser incluso más extremos entonces, para el gradiente de descenso simplemente es más difícil encontrar la ruta, serpenteando alrededor de las elipses, y puede tomar mucho tiempo encontrar el camino hacia el mínimo global.

En esta configuración, una cosa útil es escalar las características.

Específicamente, si por el contrario defines que la característica x1 sea el tamaño de la casa dividido entre 2,000, y defines que x2 sea el número de habitaciones dividido entre cinco, entonces los contornos de la función costes "J" puede mucho menos sesgada por lo que los contornos pueden parecer más circulares. Y si ejecutas el gradiente descendente en una función de costes así esta, entonces el gradiente de descenso, puede encontrar una ruta más directa al mínimo global en lugar de tomar un camino mucho más complicado en donde estás tratando de seguir una trayectoria mucho más complicada para llegar al mínimo global.



Así que, al escalar las características que existen, los rangos de valores del consumidor. En este ejemplo, terminamos con ambas características, x1 y x2, entre 0 y 1 y ayudamos al gradiente a converger más rápido.

De manera más general, cuando estamos realizando el escalamiento de características, lo que a menudo queremos es lograr que cada característica tenga un rango aproximado de valores y que específicamente, tu característica x0 sea siempre igual a 1. Así, que ya está en ese rango, pero puedes terminar dividiendo otras características entre diferentes números para llevarlos a ese rango. Imaginamos una variable x1 que varía entre -1 y 1 (los números -1 y +1 no son muy importantes). Por lo que si tienes una característica, x2 que termina estando entre 0 y 3, no será un problema. Si terminas con una característica diferente que varía entre -2 y +0.5 de nuevo, está suficientemente cerca de -1 y +1 lo que, como sabemos, está bien.

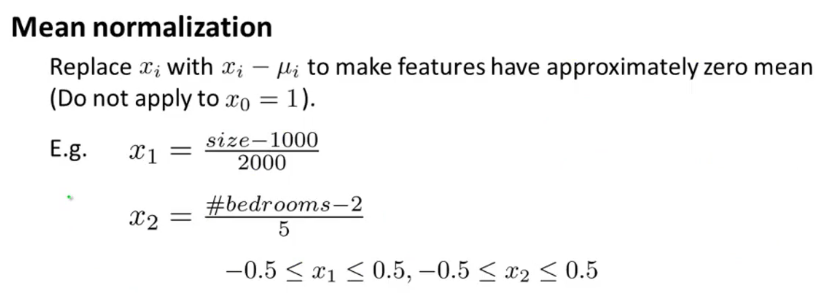
Pero sólo que si tienes una característica diferente, digamos x3 que está entre -100 y +100, entonces, estos son valores muy diferentes a -1 y +1. Por lo tanto, esta podría ser una característica no tan bien escalada y del mismo modo si tus características toman un rango muy pero muy pequeño de valores, por lo que x4 toma los valores entre -0.0001 y +0.0001, entonces de nuevo esto toma un rango de valores mucho menor que el rango de -1 a +1. Y de nuevo, podríamos considerar esta característica mal escalada.

Así que queremos que el rango de valores, pueda ser mayor que +1 o menor que +1, pero no mucho mayor, +100, y no mucho menor como el 0.001 de allá. Diferentes personas tienen diferentes reglas de oro. Pero la que yo utilizo es que si una función toma el rango de valores de digamos, -3 a +3 o un rango más pequeño está bien, pero quizá si toma valores mucho menores como x4, entonces podemos empezar a preocuparnos.

Así que, el mensaje para llevar a casa es que no debes preocuparte si tus características no están exactamente en la misma escala o exactamente en el mismo rango de valores. Pero mientras que todas estén lo suficientemente cerca (numéricamente hablando) el gradiente de descenso debería funcionar bien.

### Normalización respecto a la media

Consiste en restar la media a nuestros valores y dividirlos entre el valor máximo con lo que conseguimos una escala similar medida en desviación respecto a la media.



A veces la gente también realiza lo que se conoce como normalización en torno a la media. Y a lo que me refiero con eso, es que quieres tomar una característica xi y remplazarla con para hacer que tus características tengan una media aproximada de 0.

Y, obviamente no debemos aplicar esto a x0, porque x0 es siempre igual a 1, y si lo aplicáramos tomaría valor 0 y desaparecería.

Pero específicamente para otras características si el rango de tamaños de la casa toma valores de entre 0 y 2000 y si sabes que el tamaño promedio de una casa es igual a 1,000 entonces es posible que debas usar esta fórmula.

El tamaño establece la característica x1 al tamaño menos el valor promedio dividido entre 2000 y de manera similar, en promedio si tus casas tienen de uno a cinco dormitorios y si las casas promedio tienen dos dormitorios, entonces podrías usar esta fórmula para normalizar la media de tu segunda característica x2.

En ambos casos, terminarás por lo tanto con características x1 y x2. Pueden tomar valores más o menos de entre -0.5 y +0.5. No es exacto ya que -x2 puede en realidad ser ligeramente mayor que 0.5, pero estar lo suficientemente cerca. Y la regla más general es que puedes tomar una característica x1 y remplazarla con:

Donde:

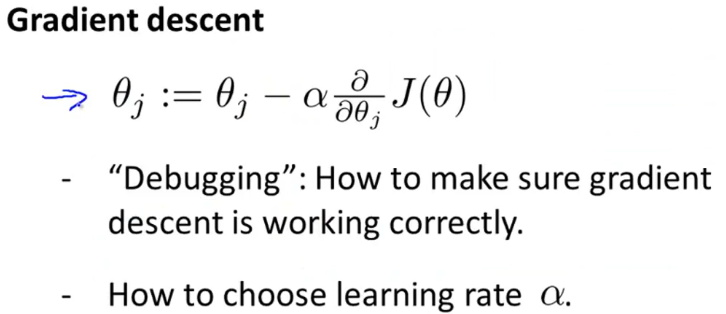
* : es el valor promedio de
* Es el rango de valores esa característica y se refiere al valor máximo menos el valor mínimo ().
* : también puede ser la desviación típica pero dan resultados diferentes. Hay que tener en cuenta que en los ejercicios de programación se usa la desviación típica.

Y este tipo de fórmula hará que tus características, como sabes, no exactamente, pero más o menos estén dentro de este tipo de rangos. Otra consideración para aquellos que están siendo muy cuidadosos técnicamente si estamos tomando el rango como máximo menos mínimo con la característica habitaciones sí el máximo es 5 y el mínimo 1 entonces el rango de sus propios valores es en realidad igual a 4, pero todos estos son aproximados y cualquier valor que hace que las características sean cercanas a cualquiera de estos rangos estará bien. Y el escalamiento de características no tiene que ser demasiado exacto, solo tiene el objetivo de lograr que el descenso del gradiente se ejecute mucho más rápido.

Así es que ahora sabes sobre el escalamiento de características y que si aplicas este simple truco, lograrás que el gradiente de descenso funcione mucho más rápido y converja en muchas menos otras iteraciones.

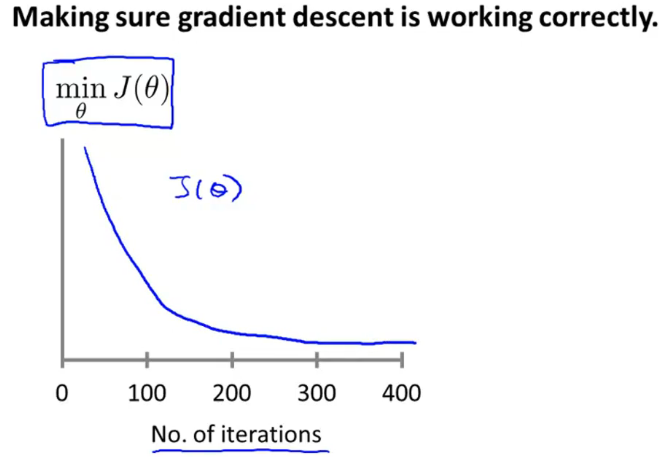
## Tasa de aprendizaje

En esta sección, quiero darte más consejos prácticos para conseguir que el gradiente de descenso funcione. Las ideas de esta sección se centran alrededor del índice de aprendizaje α. Específicamente, aquí está la regla de actualización del gradiente de descenso y lo que quiero hacer en este video es hablarte acerca de lo que pienso de la depuración (debugging) y algunos consejos para asegurarte que funciona correctamente el gradiente de descenso y en segundo lugar, quiero hablarte sobre cómo elegir los índices de aprendizaje.



Aquí hay algo que hago a menudo para asegurarme de que el gradiente de descenso está funcionando correctamente.

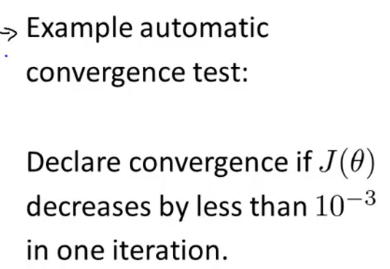
El trabajo del gradiente de descenso es que te encuentre el valor de «theta», que como sabes, esperamos minimice la función de costo "J" de «theta». Por tanto, lo que hago a menudo es arrancar la función de costo de "J" de «theta» conforme se ejecuta el gradiente de descenso. Así es que, el eje x aquí es el número de iteración del gradiente de descenso y conforme el gradiente de descenso se ejecuta, espero que obtengas una gráfica que tal vez se parece a esto:



Observa que el eje x es el número de iteraciones que previamente estábamos buscando en gráficas de "J" de «theta». El eje horizontal es el valor que toma la función de costes en cada iteración. Específicamente, lo que este punto es, voy a clasificar el gradiente de descenso para cien iteraciones.

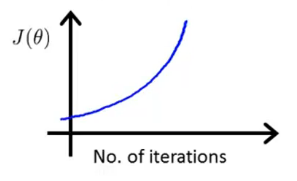
Así que lo que esta gráfica está mostrando, está mostrando el valor de tu función de costo después de cada iteración del gradiente de descenso. Y, si el gradiente de descenso está funcionando correctamente, entonces "J" de «theta» debe disminuir después de cada iteración. Y una cosa útil que este tipo de gráfica puede decirte también es que si te fijas, parece que para el momento en que ha llegado a trescientas iteraciones, parece que "J" de «theta» no disminuye mucho más. Así que para cuando alcances cuatrocientos iteraciones, parecerá que la curva se ha vuelto prácticamente plana por lo tanto podemos intuir que más o menos, el gradiente ha convergido porque su función de costo no bajará mucho más. Así que al observar esta figura puede además ayudarte a juzgar si el gradiente de descenso converge o no.

Por cierto, el número de iteraciones que el gradiente de descenso toma para converger en una aplicación física puede variar mucho. Así que tal vez, para una aplicación de gradiente de descenso puede converger después de solamente treinta iteraciones, y para una aplicación de gradiente de descenso diferente puede hacerlo a las 3,000 iteraciones. Para otro algoritmo de aprendizaje puede tardar tres millones de iteraciones. Resulta ser muy difícil saber de antemano cuantas iteraciones necesita el gradiente descendente para converger, y se logra por lo general tranzando este tipo de gráfica. Trazando la función de costo a medida que aumentamos el número de iteraciones. Por lo general al examinar estas gráficas intentamos saber si converge el gradiente de descenso. También es posible saber con la prueba automática de convergencia; es decir tener un algoritmo para intentar saber si el gradiente de descenso ha convergido y aquí está tal vez un ejemplo bastante típico de una prueba automática de convergencia:

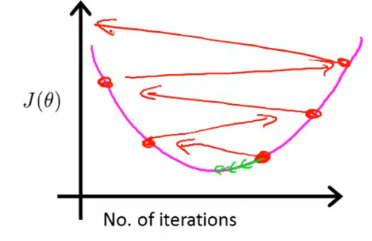


Así, si tu función de costo de "J" de «theta» disminuye en menos de un pequeño valor de «épsilon», algún pequeño valor de 10 a la -3 en una iteración, puede ser que ya haya convergido pero por lo general es bastante difícil elegir cual es este umbral. Así que, con el fin de comprobar que tu gradiente de descenso ha convergido.

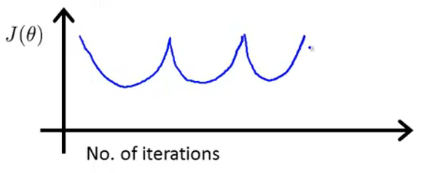
Por lo general observo las gráficas como esta figura de la derecha en lugar de confiar en una prueba de convergencia automática. Observar esta clase de figura también puede decirte o darte una advertencia por adelantado si tal vez el gradiente de descenso no está funcionando correctamente. Específicamente, si aplicas "J" de «theta» como una función de un número de iteraciones, entonces, si ves una figura como esta:



en donde "J" de «theta» está en realidad aumentando quiere decir que algo no está funcionando y puede ser que el valor de aprendizaje sea demasiado alto como vimos anteriormente:

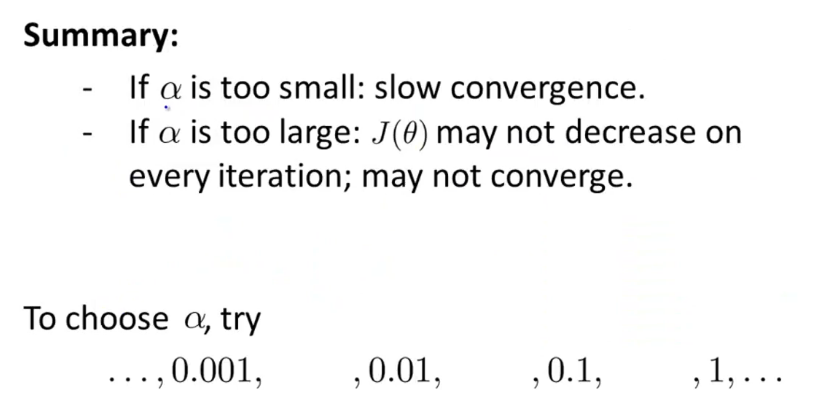


En algunas ocasiones también podemos encontrarnos este tipo de casos:



Pero también la solución pasa por lo mismo.

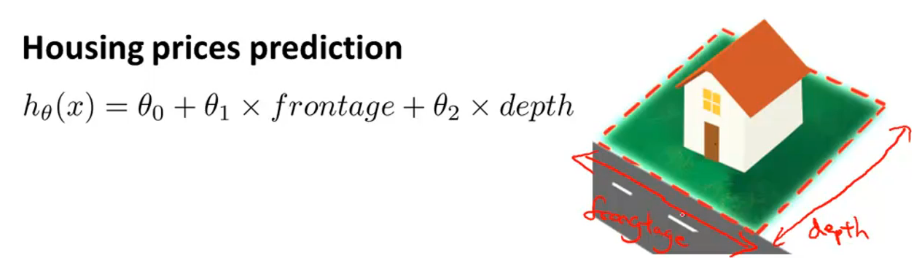
Recuerda que si tenemos una tasa de aprendizaje demasiado pequeña el gradiente convergerá muy lentamente.



## Características y regresión polinomial

Ahora sabes sobre regresión lineal con múltiples variables. En esta sección, vamos a hablar un poco acerca de la elección de variables y cómo puedes obtener diferentes algoritmos de aprendizaje, a veces muy poderosos eligiendo las variables apropiadas. Y, de manera particular también hay que saber que la regresión polinomial te permite utilizar la maquinaria de la regresión lineal para ajustar funciones muy complicadas, incluso no lineales. Tomemos el ejemplo de predecir el precio de una casa.

Supongamos que tienes dos variables, la fachada de la casa y la profundidad de la casa. Por lo que, aquí está la imagen de la casa que estamos tratando de vender. Así que, la fachada se define como el ancho o la longitud del ancho de tu casa, y la profundidad de la casa se refiere a que tan profunda es tu propiedad, así que hay una fachada, hay una profundidad. Así tienes dos variables llamadas fachada (frotage) y profundidad (depth).



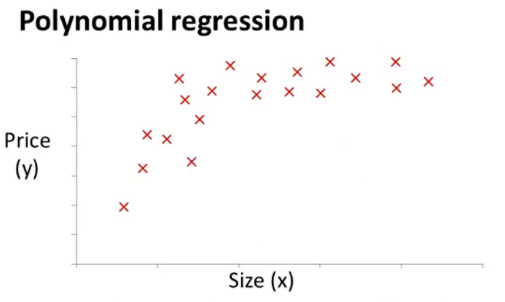
Es posible construir un modelo de regresión lineal como este en donde la fachada es tu primera variable x1 y la profundidad es tu segunda variable x2, pero cuando estás aplicando la regresión lineal, no necesariamente tienes que usar solamente las variables x1 y x2 que te dan. Lo que puedes hacer es crear nuevas variables por ti mismo.

Así, si quiero predecir el precio de una casa, lo que podría hacer en su lugar es decidir que lo que realmente determina el tamaño de la casa es el área. Así, podría crear una nueva variable, a la que puedo llamar "x" que es la fachada, multiplicada por la profundidad.

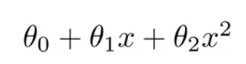
Y entonces puedo seleccionar mi hipótesis así, utilizando solamente una variable que es el área de mi tierra, ya que el área de un rectángulo es como sabes, el producto de la longitud de sus lados.

Así es que, dependiendo del entendimiento que puedas tener sobre un problema particular, en lugar de simplemente tomar las variables fachada y profundidad que son las que nos han dado para comenzar, a veces mediante la definición de nuevas variables en realidad podrías conseguir un mejor modelo.

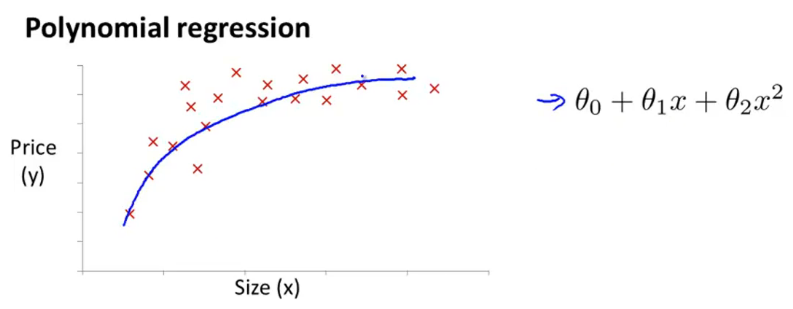
Estrechamente relacionada con la idea de elegir tus variables está la idea llamada regresión polinomial. Digamos que tienes un conjunto de datos de precios de vivienda que tienen este aspecto:



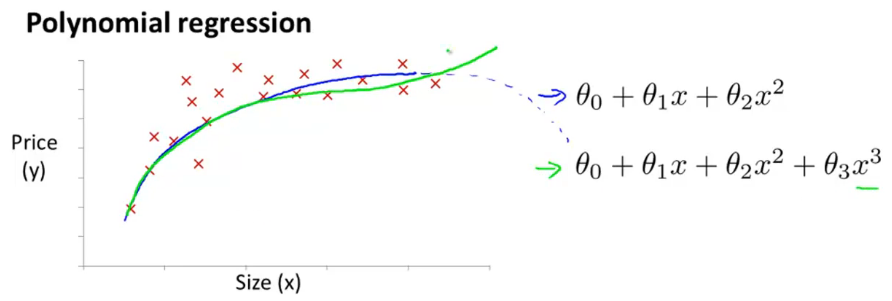
Entonces hay algunos modelos diferentes que podrías ajustar a esto. Una cosa que podrías hacer es ajustar un modelo cuadrático así:



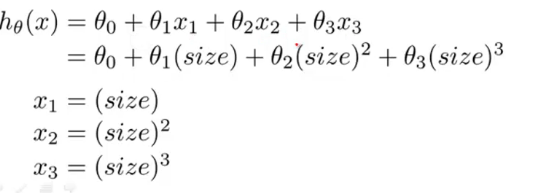
No parece que una línea recta se ajuste muy bien a estos datos. Así que tal vez quieras ajustar un modelo cuadrático como éste, en donde piensas que el precio es una función cuadrática que depende del tamaño y tal vez eso te puede dar un ajuste a los datos mejor:



Pero entonces puedes decidir que tu modelo cuadrático no tiene sentido con una función cuadrática, porque eventualmente esta función vuelve a bajar y bien, no creemos que los precios de vivienda deban bajar mientras que el tamaño sube tan alto. Entonces tal vez podamos elegir un modelo polinomial diferente y optar por utilizar en su lugar una función cúbica, en donde tenemos ahora un término de tercer orden y ajustamos eso, tal vez obtenemos este tipo de modelo:

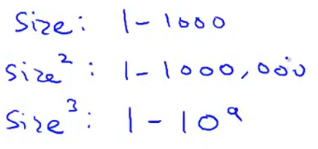


Y tal vez la línea verde se ajusta un poco mejor a los datos ya que no volverá a bajar eventualmente. Así que ¿cómo podemos ajustar un modelo como este a nuestros datos? Usando la maquinaria de la regresión lineal multivariante, podemos hacer esto con una modificación muy simple a nuestro algoritmo. La forma de la hipótesis, sabemos cómo el ajuste se parece a esto, en donde decimos que "h" de "x" es «theta»0 + «theta»1 x1 + «theta»2 x2 + «theta»3 x3. Y si queremos ajustar este modelo cúbico que he resaltado en verde, lo que estamos diciendo es que para predecir el precio de una casa, es «theta»0 + «theta»1 multiplicado por el tamaño de la casa + «theta»2 multiplicado por el tamaño de la casa al cuadrado y «theta»3 multiplicado por el tamaño de la casa al cubo:



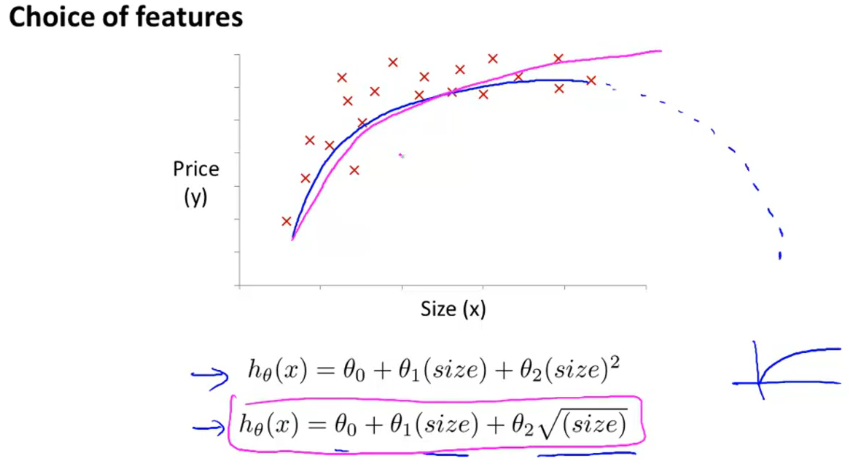
Para poder aplicar estas dos definiciones entre sí, la forma natural de hacerlo es establecer que la primer variable x1 será el tamaño de la casa, y establecer que la segunda variable x2 será el cuadrado del tamaño de la casa, y establecer que la tercera variable x3 será el cubo del tamaño de la casa. Y, solo por elegir mis tres variables de esta forma y aplicar la maquinaria de la regresión lineal puedo ajustar este modelo y terminar con un ajuste cúbico de mis datos.

Hay que señalar que si eliges tus variables de esta forma, entonces el escalamiento de variables se hace cada vez más importante. Así que si el tamaño de la casa está dentro del rango de uno a mil, entonces, como sabes, de uno a mil pies cuadrados, digamos, entonces el tamaño al cuadrado de la casa estará en el rango de uno a un millón, el cuadrado de mil, y tu tercera variable x3 que es el tamaño al cubo de la casa, estará en el rango de uno a diez a la novena potencia:



Y así estas tres variables adquieren muy diferentes rangos de valor, y es importante aplicar el escalamiento de variables si estás usando el gradiente de descenso para ponerlos en rangos de valores comparables.

Para terminar, aquí hay un último ejemplo de cómo tienes realmente amplias opciones en las funciones que utilizas. Anteriormente hablamos de cómo un modelo cuadrático como este podría no ser lo ideal porque tal vez un modelo cuadrático se ajusta mejor a los datos, pero la función cuadrática vuelve a bajar y realmente creemos que eso no es cierto ya que los precios de la vivienda siempre subirán conforme al tamaño. Pero en lugar de usar un modelo cúbico ahí, tienes, tal vez, otras opciones de variables y hay muchas opciones posibles. Pero sólo para darte otro ejemplo de una opción razonable, otra opción razonable podría ser decir que el precio de una casa es «theta»0 + «theta»1 multiplicado por el tamaño, y a continuación + «theta»2 multiplicado por la raíz cuadrada del tamaño, de modo que la función de la raíz cuadrada es este tipo de función, y tal vez habrá algún valor de «theta»1, «theta»2, «theta»3, que te permitirá tomar este modelo:

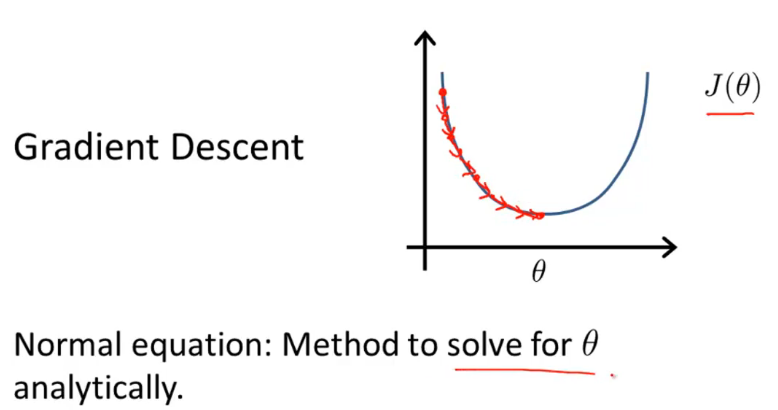


Y para la curva sube siempre, pero llega un momento donde se aplana un poco y nunca vuelve a bajar. Y, por lo que, al tener conocimiento de, en este caso, la forma de la función de la raíz cuadrada, y, de la forma de los datos, al elegir diferentes variables, a veces puedes obtener mejores modelos.

En esta sección, hablamos de la regresión polinomial. Es decir, cómo ajustar un polinomio, como una función cuadrática, o una función cúbica, a tus datos. También te di esta idea, de que tienes la elección de qué variables usar, como en vez de utilizar la fachada y la profundidad de la casa, tal vez, puedes multiplicarlas juntas para obtener una variable que capture el área del terreno de una casa. En caso de que esto parezca un poco desconcertante, con todas esas opciones de variables diferente, ¿cómo decido qué variables utilizar? Más adelante en esta clase, hablaremos sobre algunos algoritmos que automáticamente eligen qué variables son utilizadas, para que puedas tener una visión algorítmica de los datos y selecciona te selecciona automáticamente si deseas ajustar una función cuadrática, o una función cúbica, o algo más. Pero, hasta que lleguemos esos algoritmos, ahora solo quiero que estés consciente de que tienes opciones en cuanto a qué variables utilizar, y mediante el diseño de diferentes variables puedes ajustar funciones más complejas a tus datos, solamente ajustando una línea recta a los datos y en particular puedes poner funciones polinomiales también y a veces con el conocimiento apropiado de la variable te permite obtener un modelo mucho mejor para tus datos.

## Ecuación normal

En esta sección hablaremos acerca de la ecuación normal, la que para algunos problemas de regresión lineal nos dará una mejor manera de resolver para el valor óptimo de los parámetros «theta». En concreto, el algoritmo que hasta ahora hemos usado para la regresión lineal es el gradiente de descenso para minimizar la función de costos J de «theta», usaríamos este algoritmo iterativo que requiere muchos pasos, múltiples iteraciones del de gradiente de descenso para converger con el mínimo global. En contraste, la ecuación normal nos daría un método para despejar «theta» de forma analítica para que en vez de tener que utilizar este algoritmo iterativo, podamos en cambio despejar el valor óptimo de «theta» de una sola vez, para que en prácticamente un paso obtengas el valor óptimo ahí mismo.

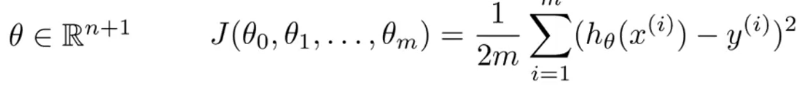


Resulta que la ecuación normal tiene algunas ventajas y algunas desventajas, pero antes de llegar a ellas y de hablar sobre cuándo deberías usarlas, vamos a obtener un poco de intuición sobre lo que hace este modelo. Para el ejemplo, imaginemos, una muy simplificada función de costos J de «theta», que es la función de un número real «theta».Por ahora imaginemos que «theta» es solo un valor escalar o que «theta» es solo un valor de fila. Es solo un número, en lugar de un vector. Imagina que tenemos una función de costo J que es una función cuadrática de un valor real que es parámetro «theta», de tal forma que J de «theta» se ve así:

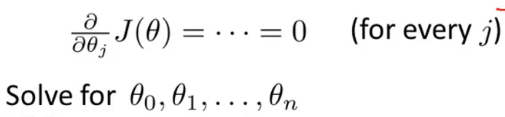


¿Cómo minimizas una función cuadrática? Para aquellos de ustedes que saben un poco de cálculo, sabrán que la forma de minimizar una función es tomando las derivadas e igualar las derivadas a cero. Así que sacas la derivada de J con respecto al parámetro de «theta». Obtienes una fórmula que no voy a derivar e igualas la derivada a cero, y esto te permite despejar el valor de «theta» que minimiza J de «theta». Ese era un caso más sencillo en el que los datos eran solo números reales.

En el problema que nos interesa, «theta» ya no es solamente un mero número real, sino que es un vector de parámetros de n+1 dimensiones y una función de costos J es una función de este valor del vector o de «theta»0 hasta «theta» m.

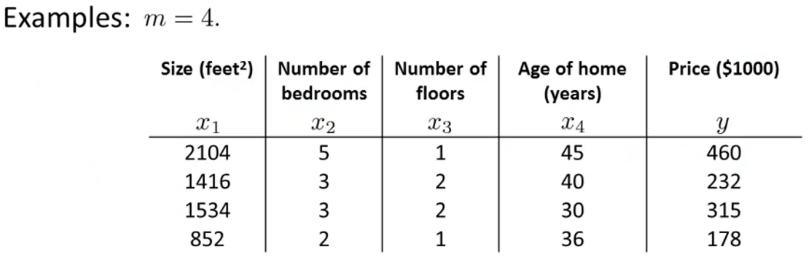


¿Cómo minimizamos la función de costos J? El cálculo nos dice que, una forma de hacerlo es tomando la derivada parcial de J con respecto a cada uno de los parámetros de «theta» J, y entonces igualarlos todos a 0. Si haces eso y resuelves para los valores de «theta»0, «theta»1, hasta «theta» n, entonces, tendrías los valores de «theta» para minimizar el costo de la función J:

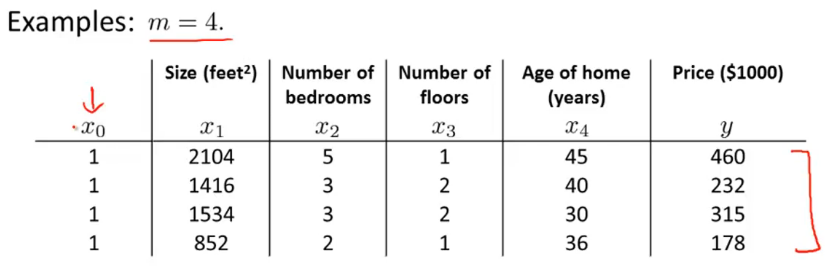


En cambio, si de verdad haces el cálculo para hallar la solución de los parámetros «theta»0 hasta «theta» n, la derivación se torna un tanto tediosa. Y, lo que voy a hacer en esta sección, en realidad es no pasar por la derivación, que es un poco larga y un poco tediosa, pero lo que quiero hacer solamente decirte lo que tienes que saber para que puedas implementar este proceso y puedas despejar los valores de «theta» que correspondan a los lugares en que las derivadas parciales sean igual a cero. O lo que equivale o en otras palabras, los valores de «theta» que minimizan la función de costos J de «theta».

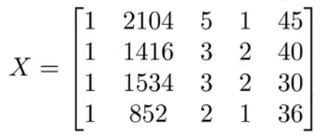
Para el ejemplo que quiero usar como ejemplo de funcionamiento digamos que tengo m= 4 ejemplos de entrenamiento:



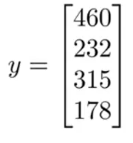
Para implementar esta ecuación normal a gran escala, haré lo siguiente. Voy a tomar el conjunto de datos, en este caso supongamos que estos cuatro ejemplos son todos los datos que tengo. Lo que haré es tomar mi conjunto de datos y añadir una columna adicional que corresponde a mi variable adicional, x0, que siempre toma el valor de 1:



Lo que haré es voy hacer es construir una matriz que se llame "X" que es una matriz que, básicamente, contiene todas las variables de mis datos de entrenamiento:



Aquí están todas mis variables, simplemente copio los datos una columna a la vez y luego le hago algo parecido a las "y", tomamos los valores que estoy tratando de predecir y construyo un vector así y lo llamo vector "y":



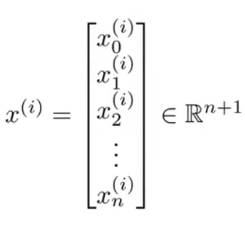
Así que "X" va a ser un matriz de dimensiones m multiplicado por (n+1) e será un vector de m dimensiones. Donde m es la cantidad de ejemplos de entrenamiento y n es, n es la cantidad de variables, n+1, a causa de la variable adicional x0 que tengo. Por último, si tomas tu matriz "X" y tomas tu vector "y", y si calculas esto y estableces que «theta» es igual a X transpuesta X inversa multiplicada por X transpuesta Y, esto te daría el valor de «theta» que minimiza tu función de costos:



Aún no queda totalmente claro cómo se hace esto. En un caso general, digamos que tenemos "m" ejemplos de entrenamiento por lo que x1, y1 hasta xm e ym y n variables:

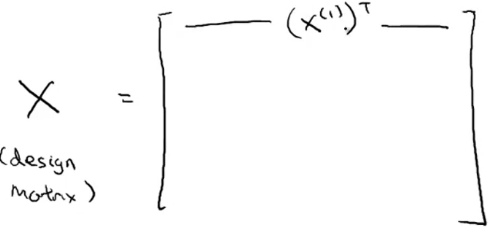


Cada uno de los ejemplos de entrenamiento x(i) parecerán vectores así:

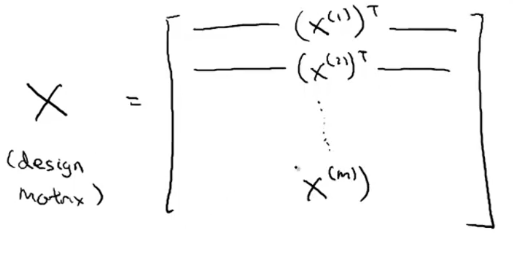


que es un vector de variables de n+1 dimensiones. La manera en que construiré la matriz "X", esto también es conocido como una matriz de diseño, es como sigue:

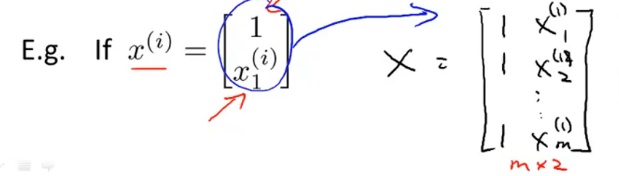
Cada ejemplo de entrenamiento me da un vector de variables como x(i), una especie de vector de n+1 dimensiones. La manera en que voy a construir mi matriz de diseño "X" es simplemente tomando el primer ejemplo de entrenamiento (eso es un vector), tomar su transpuesta y hacer de x1 transpuesta la primera fila de mi matriz de diseño:



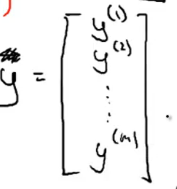
Después tomaré mi segundo ejemplo de entrenamiento, x2 transpuesta y lo pongo como la segunda fila de "X" y así sigo hasta llegar al último de mis ejemplos de entrenamiento:



Y por eso tomo mi matriz "X", una matriz de m multiplicado por n+1 dimensiones. Para ver un ejemplo concreto, digamos que tengo solo una variable además de x0, que siempre es igual a 1. Así que si mis vectores de variables xi son iguales a 1 (que es x0), y alguna variable real, como quizás sea el tamaño de la casa, entonces mi matriz de diseño "X", sería igual a:



Por lo que está será una matriz de m por 2 dimensiones. Así es como se construye la matriz "X". Y, el vector "y", a veces le dibujo una flecha en la parte superior para denotar que es un vector ( ), pero muy a menudo escribo solo "y", da igual. El vector "y" se obtiene al tomar todos los valores, todos los precios correctos de las casas en mi conjunto de entrenamiento, y ponerlas unas sobre otras en un vector de m dimensiones, y eso es "y":



Por fin, después de haber construido la matriz "X" y el vector "y", calculamos «theta» como:



¿Cómo se implementa este desarrollo?

Específicamente, si fueran que el conjunto A = entonces , y sabemos que eso sería el equivalente a por lo tanto es solo tomar la matriz "A" e invertirla.

Para calcular en Octave



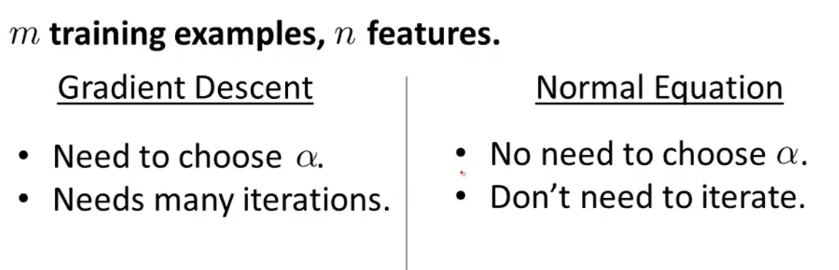
Hacemos:



pinv es una función para calcular la inversa de una matriz, así que esto calcula X transpuesta X inversa, y después multiplicas eso por X transpuesta y multiplicas eso por "y". Y terminas calculando la fórmula. Esta fórmula te da el valor óptimo de «theta» en el sentido de que si igualas «theta» a esto, ese es el valor de «theta» que minimiza la función de costo J de «theta» para la nueva regresión.

Un último detalle del video anterior. Hablé de la habilidad de variables y de la idea de hacer que las variables estén dentro de rangos similares de escalas, de rangos similares de valores parecidos entre sí. Si usas este método de la ecuación normal, entonces el escalamiento de variables no es del todo necesario y, de hecho, está bien si, digamos, alguna variable x1 está entre 0 y 1, y otra variable x2 está entre 0 y 1000 y otra variable x3 está entre 0 y 10 a la menos 5 y si usas el método de la ecuación normal esto está bien y no hay necesidad de aplicar el escalamiento de variables, aunque, por supuesto, si estas usando el gradiente de descenso, entonces, el ajuste de las escalas sigue siendo importante.

Por último, dónde deberías usar el gradiente de descenso y dónde el método de la ecuación normal. Aquí presento algunas de sus ventajas y desventajas. Digamos que tienes "m" ejemplos de entrenamiento y "n" variables. Una desventaja del gradiente de descenso es que necesitas escoger la tasa de aprendizaje «alfa». Y, con frecuencia, esto significa tener que ejecutarlo varias veces con diferentes tasas de aprendizaje «alfa» para ver cuál funciona mejor. Y eso es trabajo adicional y más complicaciones. Otra desventaja del gradiente de descenso es que requiere muchas más iteraciones. Por lo que, depende de los detalles, podría hacerlo más lento, aunque ese no es el fin de la historia como veremos en un momento. En cuanto a la ecuación normal, no tienes que escoger ninguna tasa de aprendizaje «alfa». Por eso, como ves, es muy conveniente, hace que sea fácil de implementar. Simplemente la usas y por lo general funciona. Y no tienes que iterar, así que no tienes que trazar J de «theta» o revisar la convergencia ni realizar esos pasos adicionales Hasta aquí, todo parece favorecer la ecuación normal.

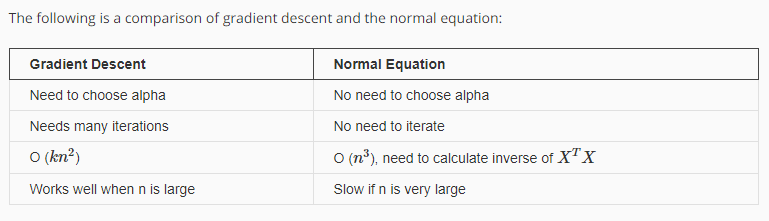


Ahora veremos algunas desventajas de la ecuación normal y algunas ventajas del gradiente de descenso. El gradiente de descenso funciona bastante bien, aun cuando tengas una gran cantidad de variables. Aunque tengas millones de variables puedes usar el gradiente de descenso y será razonablemente eficiente. Hará algo razonable. En contraste con la ecuación normal que para despejar los datos de los parámetros tenemos que despejar este término. Tenemos que calcular este término Esta matriz es una matriz n\*n donde "n" es la cantidad de variables y en la mayoría de las implementaciones el costo de invertir la matriz, aumenta aproximadamente como el cubo de la dimensión de la matriz . Pues bien, calcular esta inversa tiene un costo de orden y tiempo cúbicos. A veces es más rápido que "n" al cubo pero, es una buena aproximación para nuestros propósitos. Luego si "n", el número de variables, y este es muy grande calcular esta cifra puede ser lento y el método de la ecuación normal puede resultar mucho más lento. Por lo que sí "n" es grande, entonces quizás sería mejor utilizar el gradiente de descenso. Pero, si "n" es relativamente pequeña, entonces la ecuación normal pudiera darte una mejor manera de resolver los parámetros.

¿Qué es pequeño y qué es grande? Bueno, si "n" es del orden de cien, entonces invertir una matriz cien por cien no es problema para los estándares computacionales modernos. Si "n" es mil, yo todavía utilizaría el método de la ecuación normal. Invertir una matriz mil por mil es en realidad muy rápido en una computadora moderna. Si "n" es diez mil, empezaría a planteármelo. Invertir una matriz diez mil por diez mil ya empieza a ser un tanto lento y quizás comience a inclinarme hacia el gradiente de descenso, pero quizás no del todo. Si "n" es diez mil, tú puedes más o menos convertir una matriz diez mil por diez mil. Pero si es mucho más grande que eso, entonces, es probable que use el gradiente de descenso. Así que si "n" es igual a diez elevado a la seis (un millón de variables), entonces invertir una matriz de un millón por un millón será muy costoso y de seguro yo favorecería el gradiente de descenso si tienes esa cantidad de variables.

Exactamente cuán grande tiene que ser un conjunto de variables para convertirlo a gradiente de descenso. Es difícil dar un número exacto. Pero, para mí suele ser alrededor de diez mil cuando comienzo a considerar el cambio a gradientes de descenso o quizás, a otros de los algoritmos que veremos después en secciones siguientes.

En resumen, siempre y cuando la cantidad de variables no sea demasiado grande, la ecuación normal nos da un método alternativo para resolver para los parámetros de «theta». En concreto, siempre y cuando el número de variables sea menor que 1000, bien, yo usaría, se suele usar el método de la ecuación normal en lugar del gradiente de descenso. Para adelantar algunas ideas que veremos más adelante en el curso a medida que lleguemos al algoritmo de aprendizaje más complejo, cuando hablemos, por ejemplo, de un algoritmo de clasificación, como un algoritmo de regresión logística. Veremos que esos algoritmos en realidad… El método de la ecuación normal en realidad no funciona para esos algoritmos de aprendizaje más sofisticados y tendremos que recurrir al gradiente de descenso para esos algoritmos. Es útil conocer el algoritmo del gradiente de descenso tanto para la regresión lineal donde tenemos una gran cantidad de variables y como para algunos de los otros algoritmos que veremos en este curso porque el método de la ecuación normal simplemente no les aplica o no funciona. Pero para este modelo específico de regresión lineal la ecuación normal puede darte una alternativa que puede ser mucho más rápida que el gradiente descendiente. Entonces, depende de tu algoritmo, depende de los detalles de los problemas y de cuántas variables tengas. Vale la pena conocer estos dos algoritmos.



### Matrices no invertibles

Hay un fenómeno con el que puedes tropezarte ¿y si la matriz X transpuesta X es no invertible? Como sabemos, existen matrices que no son invertibles (no tienen una inversa), a esas les llamamos matrices no invertibles, matrices singulares o degeneradas.

El tema o el problema de que X traspone X sea no invertible debería ocurrir muy raramente. Y en Octave, si se implementa esto para calcular «theta», resulta que en realidad hará lo correcto. Octave tiene dos funciones para invertir matrices:

* pinv()
* inv().

Las diferencias entre ambas son algo un poco técnico. Una se llama la pseudoinversa, la otra se llama la inversa. Puedes demostrarlo matemáticamente mientras se utiliza la función pinv(), entonces esto va a calcular el valor de «theta2 que deseas, incluso si (X transpuesta X) es no invertible. Los detalles específicos sobre cuáles son las diferencias entre pinv() y lo que es inv() son conceptos de cálculo numérico algo avanzados, a los que realmente no quiero entrar. Pero creo que es mejor dar un poco de intuición sobre lo que significa que una matriz no tenga inversa. No voy a probar esto matemáticamente, pero si X transpuesta X es no invertible, hay por lo general dos causas comunes:

1. **Multicolinealidad**. La primera causa es que si de alguna manera, en tu problema de aprendizaje, tienes características redundantes, específicamente, si intentas predecir los precios de la vivienda y si x1 es el tamaño de una casa en pies cuadrados y x2 es el tamaño de la casa en metros cuadrados entonces, como sabes, 1 metro es igual a 3.28 pies, redondeando a dos decimales, y así tus dos características siempre satisfacen la restricción de que x1 es igual a (3.28) multiplicado por x2. Y puedes demostrar, para algunos de ustedes - esto es álgebra lineal un poco avanzada-, ahora, pero si eres experto en álgebra lineal, en realidad puedes demostrar que si tus dos características están relacionadas mediante una ecuación lineal como esta, la matriz X transpuesta X será no invertible.
2. Lo segundo que puede causar que sea no invertible es si estás tratando de ejecutar un algoritmo de aprendizaje con un lote de características. Específicamente, si "m" es menor que o igual a "n". Por ejemplo, si imaginas que tienes "m" igual a 10 ejemplos de entrenamiento y que tienes "n" igual a 100 características, entonces estás tratando de ajustar un parámetro del vector «theta», el cual es de dimensión (n+1), así que si es de dimensiones uno a uno estás intentando ajustar parámetros uno a uno para sólo 10 ejemplos de entrenamiento. Y esto resulta funcionar a veces, pero no siempre es una buena idea. Porque, como veremos más adelante, puedes no tener datos suficientes si sólo tienes 10 ejemplos para ajustar 100 o 101 parámetros.

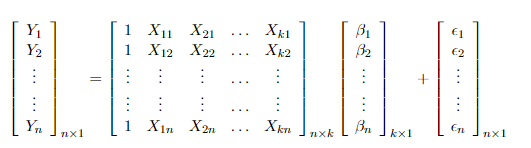
Veremos más adelante en este curso, que estos pueden ser muy pocos datos para ajustar tantos parámetros. Pero por lo general, lo que hacemos entonces si "m" es menor que "n", es ver si podemos eliminar algunas de las características o bien usar una técnica llamada regularización, que es algo de lo que también vamos a hablar un poco más adelante en este curso, que te permitirá ajustar un lote de parámetros utilizando un lote de características incluso si tienes un conjunto de entrenamiento relativamente pequeño.

Pero esta regularización será un tema más adelante en este curso. Para resumir, si alguna vez te encuentras con que X transpuesta X es singular o alternativamente es no invertible, lo que recomiendo que hagas es primero: observar tus características y ver si tienes características redundantes como estas x1 y x2 siendo linealmente dependientes, siendo una función lineal una de la otra, y si sólo eliminas una de estas características (realmente no necesitas ambas características), así que si simplemente eliminas una de estas características eso resolverá tu problema de no invertibilidad- así es que, primero analizo mis características y compruebo si alguna es redundante y si las hay, entonces sigo eliminando las características redundantes hasta que ya no son redundantes. Y si tus características no son redundantes, comprobaría si podría tener demasiadas características, y si ese es el caso, ya sea que elimine algunas características si puedo simplemente utilizar menos características, o de lo contrario consideraría usar la regularización, que es el tema del que vamos a hablar más adelante.

Por lo tanto, eso es todo sobre la ecuación normal y lo que significa que la matriz X transpuesta X sea no invertible. Pero esto es un problema que espero que te encuentres muy rara vez. Y si solo lo implementas en Octave utilizando la función pinv() que se llama la función pseudoinversa por lo que utilizas una librería de álgebra lineal diferente, eso se llama pseudoinversa, pero esa implementación debe hacer solo lo correcto incluso si X transpuesta X no es invertible lo que de todos modos debería suceder muy rara vez por lo que esto no debería ser un problema para la mayoría de las implementaciones de regresión lineal.

## (Apéndice) Derivando el modelo matricialmente

Tenemos nuestra variable para regresar sobre un conjunto de características. EL ajuste evidentemente no será perfecto y debemos dejar un parámetro adicional llamado error que toma los valores de la diferencia entre el valor estimado mediante nuestros parámetros y el valor real.



Esto puede escribirse matricialmente como:



Como hemos dicho, el error no es más que la diferencia entre nuestra variable observada (la real) y la estimada que predigo a multiplicando las características por los diferentes pesos de los parámetros por lo que si tengo una casa que vale 90000 euros y un modelo que me dice que el valor de la casa viene dado por

La misma casa tiene 150 metros cuadrados y 3 habitaciones, por lo que su valor estimado será de:

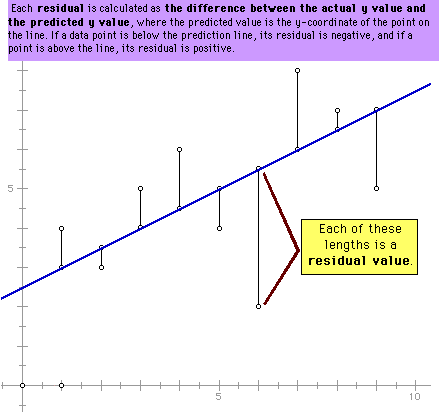
Que es igual a 82500 mi error será

El vector de los residuos también se puede expresar como:

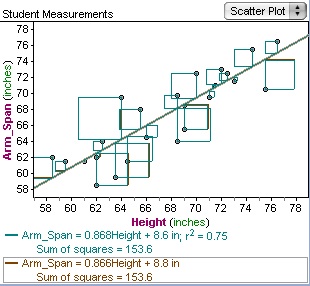


Por lo tanto, a lo que nosotros nos interesa es hacer que esos residuos sean lo más pequeños posibles. Y como esos errores pueden ser tanto positivos (el valor predicho es menor que el real) o negativos (el valor predicho es mayor que el real), debemos ponerlos al cuadrado para evitar que se cancelen ya que dos errores uno igual a 7500 y otro igual a -7500 son iguales a 0 y no sería correcto.

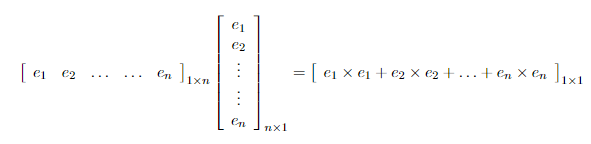
Ejemplo de residuos normales:



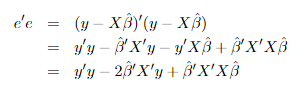
Ejemplos de residuos al cuadrado:



Por lo tanto, lo que queremos hacer que es obtener aquella combinaciones de parámetros que hagan esos errores al cuadrado lo más pequeños posibles. La suma de los errores al cuadrado se puede denotar de la siguiente manera.

= 

Y sustituyendo, podemos llegar a:



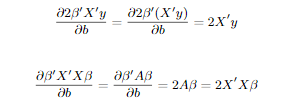
\*Recuerda que (AB)’ = B’A’.

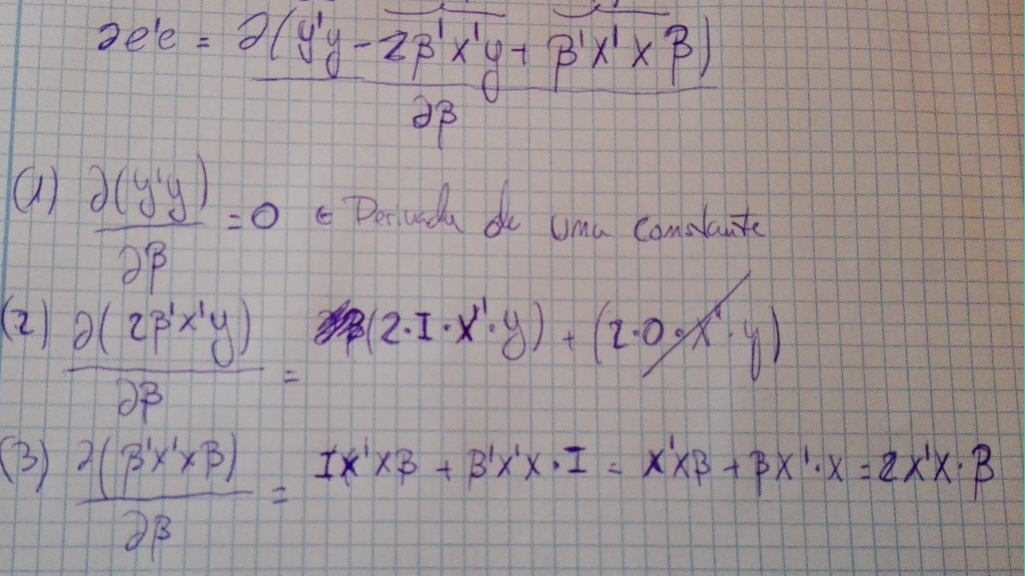
El siguiente paso, como todo problema de optimización que se precie es igualar la derivada de una función a cero para obtener sus mínimos. Por lo tanto, se deriva la función anterior respecto de β para obtener la siguiente ecuación:

Recordando las siguientes reglas de derivación de matrices:











Despejando los términos en la ecuación:



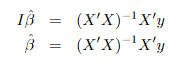
El siguiente paso es dejar la B sola, para ello en una ecuacuón con números enteros deberíamos pasar el término que está multiplicándolo dividiendo o lo que es lo mismo, dividir en un lado por ese término y multiplicar en el otro.



Sabiendo que:



Ya que I es equivalente a 1 en los números reales quedaría:



Sabiendo que

X = y X’ =

Donde n = nº de variables y m = nº de sujetos.

Ejemplo concreto con 3 sujetos y 3 variables:

X = y X’ =

X\*X’ =