EL ALGORITMO DEL GRADIENTE DESCENDENTE

ÍNDICE:

[**INTRODUCCIÓN** 2](#_Toc79830898)

[**1.- ALGORITMO DEL GRADIENTE DESCENDENTE EN DIMENSIÓN DEFINICIONES Y CONCEPTOS** 2](#_Toc79830899)

[**2.- ALGORITMO DEL GRADIENTE DESCENDENTE** 5](#_Toc79830900)

[**3.- EXISTENCIA DE MÁXIMOS Y MÍNIMOS ABSOLUTOS EN FUNCIONES MULTIVARIABLES** 7](#_Toc79830901)

[**4.- EL MACHINE LEARNING ACTUAL Y EL GRADIENTE DESCENDENTE** 8](#_Toc79830902)

[**5.- VARIANTES DEL a. g. d. I EL GRADIENTE ESTOCÁSTICO** 9](#_Toc79830903)

[**6.- VARIANTES DEL a. g. d. II EL GRADIENTE DE DESCENTO CON MOMENTO** 10](#_Toc79830904)

[**7.- CONCLUSIONES FINALES SOBRE EL ALGORITMO DEL GRADIENTE DESCENDENTE** 11](#_Toc79830905)

# **INTRODUCCIÓN**

Uno de los matemáticos clave en la historia es sin duda Agustin Louis Cauchy (1789 – 1857), con él se fundamenta y se define la noción de número real a través del concepto que actualmente se conoce como *sucesión de Cauchy* que es una sucesión que cumple:

En este sentido resultan interesantes las propiedades de estas sucesiones que se resumen en las 3 siguientes:

* Toda sucesión convergente es una sucesión de Cauchy
* Toda sucesión de Cauchy está acotada
* **Criterio de convergencia de Cauchy**: una sucesión de números reales es convergente, sí y sólo sí es de Cauchy

Básicamente el criterio de convergencia de Cauchy lo que hace es conectar el análisis matemático con la topología. Resulta por tanto que Cauchy es muy conocido en el mundo de las matemáticas, tanto por la gran cantidad de resultados que demuestra, como por su libro *Análisis Infinitesimal* publicado en 1814. Pero por lo que se trae su figura aquí es porque fue el inventor del *Método de Gradiente Descendente* en 1847 en su Compte Rendu da l’Acadèmie des Sciences, cuando se hallaba estudiando un problema de Astronomía.

# **1.- ALGORITMO DEL GRADIENTE DESCENDENTE EN DIMENSIÓN 1. DEFINICIONES Y CONCEPTOS**

El algoritmo a analizar es de tipo iterativo, como se desarrolla a continuación, este algoritmo va a permitir encontrar valores mínimos (o máximos en el caso del ascendente), de funciones convexas y diferenciables en todo su dominio. El nombre proviene del interés de encontrar el mínimo de una función a partir de un punto *P* de su dominio, lo que se hace es que se toman “pasos” proporcionales al sentido opuesto del gradiente de la función en dicho punto *P*. Pero antes de describir su funcionamiento conviene ver unas pocas definiciones previas que permitan asentar el concepto de lo que se quiere explicar.

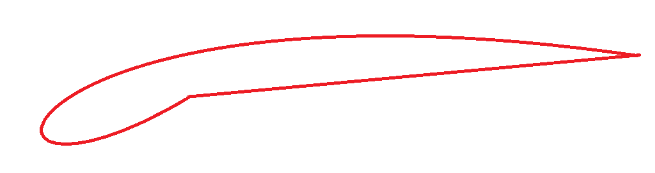
Definición: Conjunto convexo

Un conjunto *C* se dice convexo si para cualquier par de puntos e pertenecientes a *C* se verifica que el segmento formado por dichos puntos queda siempre dentro de *C*

\_\_\_\_\_

Ejemplo 01: Ejemplo de un conjunto no convexo

La circunferencia, el cuadrado, el rombo, … son conjuntos convexos, sin embargo, la siguiente figura no lo es:



\_\_\_\_\_

Definición: Función convexa

Una función se denomina convexa dentro de un sub-conjunto convexo *C* si para cualesquiera 2 puntos, e pertenecientes a *C* se verifica la siguiente propiedad:

\_\_\_\_\_

Definición: Derivada de una función real de variable real en un punto *x*0

Se denota con derivada de una función de variable real *x* en el punto *x*0 al siguiente límite:

Nótese que en el denominador se tiene que:

\_\_\_\_\_

Así pues, la derivada de una función en un punto es el cociente entre el incremento de la función en entornos del punto muy pequeños con respecto al cambio incrementar en dicho punto de modo qué, si es suficientemente pequeño, ese cociente tiende a un número que va a coincidir con la pendiente de la tangente en dicho punto. Así pues, en funciones que transforman un espacio de dimensión 1 en un subconjunto de la recta real y que pueden por tanto ser representadas en 2 dimensiones, el valor de la derivada ofrece información muy valiosa:

* Por un lado, su magnitud habla sobre la intensidad de la curvatura de la función en ese punto. Una elevada magnitud implica que la curvatura de la función es muy elevada y una magnitud nula o cercana a 0 implica que la función podría estar ante un máximo o un mínimo local (entre otros tipos de puntos críticos), que es lo que va a interesar
* Y el signo de la derivada va a indicar hacia dónde (izquierda o derecha) se encontraría ese mínimo o máximo local (o absoluto)

Gráfico, Gráfico de líneas

Descripción generada automáticamente

**Figura 2 Interpretación geométrica de la derivada en el punto *x*0 = *a***

**Fuente:** [**https://www.superprof.es/apuntes/escolar/matematicas/calculo/derivadas/concepto-de-derivada.html**](https://www.superprof.es/apuntes/escolar/matematicas/calculo/derivadas/concepto-de-derivada.html)

Por tanto, en funciones convexas reales de variable real se cumpliría el siguiente resultado:

TEOREMA (Gradiente descendente en dimensión 1)

     Sea una función convexa de clase con un mínimo en . Para encontrar dicho punto puede construirse una sucesión de puntos tal que converge a tal como sigue:

Donde el parámetro (que puede tomarse como una constante entre 0 y 1), se selecciona de tal manera que y

\_\_\_\_\_

Como se observa en este resultado, el algoritmo que se plantea va, para funciones reales de variables reales, dando “pasitos” de aproximación hacia un mínimo que puede ser local o absoluto

Ejemplo 02: Cálculo de mínimos en funciones reales de variable real

En el código que se encuentra en *ejemplos/programa\_0101.py* se observa el caso de la función *x ·* sin(*x*) que ayudará a recordar qué se entiende por mínimo local, mínimo global además de cómo con distintas inicializaciones, el algoritmo puede dar un resultado u otro

Gráfico, Gráfico de líneas

Descripción generada automáticamente

|  |  |
| --- | --- |
|  |  |

\_\_\_\_\_

# **2.- ALGORITMO DEL GRADIENTE DESCENDENTE**

El algoritmo del gradiente descendente descrito anteriormente es generalizable a dimensiones más elevadas, con tan sólo introducir un nuevo concepto, la potencia de este algoritmo radica en que es capaz de extraer mínimos de funciones de coste independientemente de la dimensión que sea, así pues, cuando se tiene una regresión lineal múltiple, las funciones de coste dependerán de unos *p* + 1 parámetro y por tanto el problema se reduce a encontrar esos parámetros que por ejemplo sean capaces de minimizar una función tal como la que sigue:

Para poder dar el paso a la generalización de la derivada, se necesita un nuevo concepto:

Definición: Gradiente de una función multivariable real en un punto

Se denota como gradiente de una función multivariable en el punto a la siguiente expresión:

\_\_\_\_\_

Aunque la anterior expresión se puede desarrollar más se sobre-entiende que lo que se está haciendo es derivar parcialmente respecto a cada una de sus componentes a la función *f*, para finalmente asignarle el valor del vector o punto con respecto al cual se deriva, así pues, si la anterior función de coste tuviera 2 componentes (la constante y un término pendiente), la expresión anterior quedaría como:

Y por tanto su función gradiente sería el siguiente vector genérico (si no se fija un punto paramétrico):

Una vez obtenido lo anterior se puede crear un proceso de actualización (simplificando un poco) y de convergencia hacia el mínimo del siguiente modo:

Básicamente este proceso iterativo es prácticamente idéntico al anterior, por lo que cabe enunciar el siguiente teorema general:

TEOREMA (Generalización del Teorema del Gradiente Descendente)

Sea una función convexa de clase con un mínimo en . Para encontrar dicho punto puede construirse una sucesión de puntos tal que converge a tal como sigue:

Donde el parámetro se selecciona de tal manera que y

En una implementación práctica, en general el parámetro se toma con un valor constante entre 0 y 1 y se establece un criterio de parada tal como sigue:

\_\_\_\_\_

Ejemplo 03: Gradiente descendente en la regresión lineal

En el código de *ejemplos/programa0102.py* se programa y se comparan los resultados que se obtiene de un *gradiente descendente*, frente a una implementación Python de una regresión lineal

En el siguiente fragmento de código se muestra “el corazón” de la implementación del método del gradiente descendente. Obsérvese como se implementa la función de coste considerada y sus correspondientes derivadas parciales:

Texto

Descripción generada automáticamente

\_\_\_\_\_

# **3.- EXISTENCIA DE MÁXIMOS Y MÍNIMOS ABSOLUTOS EN FUNCIONES MULTIVARIABLES**

Para tener una idea más cercana de lo hecho anteriormente y de la potencia de este método, conviene tener una idea de la representación de las funciones de coste que se están tratando.

En el ejemplo considerado en el punto 2, la función de coste considerada, para el caso de 2 variables es la siguiente:

Si se quiere representar como una superficie donde el eje *z* sea el coste y la base las 2 variables se llega a la siguiente representación, suponiendo que se tiene los siguientes 5 pares de observaciones : ; a casos como a los que se desarrolla en el siguiente ejemplo

Ejemplo 04: Función de coste asociada a 5 puntos (modelo lineal unidimensional)

En el código de *ejemplos/programa0103.py* se dibuja la función de coste asociada, se observa que no tiene “valles” ni “montañas”, es una función de coste “lisa” donde se podría demostrar que existe y es único el mínimo y donde por gradiente descendente (o mínimos cuadrados) se podría llegar a que dicho mínimo se encontraría en llegando a valer dicho mínimo 0.540945 Gráfico, Gráfico de superficie

Descripción generada automáticamente

\_\_\_\_\_

Pero ¿Qué sucede si al anterior ejemplo se añade un término de regularización como pueda ser el de tipo l2? En este caso la metodología funciona, sólo que la función de coste a minimizar adopta una forma más compleja tal como la que ya se analizó:

En este caso, hay que fijar un valor para lambda para que realmente el algoritmo haga una regularización, si no el gradiente descendente dará como solución la del caso anterior ya que se verifica trivialmente que:

Ejemplo 05: Función de coste asociada a 5 puntos (modelo lineal unidimensional) con regularización l2 y

El código *ejemplos/programa0104.py* está preparado para hacer distintos experimentos de manipulación con el valor de lambda anterior, de hecho, se observa que si lambda es muy grande, el mínimo estaría cercano a un valor donde los coeficientes estarían próximos a 0 y que sería 0 si se cambia la suma cuadrática por el valor absoluto en los parámetros.

\_\_\_\_\_

# **4.- EL MACHINE LEARNING ACTUAL Y EL GRADIENTE DESCENDENTE**

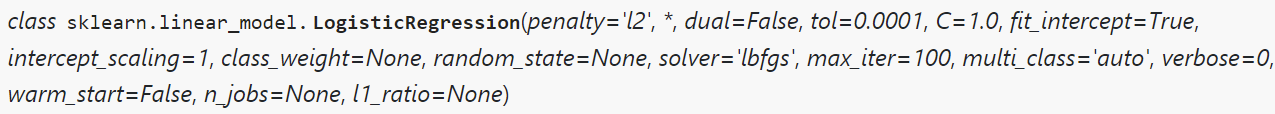
En muchas implementaciones del ML actual el método que está por debajo en términos de cálculo de coeficientes es un algoritmo del gradiente descendente con alguna modificación para hacer que su velocidad de convergencia sea aún mayor. Esto sucederá en algoritmos como la regresión línea, logística con o sin regularizar e incluso con las redes neuronales y el Deep - Learning donde la variante asociada toma el conocido nombre de Back - Propagation, pero se basa en la misma idea de minimización de errores a partir de pequeños pasos dados mediante un vector tipo gradiente.

*Python* por ejemplo resulta muy proclive a dotar a sus algoritmos vía la librería *sklearn* de *gradientes descendentes*. Esto tiene una gran ventaja sobre hacer uso de una fórmula cerrada, como la solución del algoritmo de mínimos cuadrados y es que en general el proceso acabará y no generará aparentemente ningún error. De hecho, en ocasiones puede llegar a ser hasta peligroso el aplicar regresiones con *Python* y no haber tenido un estudio previo de correlaciones entre variables, ya que como se ha mostrado anteriormente, cuando se impone como criterio de parada que el a. g. d. converja o que alcance un máximo en número de iteraciones, esto último siempre va a suceder y al final dará un determinado valor tal y como se observa en el siguiente ejemplo sencillo.

Ejemplo 06: Multicolinelaidad y a. g. d. comparación estadística *R* vs *Python*

En los códigos *ejemplos/programa0105.py* y *programa0105.R* se observa qué sucede cuando existe una variable perfectamente multicolineal tanto en *Python* como en *R*. En Python se observa que se alcanza unos valores finales y depende qué librería se use avisará de un modo más o menos directo, en *R*, cuando se lleva a cabo una regresión lineal, indica con un NA que uno de los 2 parámetros no es estimable y por tanto no lo calcula.

En el caso de que se use la librería *sklearn* debe ponerse especial atención ya que por defecto si no se indica nada, en el método *LogisticRegression* se va a llevar a cabo una *regularización de tipo l2* y con un solver de tipo *lbfgs* por defecto



\_\_\_\_\_

# **5.- VARIANTES DEL a. g. d. I EL GRADIENTE ESTOCÁSTICO**

Una variante interesante del a. g. d. es la denominada del gradiente estocástico. Este caso es especialmente recomendable cuando se tiene un elevado número de datos. En este caso, su aplicación es especialmente efectiva cuando la función de costes se puede descomponer en un número de sumandos que en general va a ser numeroso como ocurre por ejemplo con la siguiente expresión:

En este caso existen *m* sumandos y la aplicación de un gradiente descendente, cuando se habla de cientos de millones de datos, implicaría la realización de una suma muy larga para cada paso en la convergencia.

Donde la *E* denota el término de Esperanza Matemática y elementos de una población de sumandos que cuando hay muchos se puede tomar una submuestra aleatoria teniendo similares propiedades de convergencia hacia el mínimo local, con esto se tienen las siguientes ventajas del s. g. d.:

* Si la función objetivo es la suma de costes individuales (errores) sobre un conjunto muy grande de datos. La muestra suele ser representativa y producir un valor muy cercano al de la población
* Se reduce el número de cálculos en cada iteración
* Cuando hay datos atípicos ( *outliers* ), las muestras pueden ser robustas a esas “pocas” grandes desviaciones (salvo en aquellas muestras que sean incluidos, que se esperan sean pocas)
* Si la función objetivo es (ruidosa, tiene muchos mínimos locales pequeños). El gradiente estocástico permite *suavizar* la función objetivo y reduce el riesgo de tener una convergencia temprana

Aunque como todo algoritmo, no está exento de desventajas como las siguientes:

* El efecto de los *outliers* en el gradiente de una muestra puede afectar más fuertemente y desviar al algoritmo de su trayectoria de convergencia

# **6.- VARIANTES DEL a. g. d. II EL GRADIENTE DESCENDENTE CON MOMENTO**

El a. g. d. en su estado más puro directamente dirige la búsqueda hacia un mínimo que como se observa en superficies de mayor complejidad y en dimensión muy elevada como los que van a ser habituales en los problemas de redes neuronales y superiores, provoca que, con una alta tasa de probabilidad, el proceso de optimización se encuentre atrapado en un mínimo local, muy lejos de lo que podría ser un óptimo global (o al menos suficientemente bueno).

Así pues, una idea que se introduce en este tipo de algoritmos es la de tener “un recuerdo del instante anterior”, de modo que la actualización, del algoritmo original, con la notación utilizada hasta el momento quedaría como:

Por lo que la actualización a llevar a cabo sería la siguiente:

El coeficiente de momento, a diferencia del coeficiente de aprendizaje, se suele elegir cercano a 1 con valores en torno a 0.9, suponiendo como en la mayoría de las aplicaciones a constante con valores sobre 0.1, se tiene que la siguiente regla de actualización es equivalente a:

En esta expresión se observa que se tiene una media ponderada exponencialmente del gradiente con lo que, si se mueve hacia la dirección correcta, el avance sea más rápido con una serie de “rebotes” que se atenuarían con el paso del tiempo hasta prácticamente eliminarse, por este motivo, este algoritmo puede ser más rápido en ocasiones que el a. g. d. en su estado más puro.

Ejemplo 07: Implementación sencilla en python del a. g. d. con momento

En el *ejemplos/programa0106.py*, se observa, en un caso muy sencillo cómo es la aceleración que puede llegar a producir el a. g. d. cuando no se hace uso del concepto momento, frente a utilizarlo:

|  |  |
| --- | --- |
| Gráfico, Gráfico de líneas  Descripción generada automáticamente | Gráfico, Gráfico de líneas  Descripción generada automáticamente |

Nótese como en la figura de la izquierda se nota la aceleración y la convergencia hacia el correspondiente mínimo de un modo mucho más rápido que en el de la derecha, con pasos más largos si cabe.

*\_\_\_\_\_*

# **7.- CONCLUSIONES FINALES SOBRE EL ALGORITMO DEL GRADIENTE DESCENDENTE**

EL a. g. d. y aquellos que están basados en su idea tienen unas ventajas en cómputo y convergencia superiores a los algoritmos tradicionales que, aunque ofrezcan fórmulas cerradas, no permiten su aplicación a situaciones más generales.

Es cierto que con el a. g. d. se pierde alertas y cierta coherencia cuando existen variables problemáticas en los dataset con alta multicolinealidad, o con outliers, pero es el analista quien debe valorar, estudiar e incluso modificar si corresponde los valores de dichas variables para que los procesos funciones sin problemas.

El gradiente descendente permite generalización de los métodos en los que se aplican, ya da igual la forma funcional de la función de coste que se trate, sencillamente interesa saber si dicha función resulta convexa en el intervalo bajo estudio y si tiene derivadas parciales de orden 1 al menos, a partir de ahí se puede dotar al método de convergencia hacia al menos un mínimo local.

Debe tenerse en cuenta que debido a que se va a aplicar a funciones, donde en ocasiones su estructura geométrica puede ser muy compleja, no siempre se va a tener garantizado un mínimo absoluto sobre el cual converger siendo importante el establecimiento de una tasa de aprendizaje suficientemente adecuada, e incluso en algunos casos se puede acelerar la convergencia si se eligen los puntos iniciales con cierta heurística que dependerá del tipo de problema que se trate.

Con todo lo dicho, el a. g. d. permite pasar de la estadística tradicional dominada por el anteriormente comentado Teorema de Gauss-Markov a una nueva generación de algoritmos con mayor grado de sofisticación y a metodologías que superan la tradicional revisión de cumplimiento de hipótesis gracias a la superación de logros predictivo sobre datasets de prueba adecuados y numerosos.

Finalmente indicar que existen numerosas variantes y parámetros adicionales, aparte del parámetro de aprendizaje y de momento que se pueden introducir en este algoritmo, dando lugar a toda una amplia familia que llega hasta los conocidos RMSProp (Root Mean Square Propagation), ADAM (Adaptative Moment Estimation), etc. Toda una serie de algoritmos donde es recomendable echar un vistazo para ver cómo funcionan, sobre todo cuando se tenga algún problema donde los resultados obtenidos no sean los esperados, o donde los tiempos de cómputo sean demasiado elevados y haya que dejar las opciones por defecto que ofrezca cualquier paquete informático y entonces se deba “tomar los mandos de la aeronave” que es para lo que se supone que deben estar preparados los DS.