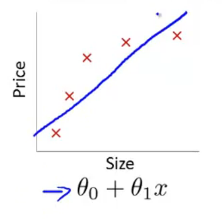
# Regularización

Para este momento, ya has visto un par de algoritmos de aprendizaje automático diferentes, regresión lineal y regresión logística. Estos, funcionan bien para muchos problemas, pero cuando se aplican pueden caer en un problema llamado sobreajuste, que causa que tengan un desempeño muy malo.

Lo que me gustaría hacer en este capítulo es explicarte lo que es este problema de sobreajuste, y durante algunos de las siguientes secciones, vamos a hablar de una técnica llamada regularización, que nos permitirá mejorar o reducir este problema de sobreajuste y lograr que estos algoritmos de aprendizaje funcionen mucho mejor.

## El problema del sobreajuste

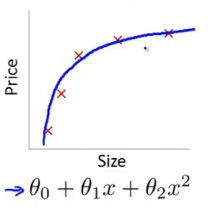
¿Qué es sobreajuste? Sigamos utilizando nuestro ejemplo de predicción de precios de vivienda con regresión lineal donde queremos predecir el precio en función del tamaño de la casa. Una cosa que podríamos hacer es ajustar una función lineal a estos datos, y si hacemos eso, probablemente obtengamos ese tipo de línea recta que se ajusta a los datos:



Pero este no parece es un buen modelo. Observando los datos, parece bastante claro que conforme el tamaño de la vivienda incrementa, vemos que llega un momento que no le afecta tanto al precio como le afectaba para valores más pequeños, es decir los precios de la vivienda, se aplana conforme nos movemos hacia la derecha por lo que este algoritmo no se ajusta al entrenamiento y llamamos a este problema **subajuste**, y otro término para esto es que este algoritmo tiene una alta oscilación (“***high bias***”). Ambos casos más o menos quieren decir que ni siquiera se están ajustando bien los datos.

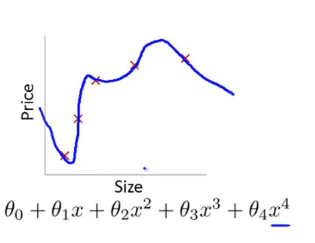
Oscilación o *bias* es un término histórico o técnico, pero la idea es que si ajustamos los datos con una línea recta, entonces es como si el algoritmo tuviera una preconcepción muy fuerte de que los precios de la vivienda van a variar linealmente con el tamaño de las misma a pesar de que los datos indiquen lo contrario.

Ahora, en el medio, podríamos ajustar una función cuadrática a los datos, con este conjunto de datos, ajustamos la función cuadrática, y tal vez obtengamos este tipo de curva:



Y, eso funciona bastante bien.

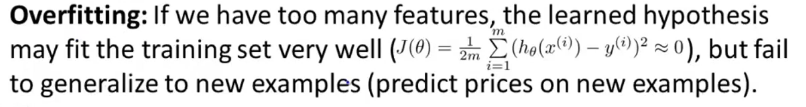
Por último, el otro extremo, sería si fuéramos a ajustar, digamos, un polinomio de grado 4 a los datos. Así que, aquí tenemos cinco parámetros, de «theta» 0 a «theta» 4, y, con eso, en realidad podemos ajustar una curva que se procese a través de nuestros cinco ejemplos de entrenamiento. Podrías obtener una curva que se vea así:



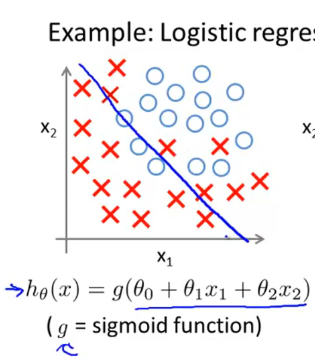
Que, por un lado, parece hacer un buen trabajo ajustando el conjunto de entrenamiento y, se procesa a través de todos mis datos. Pero esto ahora es una curva muy ondulada, y, en realidad no creemos que sea un buen modelo para predecir el precio de la vivienda. Así es que llamamos a este problema sobreajuste ***(“overfitting”),*** y otro término para esto es que este algoritmo tiene varianza alta ***(“High variance”).*** El término varianza alta también es histórico o técnico. Pero la intuición es que, si estamos ajustando un polinomio de alto grado, entonces, la hipótesis se puede ajustar perfecta o casi perfectamente, como sabes, es casi como si pudiera ajustarse a casi cualquier función y estos tipos de hipótesis son demasiado variables. Y no tenemos suficientes datos para restringirla para darnos una buena hipótesis, por lo que le llamamos sobreajuste. Para el caso del medio no hay realmente un nombre pero solo voy a escribir, “*just righ*t”. Un polinomio de segundo grado o función cuadrática parece ser lo adecuado para el ajuste de estos datos.

Recapitulando un poco:

***“Overfitting” o “high variance”***: el problema de sobreajuste se da cuando tenemos demasiadas variables, entonces la hipótesis de aprendizaje se puede ajustar muy bien a los datos de entrenamiento. Así, tu función de costes en realidad puede estar muy cerca de cero o puede ser incluso exactamente cero, pero podrías terminar con una curva como la anterior que, como sabes, se esfuerza demasiado para ajustarse al conjunto de entrenamiento, así que cuando reciba nuevos datos que están fuera del conjunto de entrenamiento seguramente estarán muy lejos de la línea que hemos dibujado.

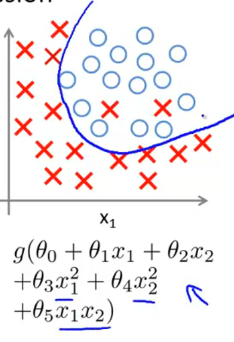


Así que es incapaz de generalizar nuevos ejemplos y falla en predecir los precios de los nuevos ejemplos, y aquí el término generalizar se refiere a qué tan bien se aplica una hipótesis incluso a nuevos ejemplos. Esto es para los datos de las casas que no se han visto en el conjunto de entrenamiento. En esta diapositiva, observamos el sobreajuste para el caso de la regresión lineal. Algo similar puede aplicarse a la regresión logística también. Aquí hay un ejemplo de regresión logística: con dos variables x1 y x2. Algo que podemos hacer, es ajustar la regresión logística con una hipótesis simple en donde, como de costumbre, "g" es mi función sigmoidea. Y si lo haces, terminas con una hipótesis, tratando de utilizar, una línea recta para separar las clases:



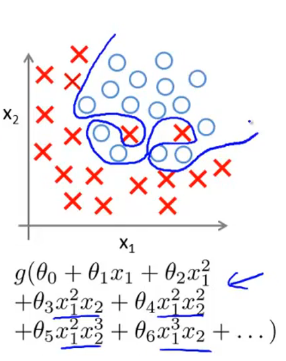
Y esto no parece un muy buen ajuste para la hipótesis. Así que, una vez más, esto es un ejemplo de subajuste o “high bias”.

En contraste, si vas a agregar a tus variables términos cuadráticos, entonces, podrías obtener una frontera de decisión que podría parecerse más a esto:



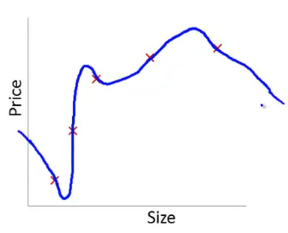
Y, como sabes, es un buen ajuste a los datos. Probablemente, lo mejor que podemos conseguir, en este conjunto de entrenamiento.

Y, finalmente, en el otro extremo, si fueras a ajustar un polinomio de muy alto grado, si fueras a generar muchos términos polinomiales de alto grado, entonces, la regresión logística puede retorcerse, puede intentar demasiado encontrar una frontera de decisión que se ajuste a tus datos de entrenamiento para ajustarse a cada ejemplo de entrenamiento individual:



Y, como sabes, si las variables x1 y x2 ofrecen predicción, tal vez, del tipo de cáncer maligno o benigno. Esto no parece una hipótesis muy buena para hacer predicciones. Y así, una vez más, esto es una instancia de sobreajuste o varianza alta y en realidad no, y, siendo poco probable que generalice bien los nuevos ejemplos.

Más adelante, en este curso, cuando hablemos sobre la depuración (o debugging) y el diagnóstico de lo que pueden salir mal con los algoritmos de aprendizaje, te daremos herramientas específicas para reconocer cuándo el sobreajuste y también el subajuste puedan estar ocurriendo. Pero, por ahora, vamos a hablar sobre el problema, cuando pensamos que se está produciendo sobreajuste, ¿Qué podemos hacer para abordarlo? En los ejemplos anteriores, hemos tenido datos de una o dos dimensiones, así que podríamos dibujar las hipótesis y ver lo que está pasando o seleccionar el polinomio de grado apropiado. Así que, antes para el ejemplo de los precios de la vivienda, podríamos simplemente trazar la hipótesis y, como sabes, tal vez al ver que se ajusta a un tipo de función muy ondulada que va por todas partes para predecir el precio de la vivienda:



Así es que dibujar la hipótesis, podría ser una manera de intentar decidir qué polinomio de grado utilizar. Pero eso no siempre funciona. Y, de hecho más a menudo podríamos tener problemas de aprendizaje en donde tenemos un montón de variables. Cuando tenemos tantas variables, también se vuelve mucho más difícil dibujar los datos y se hace mucho más difícil visualizarlos, para decidir qué variables mantener o no. Así que volviendo al ejemplo anterior, si estamos tratando de predecir los precios de la vivienda, a veces podemos tener un montón de variables diferentes. Y todas estas variables, como sabes, podrían parecer útiles. Pero, si tenemos muchas variables, y muy pocos datos de entrenamiento, entonces, el sobreajuste puede convertirse en un problema.

Con el fin de abordar el sobreajuste, tenemos dos opciones principales que podemos realizar.

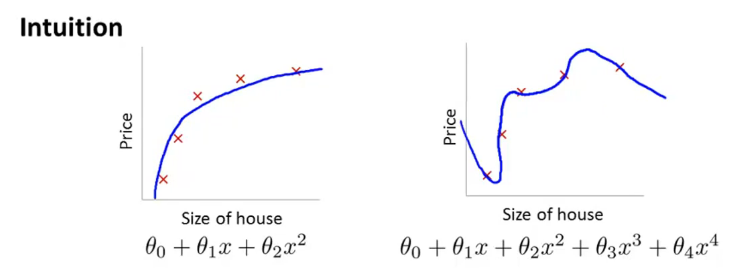
1. La primera opción es tratar de reducir el número de variables. Específicamente, una cosa que podemos hacer es revisar manualmente la lista de variables, y, usar eso para tratar de decidir cuáles son las variables más importantes, y, por lo tanto cuáles son las variables que deberíamos conservar, y cuáles son las variables que deberíamos desechar. Más adelante en este curso, también vamos a hablar de los algoritmos de selección de modelo, los cuales son algoritmos para decidir automáticamente cuales variables conservar y cuales variables desechar. Esta idea de reducir el número de variables puede funcionar bien, y puede reducir el sobreajuste. Y, cuando hablemos de selección de modelos, tocaremos este tema con mucha mayor profundidad. Pero, la desventaja es que, al desechar algunas variables, también estamos eliminando parte de la información que tenemos sobre un problema. Por ejemplo, tal vez, todas esas variables son realmente útiles para predecir el precio de una casa, entonces, tal vez no queremos deshacernos de parte de nuestra información o eliminar algunas de nuestras variables.
2. La segunda opción, de la que vamos a hablar en algunos de los siguientes videos, es la regularización. Con ella vamos a mantener todas las variables, pero vamos a reducir la magnitud o el valor de los parámetros .Y este método funciona bien, como vamos a ver, cuando tenemos muchas variables, y cada una de ellas contribuye un poco a predecir el valor de , como vimos en el ejemplo de predicción del precio de la vivienda, en donde podemos tener un montón de variables, cada una de las cuales son, como sabes, útiles de alguna manera, por lo que tal vez, no queremos deshacernos de ellas. Así que esto suscribe la idea de la regularización en un nivel muy alto. Y, me doy cuenta de que todos estos detalles probablemente no tienen sentido para ti todavía. Pero en el siguiente sección, vamos a empezar a formular exactamente cómo aplicar la regularización y, exactamente lo que la regularización significa. Y, entonces empezarás a descubrir, cómo utilizar esto, para lograr que los algoritmos de aprendizaje funcionen bien y así evitar el sobreajuste.

## Regularización

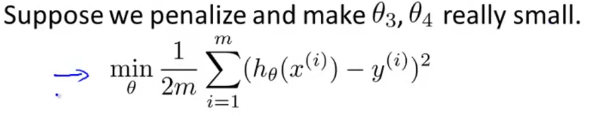
En este sección, me gustaría trasmitirte las intuiciones principales detrás de cómo funciona la regularización. Y, también escribiremos la función de costos que usaremos, cuando utilicemos la regularización.

Con los ejemplos dibujados a mano que tenemos en estas diapositivas, creo que seré capaz de transmitirte parte de la intuición. Pero, incluso una mejor forma de ver tú mismo cómo funciona la regularización, es si la implementas, y ves cómo funciona. Y si haces los ejercicios apropiados después de esto, tendrás la oportunidad de ver tú mismo la regularización en acción. Entonces, aquí está la intuición.

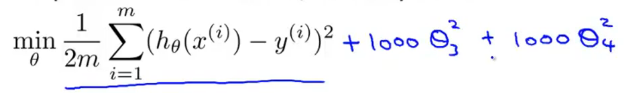
En la sección anterior, vimos que, ajustar una función cuadrática a los datos, nos daba un buen ajuste. Considerando que, si fuéramos a ajustar un polinomio de muy alto grado, terminaríamos con una curva que podría ajustarse al conjunto de entrenamiento bastante bien, pero que en realidad pero que sobreajustará los datos, y no generalizará bien.



Considera lo siguiente, supongamos que fuéramos a penalizar, y hacer los parámetros «theta»3 y «theta»4 muy pequeños. Esto es lo que quiero decir, aquí esta nuestro objetivo de optimización, o aquí está nuestro problema de optimización, en donde minimizamos nuestro error al cuadrado usual que causa la función:

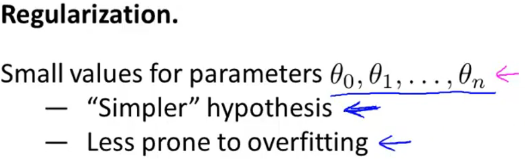


Digamos que tomo este objetivo y lo modifico y le añado, + 1000 «theta»3 al cuadrado, + 1000 «theta»4 al cuadrado. Solamente estoy escribiendo 1000 como ejemplo de un número muy grande.

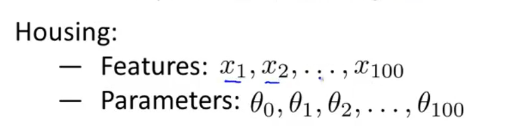


Ahora, si fuéramos a minimizar esta función, la única manera de hacer esta nueva función de costos más pequeña es si «theta»3 y «theta»4 son pequeñas, porque de otra forma, si tienes mil veces «theta»3, esta nueva función de costos va a ser grande. Así que cuando minimizamos esta nueva función vamos a terminar con el valor de «theta»3 cercano a 0 y de «theta»4 cercano a 0, como si nos deshiciéramos de estos dos términos de ahí. Y si hacemos eso, «theta»3 y «theta»4 están cerca de 0 nos estamos quedando con una función cuadrática, más tal vez, pequeñas contribuciones de términos pequeños, «theta»3, «theta»4, que pueden estar muy cerca de 0. Y, así, terminamos con en esencia, una función cuadrática, lo que es bueno, debido a que esta es una mucho mejor hipótesis.

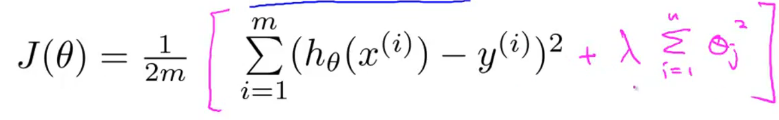
En general, esta es la idea detrás de la regularización. La idea es que, si tenemos valores pequeños para los parámetros de alguna manera, corresponderá generalmente con tener una hipótesis más simple. Así que, para nuestro último ejemplo, penalizaremos únicamente «theta»3 y «theta»4 y cuando ambos estén cerca de cero, terminaremos con una hipótesis mucho más simple que en esencia es bastante similar a una función cuadrática. Pero en términos más generales, si penalizamos a todos los parámetros es posible demostrar que al tener valores más pequeños de los parámetros generalmente corresponde a funciones más suaves así como más simples. Y por lo tanto, también son menos propensas al sobreajuste.



Me doy cuenta de que el razonamiento de por qué todos los parámetros deben ser pequeños corresponde a una hipótesis más simple; me doy cuenta de que ese razonamiento puede no ser totalmente claro para ti ahora. Y es difícil explicarlo a menos de que lo implementes y lo compruebes tú mismo. Pero espero que el ejemplo de reducir «theta»3 y «theta»4 ayude a explicar por qué, o por lo menos a darte una intuición en cuanto a por qué esto podría ser cierto. Veamos el ejemplo específico:



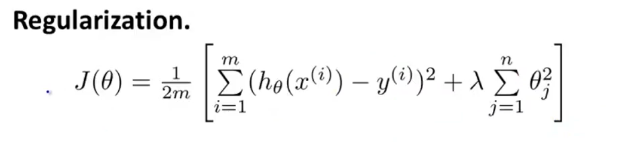
Para la predicción del precio de la vivienda podríamos tener nuestras cien variables de las que hemos hablado antes, en donde x1 es el tamaño, x2 es el número de habitaciones, x3 es el número de pisos y así sucesivamente. Podríamos tener cien variables. Y a diferencia del ejemplo del polinomio, no sabemos que «theta»3, «theta»4, son términos polinomiales de alto grado. Así que, si tenemos sólo un conjunto de cien variables, es difícil elegir de antemano cuales son las que tienen menos probabilidad de ser relevantes. Así es que tenemos cien o ciento un parámetros. Y no sabemos que parámetros elegir, para tratar de reducirlos. Así que, en la regularización, lo que vamos a hacer, es tomar nuestra función de costes, y después, modificar esta función de costes para reducir todos mis parámetros, porque no sé cuál o cuáles tengo que reducir. Así que voy a modificar mi función de costos para agregar un término al final:



Este término tiende a reducir todos mis parámetros «theta»1, «theta»2, «theta»3, hasta «theta»100.

Por cierto, por convención el sumatorio comienza desde uno, así es que no voy a penalizar «theta»0 por ser grande. Pero en la práctica, hay muy poca diferencia en los resultados incluyas «theta»0 o no. Pero por la convención, por lo general, regularizamos únicamente de «theta»1 a «theta»100.

Si escribimos nuestro objetivo de optimización regularizado, nuestra función de costes quedaría así:



Y el problema de minimización vendría como:

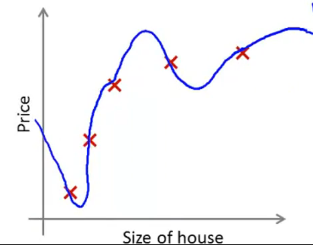


Como hemos dicho antes el nuevo término de la derecha es un término de regularización y «lambda» aquí es llamada parámetro de regularización y lo que «lambda» hace es controlar el ***“tradeoff”*** entre dos diferentes metas.

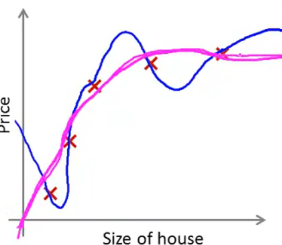
1. La primera meta, es que nos gustaría que los datos del entrenamiento se ajusten bien. Nos gustaría ajustar bien el conjunto de entrenamiento a los datos reales.
2. Y la segunda meta es, que queremos mantener los parámetros pequeños, y eso está capturado por el segundo término, al que vamos a denominar termino de regularización.

Y «lambda», es el parámetro de regularización, que lo que hace es controlar el intercambio entre estas dos metas, entre la meta de ajustar bien el conjunto de entrenamiento y la meta de mantener el parámetro pequeño y por lo tanto mantener la hipótesis relativamente simple para evitar el sobreajuste.

Para nuestro ejemplo de predicción de precios de vivienda, mientras que, anteriormente habíamos ajustado un polinomio de alto grado, podríamos haber terminado con una muy, una especie de función curva muy ondulada similar a esta:

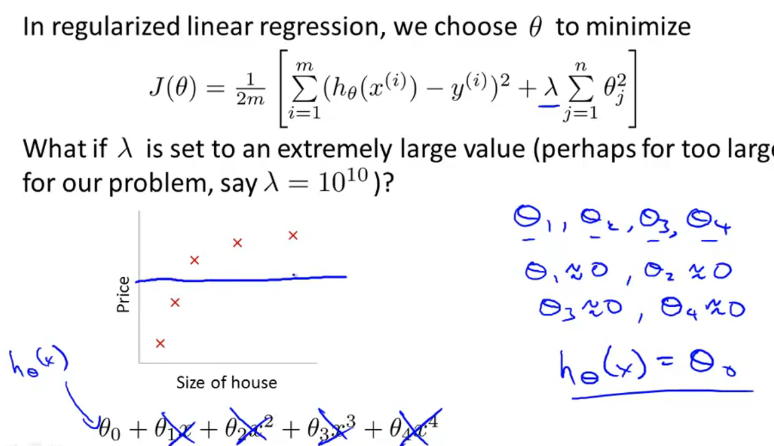


Pero en su lugar, si te aseguras de usar un objetivo de regularización, entonces lo que obtienes es, de hecho, una curva que no es una función cuadrática, pero es mucho más suave y mucho más simple y tal vez una curva como la línea magenta, como sabes, te da una hipótesis mucho mejor para estos datos:



Pero ojo, si establecemos un lambda demasiado grande también puede darse el caso de que todos los parámetros sean excesivamente pequeños y se dé el caso de subajuste.

Y si hacemos eso, es como si nos deshiciéramos de estos términos en la hipótesis así que nos quedamos solo con la hipótesis que dice que los precios de vivienda son iguales a «theta»0, y eso es similar a ajustar una línea recta horizontal plana a los datos.

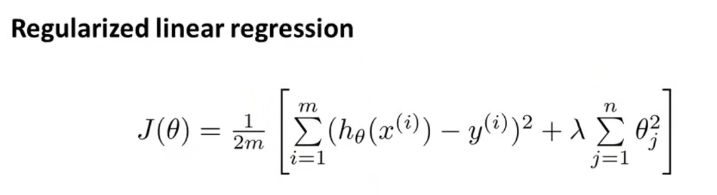


Y este es un ejemplo de subajuste, y en particular esta hipótesis, esta línea recta no puede ajustarse al conjunto de entrenamiento correctamente. Es sólo una línea recta que no se ajusta a los datos de entrenamiento. Otra manera de decir esto es que esta hipótesis tiene una preconcepción demasiado fuerte u oscilaciones demasiado altos (“high bias”) de que los precios de la vivienda son iguales a «theta»0, y a pesar de que claramente los datos dicen lo contrario. Así que para que la regularización funcione bien, debes tener cierto cuidado, para elegir también una buena opción para el parámetro de regularización «lambda». Y cuando hablemos de selección múltiple más adelante en este curso, vamos a hablar acerca de una manera, una variedad de maneras para también elegir automáticamente el parámetro de regularización «lambda». Así que esa es la idea de la regularización alta y de la función de costos que utilizaremos para poder aplicar la regularización en los dos videos siguientes, vamos a tomar estas ideas y aplicarlas a la regresión lineal y a la regresión logística, entonces podremos sacarlas para evitar problemas de sobreajuste.

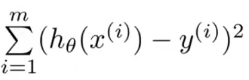
## Regularización para la regresión lineal

Para la regresión lineal, hemos desarrollado anteriormente dos algoritmos de aprendizaje, uno basado en el gradiente de descenso y el otro basado en la ecuación normal.

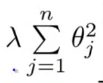
En esta sección tomaremos esos dos algoritmos y los generalizaremos para el caso de la regresión lineal regularizada. Este es el objetivo de optimización, que desarrollamos la última vez para la regresión lineal regularizada:



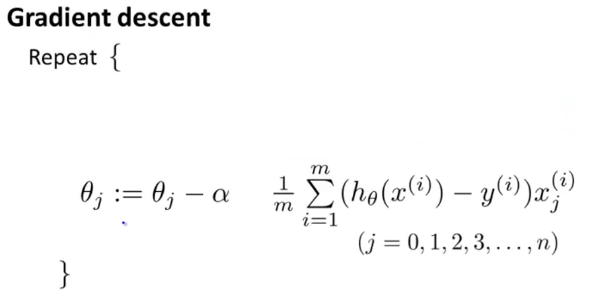
Esta primera parte es nuestro habitual objetivo para la regresión lineal:



Y ahora tenemos este término adicional de regulación:

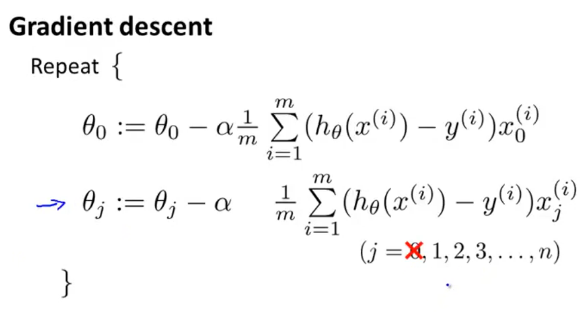


Donde «lambda» es nuestro parámetro de regularización, y queremos encontrar los parámetros «theta», que minimicen la función de costos J. Anteriormente, estábamos usando el gradiente de descenso para la original función de costos, sin el término regularización, y tuvimos el siguiente algoritmo para la regresión lineal normal, sin regularización.



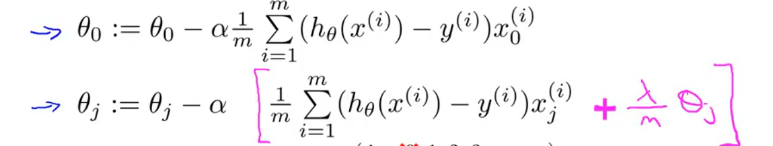
En varias ocasiones actualizaremos los parámetros para .

Antes de comenzar vamos a hacer un pequeño ajuxste y vamos a tratar por separado. Así es que voy a escribir la actualización para por separado:



De momento, no he cambiado nada todavía. Esto es solo la escritura de la actualización para «theta»0 por separado de las actualizaciones para «theta»1, «theta»2, «theta»3, hasta «theta» "n". Y la razón por la que quiero hacer esto es que quizá recuerdes que para nuestra regresión lineal regularizada, penalizamos los parámetros «theta»1, theta2 y así sucesivamente hasta «theta» "n", pero no penalizamos «theta»0. Así que cuando modificamos este algoritmo para la regresión lineal regularizada, vamos a terminar tratando a «theta»0 ligeramente diferente.

Específicamente, si queremos tomar este algoritmo y modificarlo para usar el objetivo regularizado, todo lo que necesitamos hacer es tomar el término de la parte inferior y modificarlo de la siguiente manera:

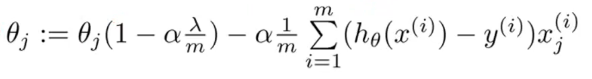


No voy a hacer el cálculo para probarlo, pero específicamente si observas el término, que está escrito entre corchetes que hemos actualizado y si sabes cálculo, es posible probar que ese término es una derivada parcial, con respecto de J(θ), usando la nueva definición de J(θ) con el término de regularización:



Y en la parte superior, aún es la derivada parcial con respecto a de J(θ).

Si miras la actualización para , es posible que muestre algo muy interesante, Si agrupas todos los términos que dependen de , podremos demostrar que esta actualización puede escribirse equivalente como sigue a continuación:

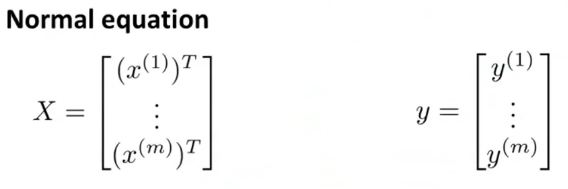


El término, va a ser un número generalmente menor ya que «alfa» multiplicada por «lambda» sobre m va a ser positivo y generalmente, si tu índice de aprendizaje es pequeño y m es grande, será bastante pequeño. Así es que este término de aquí, va a ser un número, generalmente, como sabes, un poco menor que uno. Piensa en este número como 0.99, digamos, y así, el efecto de nuestras actualizaciones de ; se remplaza por multiplicada por 0.99. Bien, entonces multiplicada por 0.99 tiene el efecto de reducir un poco hacia 0. Así que esto hace un poco más pequeña a . Más formalmente, esto como sabes, es la norma cuadrada de un poco más pequeña. El segundo término es en realidad exactamente lo mismo que la actualización original del gradiente de descenso que teníamos antes de agregar todas estas cosas de la regularización.

Así lo que estamos haciendo en cada regularización es multiplicar los datos de J por un número un poco menor a uno, para reducir el parámetro un poco.

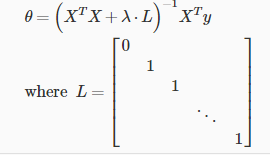
Por supuesto que es sólo la intuición detrás de lo que esta actualización en particular está haciendo. Matemáticamente, lo que está haciendo es exactamente el gradiente de descenso en la función de costes J(θ) y que utiliza un término regularización.

El gradiente de descenso fue simplemente uno de nuestros dos algoritmos para ajustar el modelo de regresión lineal. El segundo algoritmo fue el que se basa en la ecuación normal, en donde, lo que hicimos fue crear la matriz de diseño "X" donde cada fila corresponde para separar el ejemplo de entrenamiento. Y creamos un vector "y" que es un vector de "m" dimensiones y contiene el valor real asignado para los datos de nuestro conjunto de entrenamiento:



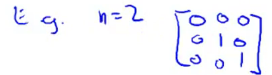
Y por último derivando la función de costes respecto de θ e igualando a 0 obteníamos la siguiente expresión que minimiza la función de costes:

Si añadimos la regularización entonces esta fórmula cambia de la siguiente manera:



Hay unos en los diagonales y ceros en cualquier otra parte de esta matriz.

Como ejemplo específico si "n" es igual a 2, entonces esta matriz va a ser una matriz de 3x3:



De forma más general, esta matriz es una matriz de dimensiones n+1 multiplicado por n+1.

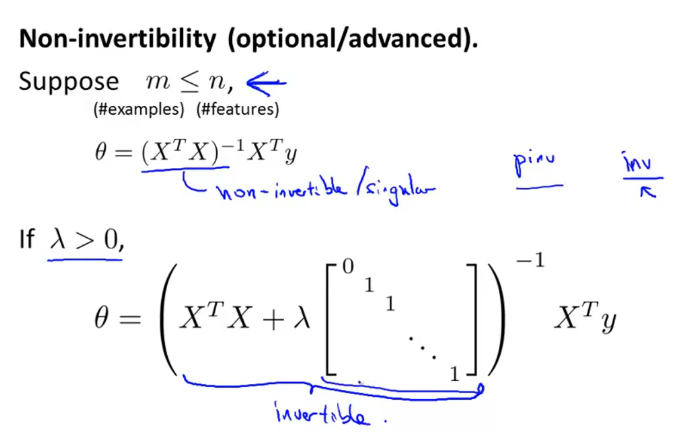
Y una vez más no voy a demostrar esta derivación, que es francamente un poco largo e involucrado. Pero es posible demostrar que si estás usando la nueva definición de J de «theta», con el objetivo de regularización es entonces esta nueva fórmula para «theta» que dará el mínimo global de J de «theta».

Finalmente, quiero describir rápidamente el problema de la no invertibilidad.

Ahora consideramos establecer dónde "m", el número de ejemplos es menor o igual que "n", el número de variables.

Si tienes menos ejemplos que variables en esta matriz será no invertible o singular, o el otro término para esto es que la matriz será degenerada y si implementas esto en Octave, de todos modos, y utilizas la función pinv para tomar la seudoinversa, hará más o menos lo correcto pero no está claro si te dará una muy buena hipótesis aunque numéricamente la función pinv en Octave te dará un resultado que más o menos tendrá sentido. Pero, si estás haciendo esto en un lenguaje diferente, y si estás tomando solo la inversa regular que en Octave se denota con la función Inv, estamos tratando de tomar la inversa regular de , luego en esta configuración encontrarás que es singular, y no invertible y si estás haciendo esto en un lenguaje de programación diferente y usando alguna biblioteca de álgebra lineal trata de tomar la inversa de esta matriz. Tal vez no funcione porque esa matriz es no invertible o singular.

Afortunadamente, la regularización también se encarga de esto para nosotros, y en específico, mientras el parámetro de regularización «lambda» es estrictamente mayor que cero, en realidad es posible demostrar que esta matriz + «lambda» multiplicado por la matriz L, no será singular y es invertible.



Así que usar la regularización también toma en cuenta cualquier problema de no invertibilidad de la matriz .

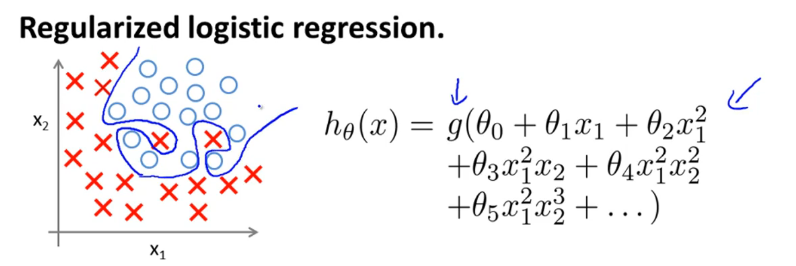
Así que, ya sabes cómo implementar la regularización de la regresión lineal. Con esto, podrás evitar el sobreajuste, incluso si tienes muchas variables en un conjunto de entrenamiento relativamente pequeño. Y esto debería permitirte lograr que la regresión lineal funcione mucho mejor para una variedad de problemas.

## Regularización para la regresión logística

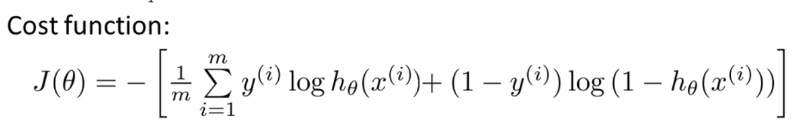
Para la regresión logística, hablamos anteriormente de dos tipos de algoritmos de optimización. Hablamos sobre cómo usar el gradiente de descenso para optimizar la función de costes J(θ). Y también hablamos sobre métodos avanzados de optimización que requieren que proporciones tu función de costes J(θ) y sus respectivas derivadas.

En esta sección, te mostraré cómo puedes adaptar ambas técnicas tanto el gradiente de descenso y las técnicas de optimización más avanzadas para ponerlas a trabajar para la regresión logística regularizada.

Vimos anteriormente que la regresión logística también es propensa a ser sobreajustada, si ajustas con variables polinomiales de alto orden. En donde G es la función sigmoidea y, particularmente, terminas con una hipótesis cuyo límite de decisión es una especie de función demasiado compleja y extremadamente enrevesada donde realmente no es una hipótesis tan buena para tu conjunto de entrenamiento y, más generalmente, si tienes una regresión logística con muchas variables. No necesariamente polinomiales, pero con una gran cantidad de variables, puedes terminar con un sobreajuste también:



Esta fue nuestra función de costo para la regresión logística:



Y si queremos modificarla para usar la regularización, todo lo que tenemos que hacer es sumarle el siguiente término:

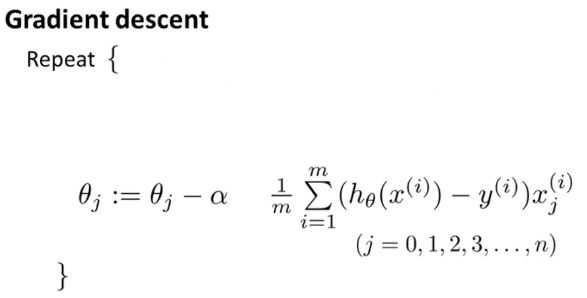


Y esto tiene el efecto de penalizar los parámetros «theta» 1 «theta» 2 y así sucesivamente hasta «theta» N para evitar que sea demasiado grande.

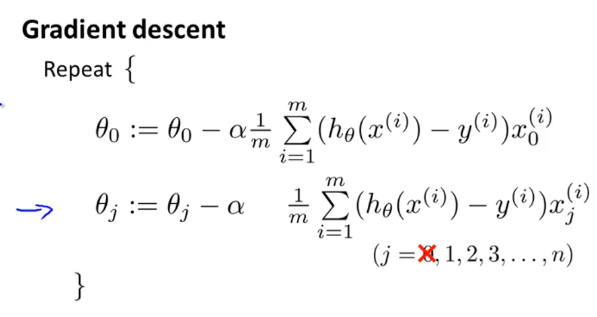
Y si haces esto, entonces tendrás el efecto de que, a pesar de que estás ajustando un polinomio de orden muy elevado con muchos parámetros, siempre que apliques regularización y mantengas los parámetros pequeños, es más probable que obtengas un límite de decisión que tal vez separe mejor los ejemplos positivos y negativos.

Así que, al usar regularización incluso cuando tienes muchas variables, la regularización puede ayudar a resolver el problema del sobreajuste.

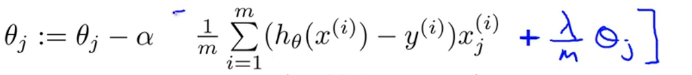
¿Cómo implementamos esto? Bien, para el algoritmo original del gradiente de descenso, esta era la actualización que teníamos:



Haremos repetidamente la siguiente actualización a «theta» J. Este lado se ve muy parecido al anterior para la regresión lineal. Y otra vez lo que voy a hacer es escribir la actualización para «theta» 0 por separado. Entonces, la primera línea es la actualización para «theta» 0 y una segunda línea es ahora mi actualización para «theta» 1 hasta «theta» N, porque yo voy a tratar «theta» 0 por separado.



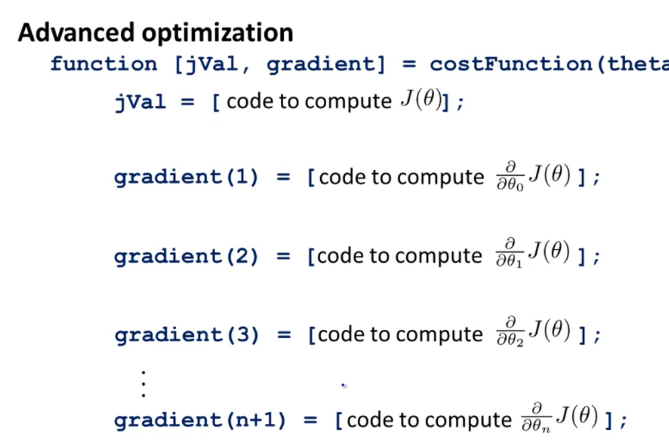
Y con el fin de modificar este algoritmo para usar una función de costo regularizada, todo lo que necesito hacer es muy parecido a lo que hicimos para la regresión lineal que, de hecho, sólo es modificar esta segunda regla de actualización de la siguiente manera:



Y, nuevamente, esto, se ve cosméticamente igual a lo que teníamos para la regresión lineal. Pero, desde luego, no es el mismo algoritmo que teníamos, porque ahora la hipótesis se define usando la función sigmoidea. Entonces, no es el mismo algoritmo que el de la regresión lineal regularizada. Debido a que la hipótesis es diferente, incluso a pesar de que la actualización que escribí, de hecho, se ve cosméticamente igual a la que teníamos anteriormente pero estamos trabajando la gradiente de descenso para la regresión logística regularizada.

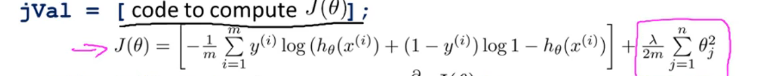
Y, desde luego, sólo para resumir la discusión, este término aquí entre corchetes, este término es, desde luego, la nueva derivada parcial con respecto a «theta» J de la nueva función de costo J de «theta». Donde J de «theta» aquí es la función de costes que definimos anteriormente y que usa la regularización.

Hablemos sobre cómo hacer que trabaje la regresión lineal usando los más avanzados métodos avanzados de optimización. Y sólo para recordarte, lo que necesitábamos hacer para estos métodos era definir la función denominada función de costes, lo que requiere que introduzcamos el vector del parámetro «theta» y, nuevamente, en las ecuaciones que hemos estado escribiendo aquí hemos usado vectores con índice 0. Así que teníamos «theta» 0 hasta «theta» N. Pero, porque Octave indexa los vectores a partir de 1 y «theta» 0 se escribe como «theta» 1 en Octave, «theta» 1 se escribe en Octave como «theta» 2, y así hasta «theta»n +1. Y lo que necesitábamos hacer era proporcionar una función.

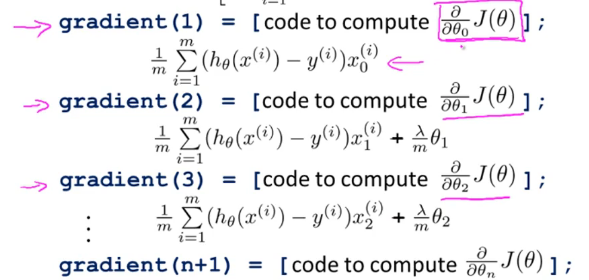


Vamos a proporcionar una función llamada función de costes donde le vamos a pasar los mismos argumentos que ya vimos. Vamos a utilizar que tomará la función de costo y la minimizará para nosotros.

Así que, las dos cosas principales que la función de costes necesitaba para regresar el valor correcto eran primero, J-val. Y, para esto, tenemos que escribir el código para calcular la función de costes J de «theta» dependiendo si usamos la regresión logística o la regresión lineal y si es regularizada o no (Con lo que actualmente con la teoría aprendida conocemos las 4 funciones que deberían ir ahi) y para el caso concreto de la función de costes para la regresión lineal regularizada jVal será:



Y después, la otra cosa que esta función de costes necesita son los respectivos gradientes. Entonces, la gradiente uno debe establecerse como la derivada parcial de J de «theta» con respecto a «theta» cero; la gradiente dos debe establecerse como como la derivada parcial de J de «theta» con respecto a «theta» uno, y así sucesivamente. Una vez más, el índice está desplazado por uno. Así es, debido a la indexación desde uno que utiliza Octave.



Vemos que la derivada para «theta» cero no cambia. En comparación con la versión sin regularización.

Los otros términos sí cambian. Sólo para asegurarnos de que hemos pasado esto correctamente. Y podemos añadir paréntesis aquí. Bien, así la sumatoria no se extiende.



Entonces, si implementas esta función de costes y pasas esto a o a una de las técnicas de optimización avanzadas, se minimizará la nueva función de costo regularizada J de «theta». Y los parámetros que obtendrás serán los que correspondan a la regresión logística con regularización.