# Regresión logística

Contenido

[Regresión logística 1](#_Toc505422917)

[1.1 Clasificación 1](#_Toc505422918)

[1.2 Representación de hipótesis. 4](#_Toc505422919)

[1.2.1 Función sigmoidea 5](#_Toc505422920)

[1.2.2 Interpretación del modelo 6](#_Toc505422921)

[1.3 Límite de decisión. 8](#_Toc505422922)

[1.3.1 Límites de decisión no lineales 11](#_Toc505422923)

[1.4 Función de costes 14](#_Toc505422924)

[1.4.1 Función de costes simplificada y descenso del gradiente 19](#_Toc505422925)

[1.5 Optimización avanzada 23](#_Toc505422926)

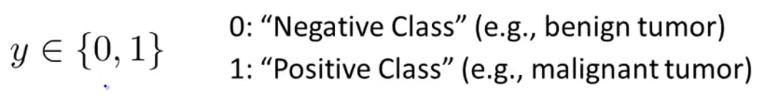
[1.6 Regresión logística Multi – Clase 29](#_Toc505422927)

## Clasificación

En esta sección y en la siguiente vamos a hablar de los problemas de clasificación, donde la variable y lo que quieres predecir tienen valores discretos. Hemos desarrollado un algoritmo llamado regresión logística, que es uno de los algoritmos de aprendizaje más populares y más ampliamente utilizados actualmente. Estos son algunos ejemplos:

* Email: clasificar entre Spam y no Spam
* Fraude: Transacción fraudulenta (si/no)
* Tumor: Maligno/Benigno

En todos estos problemas la variable que intentamos predecir (y) toma dos valores, 0 o 1:

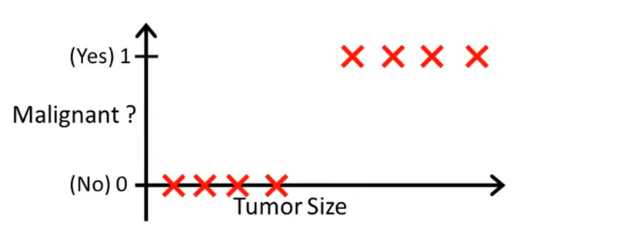


Otro nombre para la clase que denotamos con 0 es la clase negativa, y otro nombre para la clase que denotamos con 1 es la clase positiva. Así que 0 puede indicar un tumor benigno y 1, la clase positiva, puede indicar un tumor maligno. La asignación de las 2 clases, la asignación de las 2 clases a positivo y negativo, a 0 y 1 es un tanto arbitraria y en realidad no importa. Pero a menudo existe esta intuición de que la clase negativa está transmitiendo la ausencia de algo, como la ausencia de un tumor maligno, mientras que uno, la clase positiva, está transmitiendo la presencia de algo que podríamos estar buscando. Pero la definición de lo que es negativo y de lo que es positivo es algo arbitrario y que no importa mucho. Por ahora, vamos a empezar con problemas de clasificación con sólo dos clases; cero y uno. Más adelante, hablaremos de problemas multiclase, en donde la variable "y" puede tomar por ejemplo, el valor de:

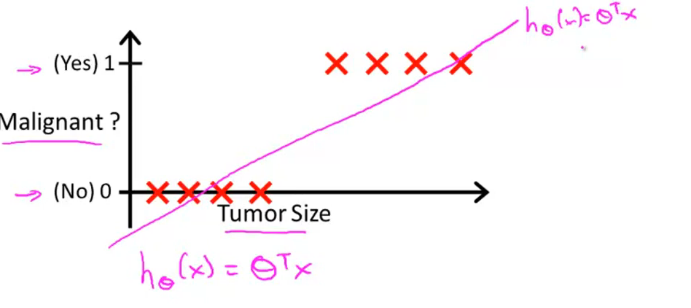
.

Esto se llama problema de clasificación multiclase, pero para las siguientes secciones, vamos a empezar con el problema de dos clases o de clasificación binaria, y nos ocuparemos de la configuración multiclase después.

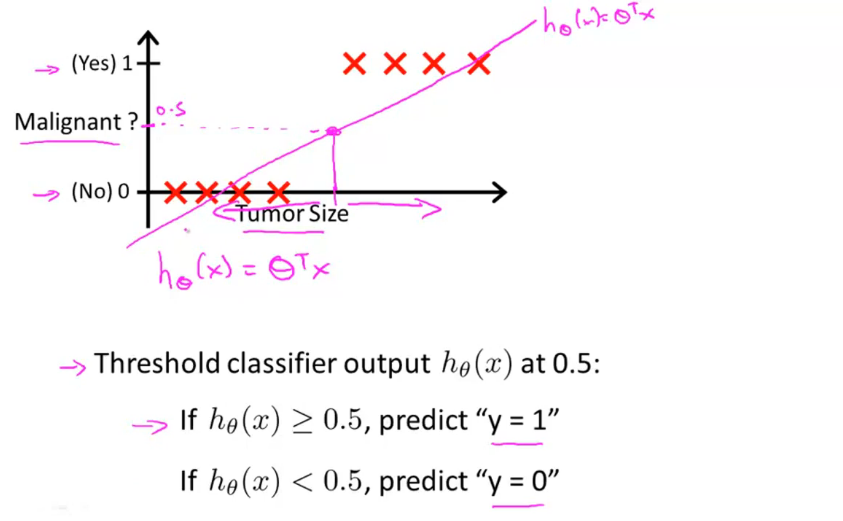
Entonces, ¿cómo se desarrolla un algoritmo de clasificación? Aquí hay un ejemplo de un conjunto de entrenamiento para una tarea de clasificación para clasificar un tumor como maligno o benigno y observa que la “malignidad” adquiere únicamente dos valores, de 0 o No o de 1 o Sí.



Así, una cosa que podríamos hacer dado este conjunto de entrenamiento es aplicar el algoritmo que ya conocemos, la regresión lineal para este conjunto de datos y tratar de ajustar la línea recta a los datos. Así que, si tomas este conjunto de entrenamiento y si ajustas una línea recta al mismo, tal vez obtienes una hipótesis que se parece a esto:

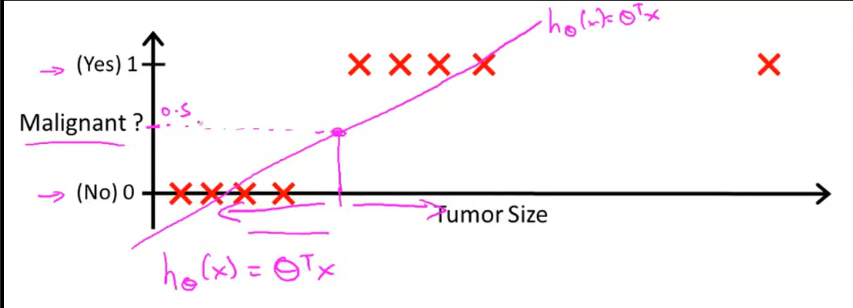


Si quieres hacer predicciones, una cosa que puedes tratar de hacer es el umbral de los resultados del clasificador en 0.5. Que está en el valor de acceso vertical 0.5. Y si la hipótesis da como resultado un valor que es mayor que o igual a 0.5 tu predicción de "y=1". Si es menor que 0.5, tu predicción de "y=0".

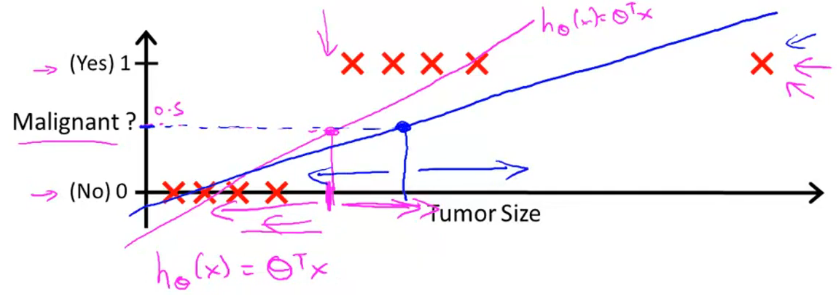


Si tomamos 0.5 que es donde hemos fijado el umbral. Y por lo tanto, al usar la regresión lineal de esta manera, todo lo que está a la derecha de este punto, terminará prediciendo la clase positiva debido a que los valores resultados son mayores que 0.5 en el eje vertical y todo lo que está a la izquierda de este punto, terminará prediciendo un valor negativo. En este ejemplo particular, parece que la regresión lineal está en realidad realizando algo razonable a pesar de que esta es una tarea de clasificación en la cual estamos interesados. Pero ahora vamos a tratar de cambiar un poco el problema.

Digamos que tenemos otro ejemplo de entrenamiento más allá a la derecha. Observa que el ejemplo de entrenamiento adicional.



Una vez que recalculamos nuestra línea quedaría así:



Lo cual no parece bastante bien. Nosotros podemos observar visualmente donde está el punto de corte que diferencia a una clase de otra pero al agregar el nuevo punto la forma en el que se calcula la línea en la regresión lineal afecta totalmente ya que la línea se ajusta a los datos y causa que nuestra hipótesis sea peor. Por lo que aplicar una regresión lineal a un problema de clasificación a menudo no es una buena idea.

Aquí hay otra cosa curiosa sobre lo que sucedería si tuviéramos que usar Regresión lineal para un problema de clasificación. Para la clasificación, sabemos que y es cero o uno. Pero si estás usando regresión lineal donde la hipótesis puede dar salida a valores que son mucho más grandes que uno o menos que cero, incluso si todos sus ejemplos de entrenamiento tienen etiquetas y es igual a cero o uno. Y parece un poco extraño que a pesar de que saber que las etiquetas deben ser cero, el algoritmo puede generar valores mucho más grandes que uno o mucho más pequeño que cero.

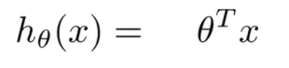
Entonces, lo que haremos en las próximas secciones es desarrollar un algoritmo llamado regresión logística, que tiene la propiedad de que el resultado (las predicciones) son siempre entre cero y uno, y nunca supera este rango de valores.

Y por cierto, la regresión logística es, y lo usaremos como un algoritmo de clasificación, es algo, a veces puede ser confuso que el término regresión aparece en este nombre a pesar de que la regresión logística es en realidad un algoritmo de clasificación. Pero ese es solo un nombre que se le dio por razones históricas. Así que no te confundas con esa regresión logística que en realidad es un algoritmo de clasificación que aplicamos a las configuraciones donde la etiqueta y es un valor discreto, cuando es cero o uno. Así que con suerte ahora sabes por qué, si tienes un problema de clasificación, usar la regresión lineal no es una buena idea.

## Representación de hipótesis.

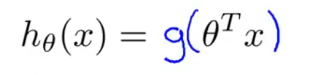
La representación de la hipótesis, que es, la función que vamos a utilizar para representar nuestra hipótesis en la que tenemos un problema de clasificación.

Anteriormente, dijimos que nos gustaría que nuestro clasificador genere valores que se encuentren entre cero y uno. Por lo tanto, nos gustaría proponer una hipótesis que satisface esta propiedad, que estas predicciones tal vez están entre cero y uno. Cuando estábamos usando regresión lineal, esta era la forma de una hipótesis:

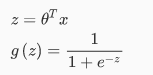


Donde "h" de "x" es «theta» transpuesta multiplicada por x.

Para la regresión logística, vamos a modificar esto un poco y hacer la hipótesis "g" de «theta» transpuesta x:

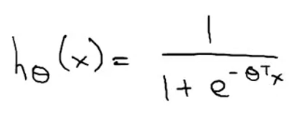


Donde voy a definir la función "g" de la siguiente manera: "g" de "z" si "z" es un número real que es igual a uno sobre uno más "e" a la "z" negativa:



Esta se llama función sigmoidea o función logística. Y el término función logística, es lo que da lugar al nombre de regresión logística. Los términos función sigmoidea y función logística son básicamente sinónimos y significan lo mismo. Así que los dos términos son básicamente intercambiables y cualquier término puede ser utilizado para referirse a esta función "g".

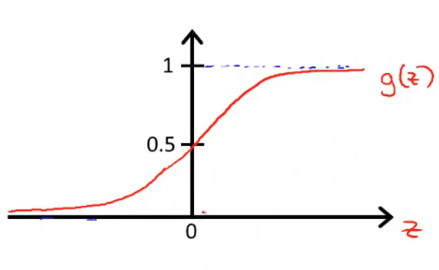
Si tomamos estas dos ecuaciones y, las ponemos juntas, entonces aquí tenemos simplemente una manera alternativa de escribir la formulación de mi hipótesis:

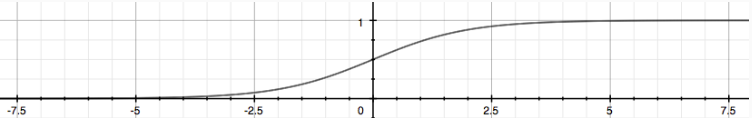


Estoy diciendo que "h" de "x" es uno sobre uno más "e" elevado a «theta» transpuesta x negativa.

## Función sigmoidea

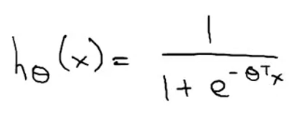
La función sigmoidea, "g" de "z", también llamada la función logística, se ve así:





Comienza cerca de cero y luego se eleva hasta que se procesa 0.5 en el origen aplanandose de nuevo cuando llega a uno. La función sigmoidea, tiene una asíntota en uno, y otra en cero mientras que "z" (eje horizontal) va hacia menos infinito, "g" de "z" se aproxima a cero y tiende a infinito si "g" de "z" se aproxima a 1, y así porque "g" de "z" ofrece valores que están entre 0 y 1. También tenemos que "h" de "x" debe estar entre 0 y 1.

Por último, dada esta representación de la hipótesis:

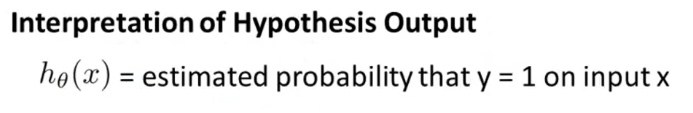


Lo que necesitamos hacer, como antes, es ajustar los parámetros «theta» a nuestros datos. Así que dado un conjunto de entrenamiento, necesitamos elegir un valor para los parámetros «theta» y esta hipótesis nos permitirá entonces hacer predicciones.

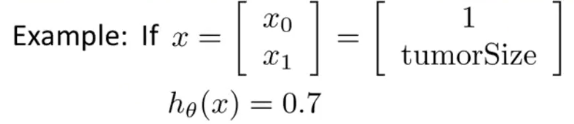
## Interpretación del modelo

Vamos a hablar de un algoritmo de aprendizaje más tarde para el ajuste de los parámetros «theta». Pero primero vamos a hablar un poco acerca de la interpretación de este modelo.

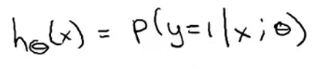
Cuando mi hipótesis dé como resultado algún número, voy a tratar a ese número como la probabilidad estimada de que "y" es igual a 1.



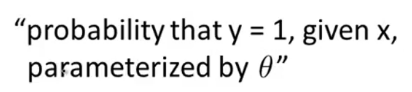
Veamos un nuevo ejemplo:



Digamos que estamos utilizando como ejemplo la clasificación de tumores. Así es que podemos tener un vector de variables "x", que es este x0=1 (como siempre) y nuestra única variable es el tamaño del tumor. Supongamos que tengo un paciente que viene y, con una radiografía de un tumor y lo mido. Entonces una vez medido introduzco su vector de variables "x" en mi hipótesis y supongamos que mi hipótesis da como resultado el número 0.7. Esta hipótesis me está diciendo que para un paciente con "x" variables, la probabilidad de que "y" sea igual a 1 es de 0.7. En otras palabras, le voy a decir a mi paciente que el tumor, por desgracia, tiene un 70% de probabilidad o una probabilidad de 0.7 de ser maligno. Para escribir este resultado un poco más formal o para escribir este resultado matemáticamente, voy a interpretar el resultado de mi hipótesis como "p" de "y" que es igual a 1, dado que "x" está parametrizada por «theta»:

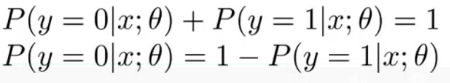


Por lo tanto, para aquellos que estén familiarizados con la probabilidad, esta ecuación podría tener sentido, Esta es la probabilidad de que "y" es igual a uno, dado por mis variables "x", y esta probabilidad está parametrizada por «theta».



Así que básicamente voy a contar con que mi hipótesis me dará las estimaciones de la probabilidad de que "y" es igual a 1. Ahora ya que esta es una tarea de clasificación, sabemos que "y" debe ser cero o uno, ¿verdad? Esos son los dos únicos valores que "y" podría asumir, tanto en el conjunto de entrenamiento o para los nuevos pacientes que puedan entrar a mi oficina o en el consultorio médico en el futuro.

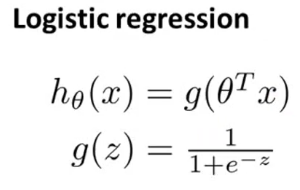
Entonces, dado "h" de "x", podemos calcular la probabilidad de que "y" es igual a cero también. Específicamente, porque "y" debe ser cero o uno, sabemos que la probabilidad de que "y" es igual a cero, más la probabilidad de que "y" es igual a uno, deben sumar uno.



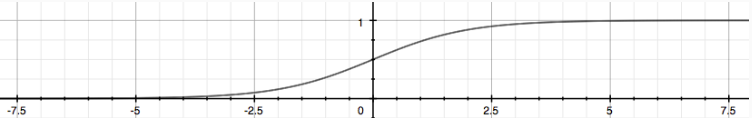
Está ecuación nos está diciendo básicamente que la probabilidad de que "y" sea igual a cero para un paciente en particular dado las variables "x", y el vector de parámetros θ más la probabilidad de que "y" sea igual a uno para el mismo paciente con variables "x" y «theta» ambas, deben sumar uno. Y esto es sólo decir que la probabilidad de que "y" es igual a 0, más la probabilidad de que "y" es igual a 1 debe ser igual a uno. Y sabemos que esto es verdad porque "y" tiene que ser cero o uno. Y si solamente tomas este término y lo mueves al lado derecho, entonces terminarás viendo que la probabilidad de que "y" es igual a cero es igual a 1 menos la probabilidad de que "y" es igual a 1.

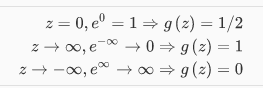
## Límite de decisión.

Dijimos que la hipótesis se representa como:



Donde g es la la función sigmoidea, que se ve así:





Lo que quiero ahora es tratar de comprender mejor cuando esta hipótesis hará predicciones donde es igual a 1 versus cuando podría hacer predicciones de que es igual a 0. Y comprender mejor cómo se ve la función de hipótesis particularmente cuando tenemos más de una característica.

Concretamente, esta hipótesis está generando estimaciones de la probabilidad de que sea igual a 1, dado y parametrizado por ().

Entonces, si quisiéramos predecir si es igual a uno o es igual a cero, aquí hay algo que podríamos hacer.

Siempre que la hipótesis genere una probabilidad de que sea igual a 1 mayor o igual a 0.5, significa que hay más probabilidad de que sea igual a 1 que sea igual a 0, entonces vamos a predecir qué y es igual a 1. Y de lo contrario, si la probabilidad estimada de ser e igual a 1 es menor que 0.5, entonces pronosticaremos que y es igual a 0.

¿Por qué se elige igualdad en uno y menor o estricto en otro?

Si miramos el diagrama de la función sigmoidea arriba, notaremos que la función sigmoidea, es mayor o igual a 0.5 siempre que z sea mayor o igual a cero. Entonces cuando z es positivo, , la función sigmoidea es mayor o igual a 0.5.

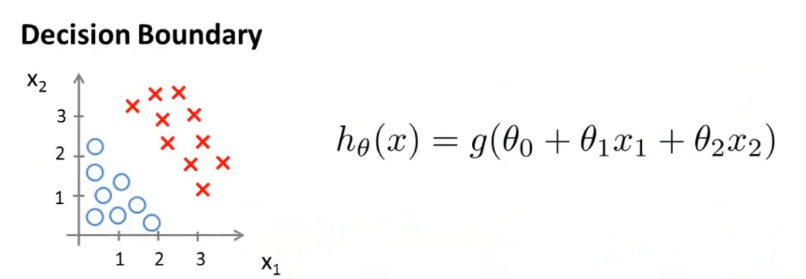
Dado que la hipótesis para la regresión logística es que es igual a esto va a ser mayor o igual a 0.5, siempre que es mayor o igual que 0. Con lo que podemos deducir correcto, porque aquí es z.

Entonces, lo que se muestra es que una hipótesis va a predecir qué es igual a 1 siempre que .

Consideremos ahora el otro caso de cuando tenemos una hipótesis y vamos a predecir . Bueno, por un argumento similar, va a ser menor que 0.5 siempre que sea menor que 0.5 porque el rango de valores de z que causa que tome valores menores a 0.5 es cuando z es negativo. Entonces cuando g (z) es menor que 0.5, una hipótesis predecirá que es igual a 0. Y, por un argumento similar a lo que teníamos antes, y entonces predeciremos y es igual a 0 siempre que sea menor que 0.

Resumiendo, vimos que si decidimos predecir o dependiendo de si la probabilidad estimada es mayor o igual a 0.5, o si es menor que 0.5, entonces eso es lo mismo que decir que cuando predecimos y = 1 siempre que es mayor o igual que 0. Y predeciremos que y es igual a 0 siempre que theta sea menor que 0.

Usemos esto para entender mejor cómo la hipótesis de la regresión logística hace esas predicciones. Supongamos que tenemos un conjunto de entrenamiento donde la hipótesis es igual a la que se muestra a continuación:



Todavía no hemos hablado sobre cómo ajustar los parámetros de este modelo. Hablaremos de eso en la próxima sección. Pero supongamos que a través de un procedimiento especificado. Terminamos eligiendo los siguientes valores para los parámetros. Digamos que elegimos , , . Entonces esto significa que mi vector de parámetros va a ser:

Entonces, una vez he elegido esta opción de parámetros para mi hipótesis, vamos a ver dónde terminaría prediciendo si es igual a uno y donde terminaría prediciendo y es igual a cero.

Sabemos que y es igual a uno si:

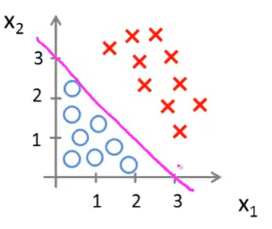


O lo que es lo mismo que que es igual a sea mayor que 0. Lo que devolverá un valor mayor o igual a 0.5 para .

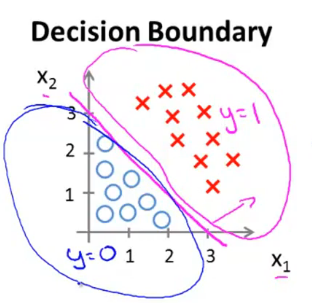
También podemos tomar -3 y llevar esto a la derecha y reescribe esto como:

Por lo que de manera equivalente, encontramos que esta hipótesis predeciría y = 1 siempre que x1 + x2 sea mayor o igual que 3.

Veamos qué significa eso en la figura, si escribo la ecuación, , esto define la ecuación de una línea recta y si dibujo cómo es esa línea recta, me da la siguiente línea que pasa por 3 y 3 en los ejes x1 y x2.



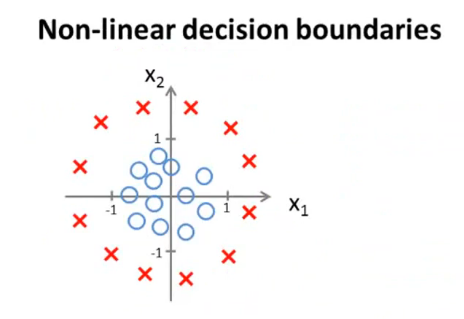
Igualando la ecuación a 3 hemos delimitado la línea separatoria para los valores donde z es igual a 0 (donde se debe de cumplir que . Si ponemos un punto por debajo de la línea magenta, el resultado de será un valor negativo y por lo tanto será < 0.5 e . Por el lado contrario, si obtenemos un valor por encima de la clase magenta, será mayor que 0, será 0.5 e y será igual a 1. Esa línea que separa las clases es mi límite de decisión o “*decisión boundary*” en inglés y se corresponde con los valores que hacen a .



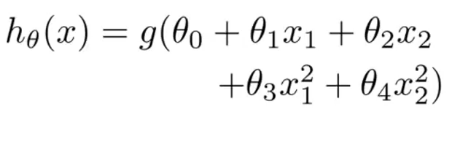
Más adelante, por supuesto, hablaremos sobre cómo ajustar los parámetros y allí terminaremos usando el conjunto de entrenamiento, usando nuestros datos. Para determinar el valor de los parámetros. Pero una vez que tenemos valores particulares para los parámetros theta0, theta1, theta2 podemos definir por completo el límite de decisión.

## Límites de decisión no lineales

Veamos ahora un ejemplo más complejo donde, como de costumbre, tengo cruces para denotar mis ejemplos positivos y circulos para denotar mis ejemplos negativos. Dado un conjunto de entrenamiento como este, ¿cómo puedo obtener la regresión logística para que se ajuste a este tipo de datos?



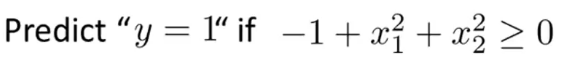
Antes, cuando hablábamos de regresión polinomial o cuando hablamos de regresión lineal, hablamos sobre cómo podríamos agregar términos polinomiales de un orden extra superior a nuestras variables. Y podemos hacer lo mismo para la regresión logística. Concretamente, digamos que mi hipótesis se parece a esto:



Donde he agregado dos características adicionales, x1 al cuadrado y x2 al cuadrado, a mis variables de partida. Entonces ahora tengo cinco parámetros.

En la siguiente sección discutiremos sobre cómo elegir automáticamente los valores para todos estos parámetros. Pero digamos que el procedimiento varia un poco y ahora:

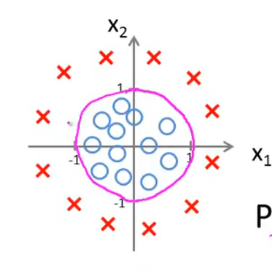
Después de nuestra discusión anterior, esto significa que mi hipótesis predecirá que y = 1 siempre que:



Esto es siempre que . Y si tomo menos 1 y solo llevo esto a la derecha:

Estoy diciendo que mi hipótesis predecirá que y es igual a 1 siempre que x1 cuadrado más x2 al cuadrado es mayor o igual a 1.

Entonces, ¿cómo se ve este límite de decisión? Bueno, si fuera a trazar la curva para x1 al cuadrado más x2 al cuadrado es igual a 1 Algunos de ustedes reconocerán eso, esa es la ecuación para círculo de radio uno, centrado alrededor del origen. Entonces ese es mi límite de decisión.

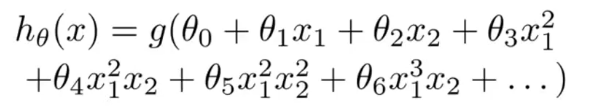


Y todo lo que esté fuera del círculo, voy a predecir cómo y = 1, y dentro del círculo es donde voy a predecir qué y es igual a 0. Entonces, agregando estos términos más complejos o polinomiales a mis características también, Puedo obtener límites de decisión más complejos.

12:44

Una vez más, el límite de decisión es una propiedad, no del conjunto de entremaiento, pero de la hipótesis bajo los parámetros. Entonces, siempre y cuando nos den mi vector de parámetro theta, que define el límite de decisión, que es el círculo. Pero el conjunto de entrenamiento no es lo que usamos para definir el límite de decisión. El conjunto de entrenamiento se puede usar para ajustar los parámetros theta. Hablaremos de cómo hacerlo más tarde. Pero, una vez que tienes los parámetros theta, eso es lo que define el límite de las decisiones.

Y, finalmente, veamos un ejemplo más complejo, ¿podemos llegar a límites de decisión aún más complejos que esto? Si tengo términos polinomiales aún más altos, entonces cosas como:

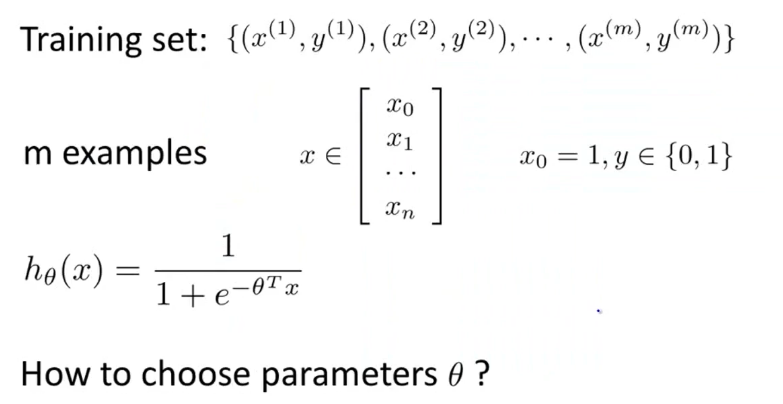


Es posible demostrar que puedes obtener límites de decisión aún más complejos y la regresión puede ser usado para encontrar límites de decisión que pueden, por ejemplo, ser una elipse o tal vez un ajuste un poco diferente de los parámetros, tal vez se puede obtener en su lugar un límite de decisión diferente. Estas características autopolinomiales más altas pueden tener límites de decisión muy complejos.

Ahora que sabemos lo que puede representar, lo que me gustaría hacer a continuación en la siguiente sección es hablar sobre cómo elegir automáticamente los parámetros Theta para que Dado un conjunto de entrenamiento, podemos ajustar automáticamente los parámetros a nuestros datos.

## Función de costes

Tenemos un conjunto de entrenamiento de m ejemplos de entrenamiento:

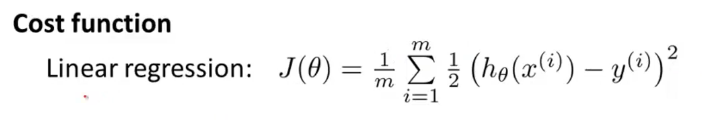


Como de costumbre, cada uno de nuestros ejemplos está representado por un vector n+1 dimensional, y como siempre tenemos x0 que es igual a 1. La primera característica o característica cero siempre es igual a uno.

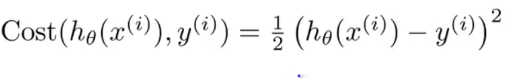
Debido a que este es un problema de clasificación, nuestro conjunto de entrenamiento tiene la propiedad de que cada etiqueta es 0 o 1.

Y finalmente nuestra hipótesis viene representada por .

Una vez tenemos el problema planteado, ¿cómo elegimos, o cómo encajamos el parámetro theta? Cuando estábamos desarrollando el modelo de regresión lineal, utilizamos la siguiente función de costo:



He escrito esto de forma ligeramente diferente, en lugar de 1 sobre 2 m, Tomé la mitad y la puse dentro de la sumatoria. Ahora quiero usar una forma alternativa de escribir esta función de costos.



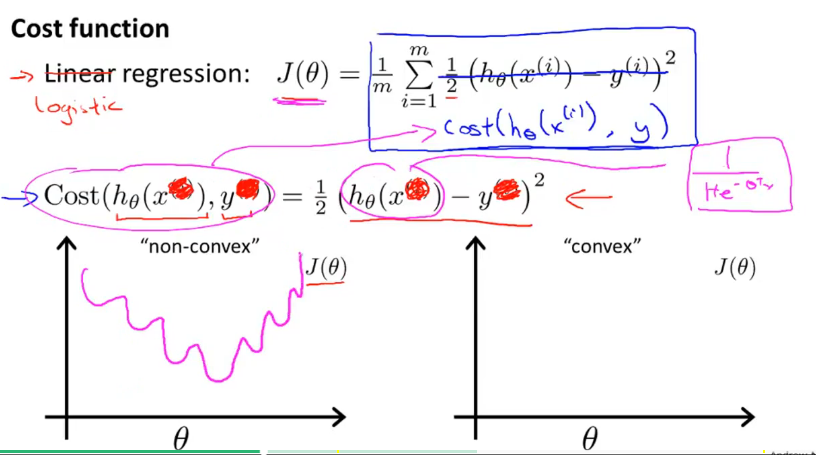
Entonces ahora podemos ver más claramente que la función de costo es una suma sobre mi conjunto de entrenamineto, que va de 1 a m del término de costes que es la diferencia entre el resultado de la hipótesis según mis variables menos la variable real todo ello al cuadrado.

Y para simplificar esta ecuación un poco más, va a ser conveniente deshacerse de esos superíndices. Así que solo defina el costo de como la mitad de este error cuadrado:

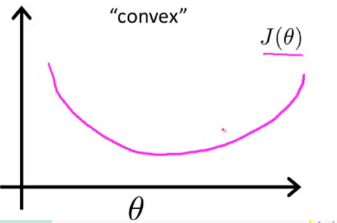
Y la interpretación es que este es el costo es la mitad de la diferencia cuadrada entre la predicción y el valor real que tenemos. Ahora esta función de costo funcionó bien para la regresión lineal. Pero aquí, estamos interesados ​​en la regresión logística. Si pudiéramos minimizar esta función de costo funcionará bien Pero resulta que si usamos esta función de costo particular, esta sería una función no convexa de los datos del parámetro. Imagina que tenemos alguna función cruzada j de theta y para la regresión logística, esta función h de arriba que como hemos dicho tiene una no linealidad que es:



Entonces esta es una función no lineal bastante complicada. Y si tomas la función, y la introduces en tu función de costes. Y luego esta función de costes la introduces en J(θ). Descubrirás que J(θ) podrá verse como una función así:



Con muchos óptimos locales Y el término formal para esto es que es una función no convexa. Y tipo de función no se garantiza que el gradiente converja al mínimo global. Mientras que en contraste, lo que nos gustaría es tener una función de costo J(θ) que sea convexa, que es una función única en forma de arco que se ves así:



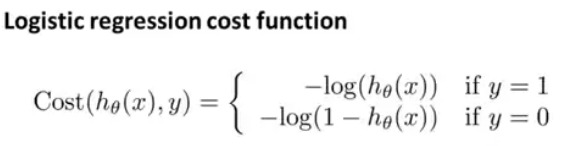
Que garantiza que converja al mínimo global. Y el problema con el uso de esta gran función de costos:



Es que debido a que es una función no lineal y por lo tanto, J(θ) termina siendo una función no convexa si tuvieras que definirla como una función de costes cuadrática.

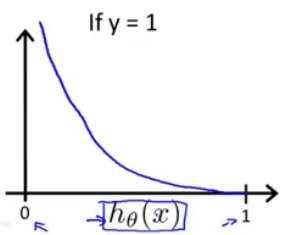
Entonces, lo que nos gustaría hacer es, encontrar una función de costes diferente que sea convexa para poder aplicar un algoritmo, como el descenso de gradiente y garantizar que encontrará el mínimo global.

Y esa función de costes que vamos a usar para la regresión logística es la siguiente:



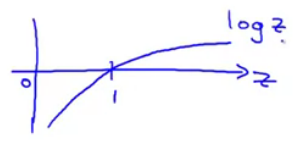
Vamos a decir que el coste, o la penalización que paga el algoritmo, si aumenta el valor de , entonces si predecimos un número como 0.7, el valor, y la etiqueta del valor real que es . El costo será -log (,) si y = 1 y -log (1- ,) si y = 0. Esto parece una función bastante complicada, pero tracemos esta función para ganar algo de intuición sobre lo que estamos haciendo.

Comencemos con el caso de y = 1. Si y = 1, entonces la función de costo es -log (). Y si trazamos eso, entonces digamos que el eje horizontal es , y sabemos que una hipótesis arrojará un valor entre 0 y 1, entonces , varía entre 0 y 1. Si dibujas esta función de costo, encuentras que se ve así:

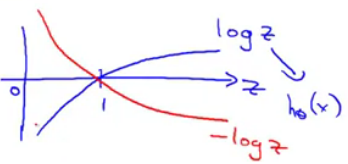


Una forma de ver por qué la trama se ve así es porque si fuera a trazar con z en el eje horizontal, entonces eso se ve así:

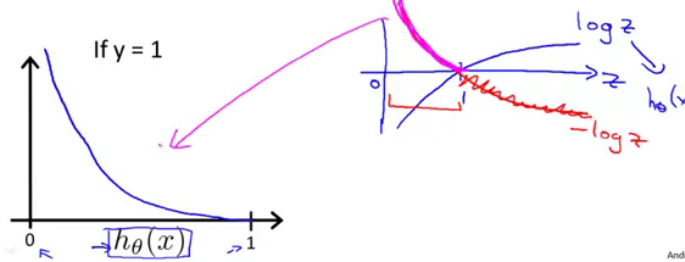
Y se acerca al infinito, ¿verdad? Entonces, así es como se ve la función de registro. Y esto es 0, esto es 1. Aquí, z por supuesto está desempeñando el papel de h de x. Y entonces -log z se verá así:



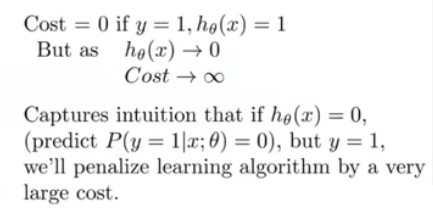
Que es la función normal del logaritmo, simplemente volteando el signo :



y solo estamos interesados ​​en el rango de cuando esta función va entre cero y uno, así que deshazte lo que haga la función del uno en adelante me da exactamente igual. Y así nos quedamos, con esta parte de la curva:



Ahora, esta función de costo tiene algunas propiedades interesantes y deseables.



Primero, nota que sí es igual a 1 y es igual a 1, en otras palabras, si la hipótesis predice exactamente que h es igual a 1 e es exactamente igual a lo que predijo, entonces el costo = 0 si solo estamos considerando el caso de y = 1.

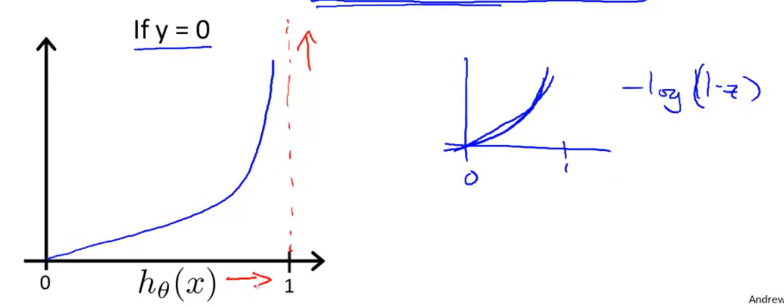
Pero ahora observe también que cuando se acerca a 0, entonces el resultado de una hipótesis se aproxima a 0, el costo aumenta y se va al infinito. Y lo que esto hace es capturar la intuición de que si una hipótesis de 0, es como decir una hipótesis donde la probabilidad de que y sea igual a 1 es igual a 0. Es un poco como ir a nuestros pacientes médicos y decir la probabilidad de que tengas un tumor maligno, la probabilidad de que y = 1, sea cero. Entonces, es absolutamente imposible que su tumor sea maligno.

Pero si resulta que el tumor del paciente, en realidad es maligno, entonces la y real es igual a uno, y resulta que estamos equivocados, entonces penalizamos el algoritmo de aprendizaje por un costo muy, muy grande. Y eso se captura al tener ese coste que va a infinito si y es igual a 1 y se aproxima a 0.

Veamos cómo se ve la función de costo para y es igual a 0:



Y si trazas la función , lo que obtienes es la función de costes que se ve así:



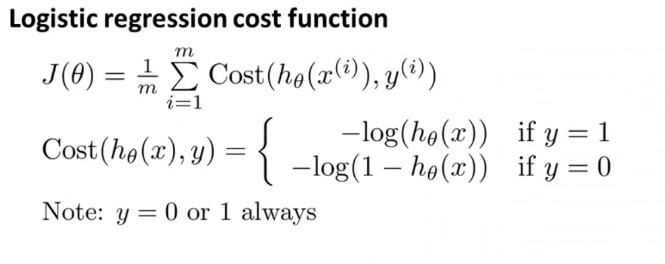
Así que va de 0 a 1, con una asíntota en el eje horizontal en 1. Y lo que hace esa asíntota es que ahora sube y se va a más infinito cuando pasa a 1.

Y, lo que esta función de costo hace es que sube o va a un infinito positivo cuando va a 1, y esto capta la intuición de que si la hipótesis predijo que es igual a 1 con certeza, y absolutamente y va es igual a 1 y resulta ser igual a 0, entonces tiene sentido que el algoritmo de aprendizaje aquí sea un costo muy grande. Y por el contrario, si es igual a 0 e es igual a 0, entonces la función de costo va a ser 0.

## Función de costes simplificada y descenso del gradiente

En esta sección descubriremos una forma un poco más simple de escribir la función de costos de lo que hemos estado usando hasta ahora. Y también descubriremos cómo aplicar el descenso del gradiente para adaptarse a los parámetros de la regresión logística.

Aquí está nuestra función de costes para la regresión logística:



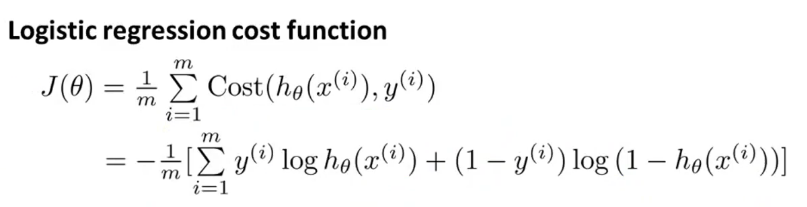
Como hemos visto, nuestra función de costes general es 1 sobre m multiplicado por el sumatorio de hacer diferentes predicciones sobre los diferentes datos. Debajo tenemos el coste una sola predicción. Y a modo de recordatorio sabemos que siemrpe va a ser igual a 1 o 0.

Debido a que es siempre cero o uno, podremos encontrar una forma más sencilla de escribir esta función de costes. En en lugar de escribir la función de costes en dos líneas separadas con dos casos separados, donde y es igual a uno e y es igual a cero. Vamos a ver una forma de tomar estas dos líneas y comprimirlos en una ecuación. Y esto haría más conveniente escribir una función de coste y aplicar el descenso del gradiente. Concretamente, podemos escribir la función de costo de la siguiente manera:

1. Sabemos que solo hay dos casos posibles. Y debe ser cero o uno. Entonces supongamos que Y es igual a uno. entonces esta ecuación indica que el costo es igual a
2. Y esta ecuación es exactamente lo que tenemos arriba si y = 1.

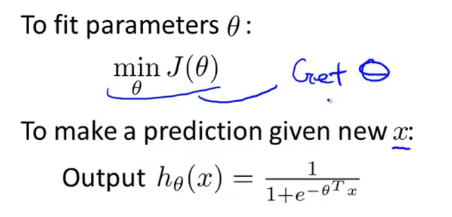
El otro caso es si y = 0. Y si ese es el caso, entonces:

Entonces, esto muestra que esta definición del coste es solo una forma más compacta de tomar estas dos expresiones para los casos donde e , y escribirlos en una forma más conveniente con solo una línea. Por lo tanto, podemos escribir todas nuestras funciones de costos para regresión logística de la siguiente manera:



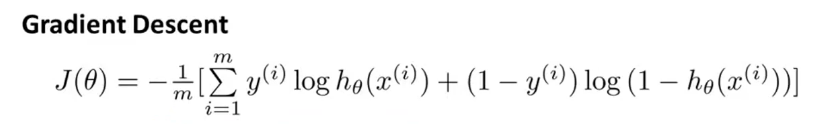
Y simplemente ponemos el signo menos afuera. Y por qué elegimos esta función en particular, mientras que parece que podría haber otras funciones de costos que podríamos haber elegido. Aunque no hay tiempo de entrar en gran detalle de esto en este curso, esta función de costo puede derivarse de la estadística utilizando el principio de estimación de máxima verosimilitud (*maximum likelihood estimation*). Lo cual es una idea en la estadística para hallar de una manera eficiente los parámetros para diferentes modelos con una muestra de datos particular. Y también tiene una propiedad agradable: que es convexa.

Ahora, dada esta función de costes, para ajustar los parámetros, lo que vamos a hacer entonces es tratar de encontrar los parámetros theta que minimizan . Entonces, si tratamos de minimizar la función, esto nos daría algunos parámetros theta. Y si se nos da un nuevo conjunto de datos con las mismas características x podemos tomar las thetas que calculamos con nuestro conjunto de entrenamiento dando como resultado nuestra predicción, que voy a interpretar como la probabilidad de que y sea igual a uno dada la entrada x y parametrizada por theta .

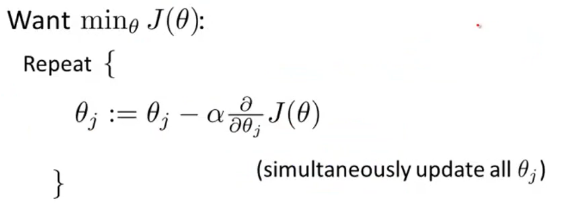


Entonces todo lo que queda por hacer es descubrir cómo minimizar .

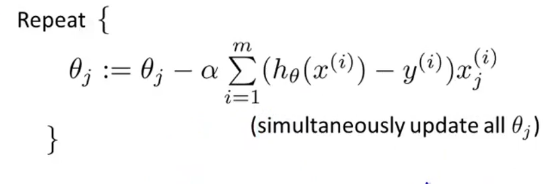
La forma en que vamos a minimizar la función de costo es usando el descenso de gradiente. Aquí está nuestra función de costos de nuevo:



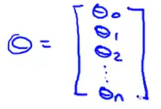
Queremos minimizarla como una función de theta y aquí está nuestra plantilla habitual para descenso gradual donde actualizamos repetidamente cada parámetro actualizándolo:



Si derivamos respecto de θ y sustituimos en nuestra ecuación obtenemos:



Entonces, si tienes n características, tendrías un vector de parámetro theta:

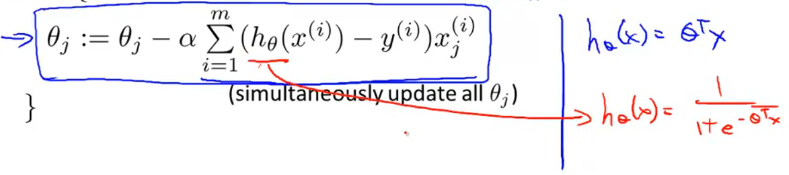


Y usará esta actualización para actualizar todos los de tus valores de theta. Ahora, si toma esta regla de actualización y la compara con lo que estábamos haciendo para la regresión lineal. Puede que se sorprenda al darse cuenta de que esta ecuación, es exactamente lo que teníamos para la regresión lineal. De hecho, si mira el tema anterior, y mira la regla de actualización, la regla del gradiente para la regresión lineal se ve exactamente como la ecuación inmediatamente superior.

Entonces, ¿la regresión lineal y la regresión logística tienen diferentes algoritmos o no? Bueno, esto se resuelve al observar que para la regresión logística, lo que ha cambiado es la definición de la hipótesis.

* Mientras que para la regresión lineal, teníamos ,
* Para la regresión logística ahora es:

Entonces, aunque la regla de actualización se ve cosméticamente idéntica, La definición de la hipótesis ha cambiado.



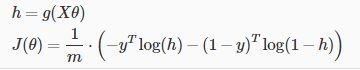
En secciones anteriores, cuando estábamos hablando de descenso de gradiente para Regresión lineal, hablamos sobre cómo monitorear el descenso en gradiente para asegurarnos que está convergiendo. Suelo aplicar el mismo método a la regresión logística, también para controlar un descenso en gradiente, para asegurarse de que está convergiendo correctamente.

Al implementar la regresión logística con el descenso del gradiente, tenemos diferentes valores para los parámetros theta que van de que necesitamos actualizar usando la expresión de arriba. Y una cosa que podríamos hacer es tener un bucle ; que actualice cada uno de estos valores de parámetros a su vez. Pero, por supuesto, en lugar de usar un bucle , idealmente, usaríamos una implementación vectorial para que pueda actualizar todos estos parámetros de una sola vez.

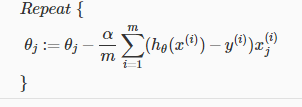
Tenemos nuestra función de costos que es:



Implementada vectorialmente quedaría como:



A su vez, tenemos nuestra función del descenso del gradiente que podemos escribirla como:



Y finalmente, vectorizada:

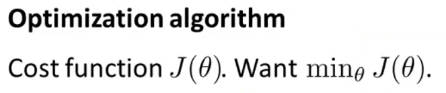
**

Entonces, ahora usted sabe cómo implementar descensos de gradiente para la regresión logística. Hubo una última idea de la que hablamos antes, Regresión lineal, que fue escalado de características. Vimos cómo la escala de las características puede ayudar a que la pendiente de gradiente converja más rápido para regresión lineal La idea de escalar funciones también se aplica al descenso de gradiente para Regresión logística. Y sin embargo, tenemos funciones que están en una escala muy diferente, luego aplicamos la función El escalamiento también puede hacer que el descenso de clasificación se ejecute más rápido para la regresión logística.

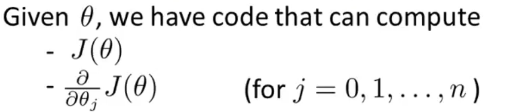
## Optimización avanzada

En la sección anterior hablamos sobre el gradiente de descenso para minimizar la función de costo J de «theta» para la regresión logística. En esta sección, me gustaría hablarte sobre algunos algoritmos avanzados de optimización y sobre algunos conceptos avanzados de optimización. Usando alguna de estas ideas, vamos a poder hacer que la regresión logística se ejecute mucho más rápido que con el gradiente de descenso. Y esto también permitirá que los algoritmos escalen mucho mejor en con un gran número de variables.

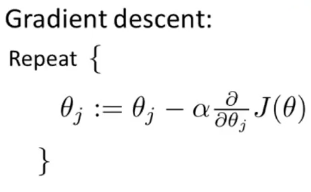
Aquí está una visión alternativa de lo que el gradiente de descenso está haciendo. Tenemos una función de costes J y queremos minimizarla:



Lo que tenemos que hacer es escribir el código que puede tomar como entrada los parámetros «theta» y que pueda calcular dos cosas: J de «theta» y los términos derivados parciales de J igual a 0, 1 hasta N.



El código que pueda hacer estas cosas, realiza de forma repetida la siguiente actualización:



Entonces, dado el código que escribimos para calcular estas derivadas parciales, el gradiente de descenso se inserta multiplicando la tasa de aprendizaje y se utiliza para actualizar nuestros parámetros «theta».

Otra forma de pensar en los gradientes de descenso, es que necesitamos suministrar código para calcular y las derivadas parciales, y luego se insertan en gradientes de descenso, que pueden intentar minimizar la función para nosotros.

Para el gradiente de descenso, creo que técnicamente, en realidad, no necesitas código calcular la función de costo J de «theta». Sólo necesitas código para calcular los términos derivados. Pero si consideras que tu código también monitorea la convergencia de algo así, pensaremos que sólo estamos proveyendo código para calcular la función de costo y las derivadas parciales de .

Así que, después de haber escrito código para calcular estas dos cosas (la función de costes y sus derivadas parciales), un algoritmo que podemos usar es el gradiente de descenso. Pero el gradiente de descenso no es el único algoritmo que podemos usar. Y hay otros algoritmos más avanzados y más sofisticados que son enfoques distintos para optimizar la función de costo para nosotros. Entonces, el gradiente conjugado, BFGS y L-BFGS son ejemplos de algoritmos de optimización más sofisticados que requieren de una forma para calcular J de «theta», y que requieren una forma para calcular las derivadas.

Los detalles de lo que son estos tres algoritmos en profundidad está más allá del alcance de este curso. Pero vamos a hablar sobre algunas de sus propiedades.

Estos tres algoritmos tienen varias ventajas. Una es que, con cualquiera de estos algoritmos generalmente no necesitas seleccionar manualmente el «alfa» de la tasa de aprendizaje. Entonces, una forma de considerar estos algoritmos es que es la forma dada para calcular la derivada y la función de costes, tienen un ciclo inteligente llamado algoritmo de búsqueda de línea que automáticamente prueba distintos valores para el «alfa» de la tasa de aprendizaje y, automáticamente selecciona un buen «alfa» de forma que incluso puede seleccionar una tasa de aprendizaje diferente para cada iteración. Y entonces no tienes que elegirlo tú mismo.

Estos algoritmos, de hecho, hacen cosas más sofisticadas que sólo elegir un buen «alfa» de la tasa de aprendizaje, y por lo tanto, con frecuencia terminan convergiendo mucho más rápido que el gradiente de descenso.

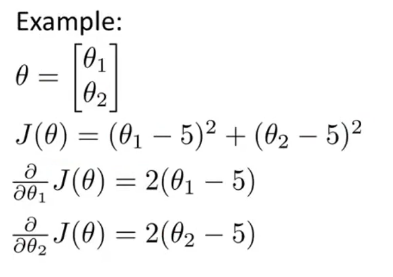
Si estos algoritmos tienen una desventaja, yo diría que la principal desventaja es que son mucho más complejos que el gradiente de descenso. Y, en particular, probablemente no deberías implementar estos algoritmos - gradiente conjugado, L-BGFS, BFGS - tú mismo a menos que seas experto en cálculo numérico computacional.

En cambio, así como yo no te recomendaría que escribas tu propio código para calcular raíces cuadradas de números o para calcular matrices inversas, para estos algoritmos, lo que también te recomendaría es sólo usar una librería de software. Así que ya sabes, para sacar una raíz cuadrada, lo que todos hacemos es usar alguna función que alguien más ha escrito para calcular las raíces cuadradas de los números.

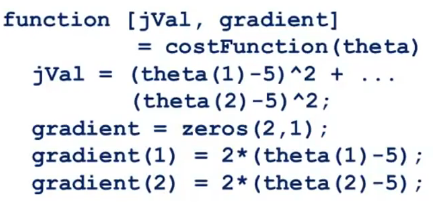
Y, afortunadamente, Octave tiene una muy buena librería que implementa algunos de estos algoritmos de optimización avanzados. Y si sólo usas la librería integrada, ya sabes, tendrás muy buenos resultados.

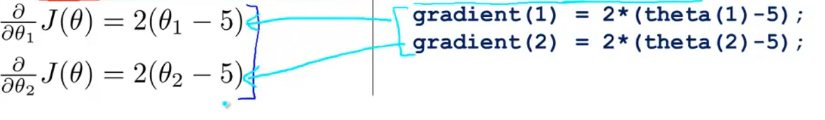
Debo decir que hay diferencias entre buenas y malas implementaciones de estos algoritmos. Entonces, si estás utilizando un lenguaje diferente para tu aplicación de aprendizaje automático, si estás usando C, C++, Java, y debes probar un par de librerías diferentes para asegurarte de encontrar una buena librería para implementar estos algoritmos. Porque hay una diferencia de desempeño entre una buena y una mala implementación de, ya sabes, un gradiente de conjugado o BFGS contra una implementación menos buena de un gradiente de alguna de ellas.

Entonces, ahora voy a explicar cómo usar estos algoritmos, y voy a hacerlo con un ejemplo:



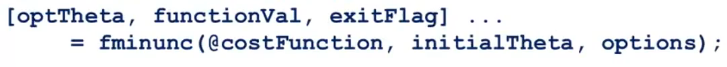
Entonces, si quieres aplicar uno de los algoritmos avanzados de optimización para minimizar la función de costo, usando algo como el gradiente descenso, pero preferiblemente más más avanzado que el gradiente de descenso, lo que harías sería implementar una función de Octave como ésta:



Implementamos una función de y lo que esto hace es devolver dos argumentos, el primer J-val es como nosotros calcularíamos la función de costo J. Y entonces, esto dice que J-val es igual a  . 

El segundo argumento que devuelve esta función es un gradiente. Entonces, el gradiente va a ser un vector de dos por uno con valores iniciales igual a cero, y los dos elementos del vector gradiente corresponden a los dos términos de la derivada parcial:

Después de haber implementado nuestra función de costes, puedes entonces llamara la función de optimización avanzada cuyo comando es (que se refiere a la minimización sin restricciones de la función en Octave) y la forma de llamarla es la siguiente:



Vemos que esta función necesita diferentes parámetros para ejecutarse:

Configuras algunas opciones. Estas opciones son como una estructura de datos que almacena las opciones que quieras:



Entonces, con los dos primeros parámetros dices que la función va a recibir de entrada un objeto que es el gradiente. Esto sólo significa que, de hecho, vas a proporcionar un gradiente a este algoritmo. Y después voy a definir el número máximo de iteraciones a, digamos, cien.

Además de esto debo darle una suposición inicial de «theta»:



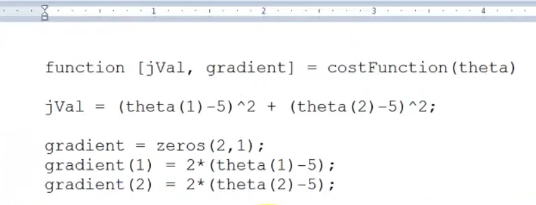
Donde escribo vector de 2 por 1. Y por últim:



Que le indica a la función de minimización que es una función de costes lo que queremos calcular y que hemos definido previamente manualmente mediante nuestra función que acabamos de definir arriba.

Esta función usará uno de los algoritmos de optimización más avanzados. Y si quieres, puedes considerarlo sólo como un gradiente de descenso con esteroides que elige automáticamente el «alfa» de la tasa de aprendizaje para que no tengas que hacerlo tú mismo para tratar de encontrar el valor óptimo de «theta» para usted. Vamos a ver cómo se ve esto en Octave.

Entonces, escribí esta función de costes de la función de «theta» exactamente como la teníamos arriba:

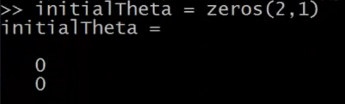


Donde tenemos J-val, que es la función de costes y el gradiente con los dos elementos con las derivadas parciales con respecto a .

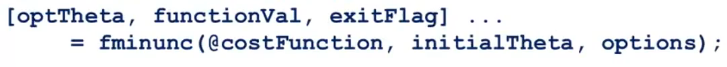
Establecemos las opciones



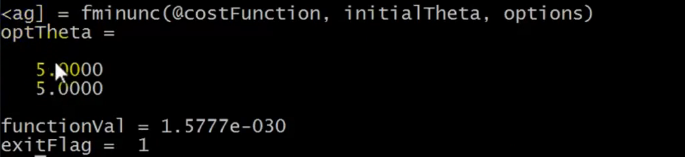
Y damos las thetas iniciales:



Y una vez tengo todos mis parámetros de entrada, escribo mi función de minimización:



Y pulso enter:

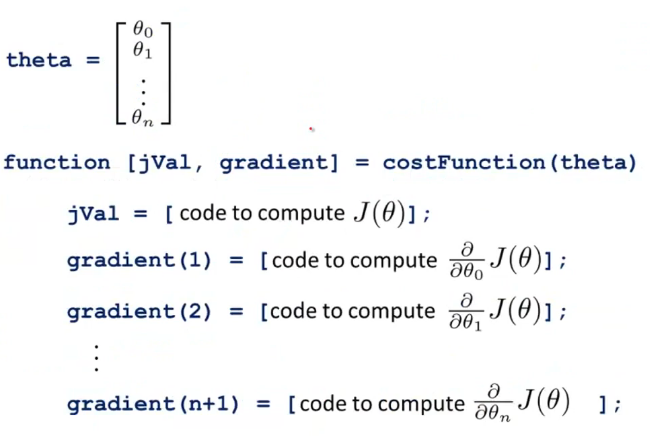


Lo que esto significa es, que encontró el valor óptimo para A «theta» es «theta» 1 igual a 5, «theta» 2 es igual a 5, exactamente como lo esperábamos. El valor de la función en el óptimo es esencialmente 10 a la menos 30. Que es prácticamente cero, que también es lo que estábamos esperando. Y la bandera de salida es 1 y esto muestra cuál es el estado de convergencia de esto. Y si quieres, puedes ingresar para leer la documentación para saber cómo interpretar la bandera de salida. Pero la bandera de salida te permite verificar si este algoritmo ha convergido.

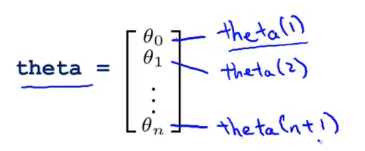
Así es cómo se ejecutan estos algoritmos en Octave.

Debo mencionar, por cierto, que para la implementación en Octave, este valor de «theta», tu vector parámetro de «theta», debe estar en . Entonces, si «theta» sólo es un número real. Y, si no lo es, al menos un vector de dos dimensiones o un poco superior al vector bidimensional, este podría no funcionar, y en el caso de que tengas una función unidimensional que utilizas para optimizar, puedes consultar la documentación de Octave sobre para obtener más detalles.

Entonces, así es cómo optimizamos nuestro ejemplo prueba de esta simple función de costo de forma rápida. Sin embargo, aplicamos esto, digamos, la regresión logística en la que tenemos un vector de parámetro «theta», y voy a usar una mezcla de notación de Octave y un tipo de notación matemática. Pero espero que esta explicación sea clara;



Primero, tenemos nuestro vector de parámetro «theta» que comprende los parámetros «theta» 0 hasta «theta» n, (los índices de Octave para los vectores usan la indexación de 1 en adelante entonces, «theta» 0, realmente, se escribe «theta» 1 y, «theta» 1 se va a escribir como «theta» 2 y finalmente «theta» n será «theta» n + 1.Y esto se debe, como hemos dicho a que Octave indexa los vectores a partir del índice de 1 como el índice de 0.



Entonces, lo que debemos hacer es escribir una función de costes que capture la función de costes para la regresión logística. Concretamente, la función de costes debe devolver J-val (que es J(θ)),y también necesitamos darle el gradiente. Entonces, el gradiente 1 va a ser algún código para calcular la derivada parcial respecto a «theta» 0, la siguiente derivada parcial respecto a «theta» 1 y así sucesivamente. Una vez más, estos son el gradiente 1, el gradiente 2 y así sucesivamente, en lugar de gradiente 0, gradiente 1,porque los índices de Octave son vectores que comienzan desde uno en lugar de desde cero.

Pero el concepto principal que espero que comprendas de esta diapositiva es que, lo que tienes que hacer, es escribir una función que devuelva la función de costes y el gradiente. Para aplicar esto a la regresión logística o, incluso a la regresión lineal.

Lo único que tienes que hacer es insertar el código apropiado para calcular función de costes y las derivadas parciales respecto a la misma.

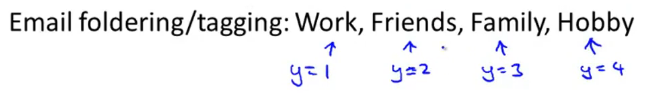
Ahora ya sabes cómo usar estos algoritmos avanzados de optimización. Porque, para estos algoritmos estás usando una librería de optimización sofisticada, que lo hace un poco más opaco y, quizás un poco más difícil de depurar. Pero como estos algoritmos suelen ejecutarse mucho más rápido que el gradiente de descenso, a menudo muy típicamente cuando tengo un problema de aprendizaje automático muy grande, usaré estos algoritmos en lugar de usar el gradiente de descenso.

Y, con estas ideas, espero que puedas poner a trabajar la regresión logística y también la regresión lineal en problemas mucho más grande. Entonces, eso es todo en cuanto a los conceptos de optimización.

## Regresión logística Multi – Clase

En esta sección definiré cómo lograr que la regresión logística funcione para problemas de clasificación multiclase, y de manera particular quiero hablarte sobre un algoritmo llamado clasificación uno-contra-todos. ¿Qué es un problema de clasificación multiclase? He aquí algunos ejemplos:

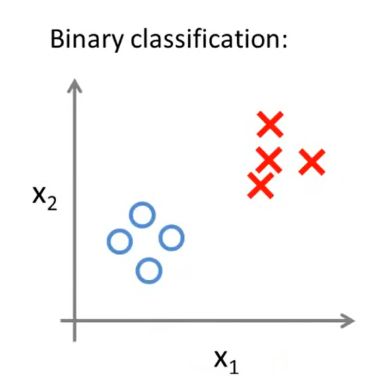
Supongamos que quieres que un algoritmo de aprendizaje coloque automáticamente tu correo electrónico en carpetas diferentes o etiquete automáticamente tus correos electrónicos. Entonces, posiblemente tendrás diferentes carpetas o diferentes etiquetas para el correo electrónico del trabajo, correo electrónico de tus amigos, correo electrónico de tu familia y correos sobre tu pasatiempo. Así pues, aquí tenemos un problema de clasificación con 4 clases, a las que podríamos asignar los números, las clases .

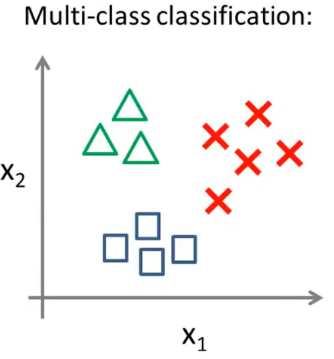


Otro ejemplo, para un diagnostico médico: si un paciente llega a tu oficina con tal vez la nariz congestionada, los posibles diagnósticos podrían ser que no está enfermo, tal vez eso es y=1; o que tiene un resfriado, 2; o que tiene gripe.

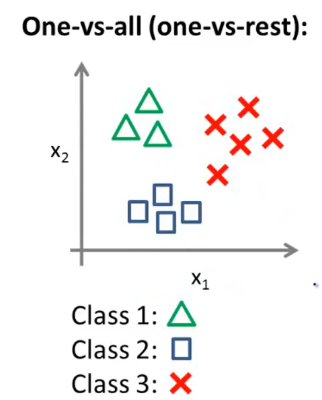
Y el tercer y último ejemplo, si estás utilizando el aprendizaje automático para clasificar el clima, ya sabes, tal vez quieres decidir si el clima es soleado, nublado, lluvioso o con nieve, o si va a nevar. Y así, en todos esos ejemplos, "y" puede tomar un número reducido de valores discretos, tal vez 1, 2 3, de 1 a 4 y así sucesivamente, y estos son problemas de clasificación multiclase. Por cierto, en realidad no importa si indexamos ya sea como 0 1 2 3 o como 1 2 3 4, tiendo a indexar mis clases a partir del 1 en lugar de empezar desde 0. Pero de cualquier manera, realmente no importa.

Mientras que anteriormente, para un problema de clasificación binaria, nuestro conjunto de datos se veía así:

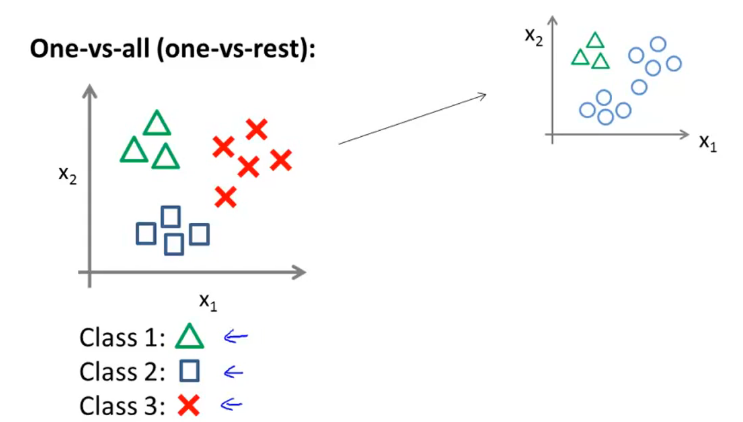


Para un problema de clasificación multiclase, nuestro conjunto de datos puede verse así:

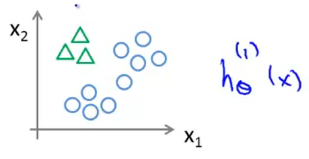
Donde estoy usando tres símbolos diferentes para representar nuestras tres clases. Entonces, la pregunta es: Dado un conjunto de datos con tres clases ¿Cómo obtenemos un algoritmo de aprendizaje que trabaje para este escenario? Ya sabemos cómo hacer la clasificación binaria, utilizando la regresión logística. Usando una idea llamada clasificación uno contra todos, podemos entonces tomar esto, y, hacerlo funcionar para la clasificación multiclase también. Así es como funciona la clasificación uno contra todos. Y, esto es también llamado a veces "uno contra el resto." Digamos que tenemos un conjunto de entrenamiento, como se ve así:



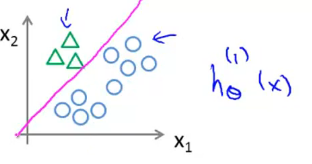
Donde tenemos 3 clases. Así que, si y=1, denotamos eso con un triángulo, si y=2 con un cuadrado, si y=3 entonces, una cruz. Lo que vamos a hacer es, tomar un conjunto de entrenamiento, y convertir esto en tres problemas de clasificación binaria separados. Entonces, voy a convertir esto en tres problemas separados de clasificación de dos clases. Vamos a empezar con la clase 1, que es un triángulo. Vamos a crear esencialmente un nuevo conjunto de entrenamiento falso, en donde las clases 2 y 3 quedan asignadas a la clase negativa y la clase 1 queda asignada a la clase positiva:



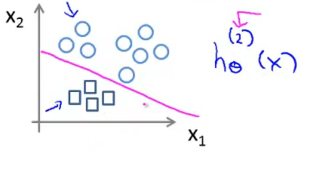
Cuando creamos un nuevo conjunto de entrenamiento, como el que se muestra a la derecha, vamos a ajustar un clasificador, que voy a llamar "h" subíndice «theta» superíndice 1 de "x" en donde aquí, los triángulos son los ejemplos positivos y los círculos son los ejemplos negativos.



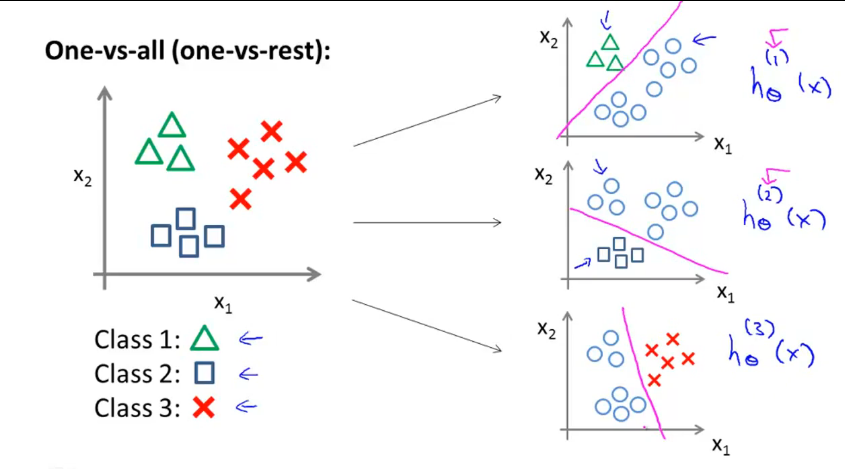
Por lo tanto, piensa que a los triángulos se les ha asignado el valor 1 y a los círculos la suma de las otras dos clases, el valor de cero. Y sólo vamos a intentar modelar un problema de regresión logística estándar y tal vez eso nos dará la siguiente solución:



¿De acuerdo? El superíndice 1 de aquí es la clase uno. Por lo tanto, estamos haciendo esto para la primera clase que es el triángulo. A continuación, hacemos lo mismo para la clase 2. Vamos a tomar los cuadrados y vamos a asignar los cuadrados a la clase positiva y asignamos todo lo demás, los triángulos y las cruces, a la clase negativa. Y después ajustamos un segundo clasificador de regresión logística que voy a llamar a este "h" de "x" superíndice 2, en el que el el superíndice 2 indica que ahora estamos haciendo esto: tratando a la clase del cuadrado como clase positiva y tal vez podemos obtener el clasificador así:



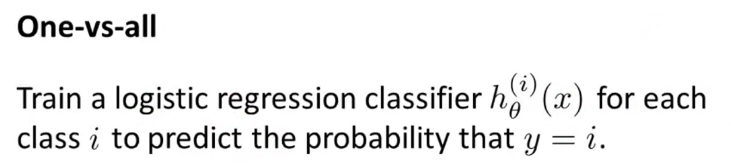
Y por último, hacemos lo mismo para la tercera clase y ajustamos un tercer clasificador "h" superíndice 3 de "x" y tal vez esto nos dará una decisión delimitadora o nos dará un clasificador que separe los ejemplos positivos y negativos así. Así que, para resumir, lo que hemos hecho es ajustar 3 clasificadores. Para i = 1, 2, 3. Ajustamos un clasificador "h" superíndice "i" subíndice «theta» de "x", tratando así de calcular cual es la probabilidad de que "y" es igual a la clase "i" dado que "x" está parametrizada por «theta».



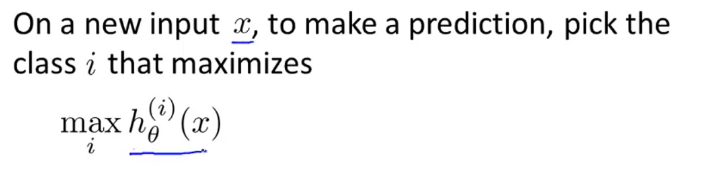
Así, en la primera instancia el clasificador va aprendiendo a identificar los triángulos. Así es que piensa en los triángulos como una clase positiva. Por lo que "h" superíndice 1 está esencialmente tratando de calcular cuál es la probabilidad de que "y" es igual a 1, dado que "x" está parametrizada por «theta». Y de manera similar, este está tratando, como sabes, la clase de los cuadrados como una clase positiva por lo que está tratando de calcular la probabilidad de que y=2 y así sucesivamente. Así que ahora tenemos 3 clasificadores, cada uno de los cuales fue entrenado para identificar una de las tres clases.



Sólo para resumir, lo que hemos hecho es que, queremos entrenar a un clasificador de regresión logística, "h" superíndice "i" de "x", para cada clase "i" para predecir la probabilidad de que y=i.

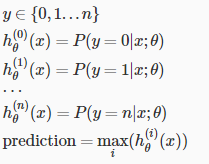


Por último, para hacer una predicción cuando hemos agregado una nueva entrada "x" y queremos hacer una predicción, lo que hacemos es simplemente ejecutamos 3 de nuestros clasificadores con los datos de la nueva entrada "x" y después tomamos la clase "i" con la mayo probabilidad.



Así que básicamente elegimos el clasificador que sea más confiable, o que de forma más entusiasta diga que piensa que tiene una clase correcta. Cualquier valor de "i" que nos de la mayor probabilidad, entonces predecimos que "y" tendrá ese valor.

Como y = {1, ... n}, dividimos nuestro problema en n problemas de clasificación binaria; en cada uno, predecimos la probabilidad de que 'y' sea miembro de una de nuestras clases.



Básicamente estamos eligiendo una clase y luego agrupando a todas las demás en una sola segunda clase. Hacemos esto repetidamente, aplicando regresión logística binaria a cada caso, y luego usamos la hipótesis que arrojó el valor más alto como nuestra predicción.

Eso es todo sobre la clasificación multiclase y el método uno contra todos. Y con este pequeño método ahora puedes tomar el clasificador de regresión logística y hacerlo funcionar también para los problemas de clasificación multiclase.