Métodos Númericos Gradiente Descendiente y Newton-Raphson

A. Rigoberto
Universidad Centroamericana Jose simeon cañas
00042220@uca.edu.sv

E. Alexander
Universidad Centroamericana Jose simeon cañas
00066819@uca.edu.sv

O. Heriberto
Universidad Centroamericana Jose simeon cañas
00177919@uca.edu.sv

Z. Daniela
Universidad Centroamericana Jose simeon cañas
00062019@uca.edu.sv

Q. Javier
Universidad Centroamericana Jose simeon cañas
00062019@uca.edu.sv

***Resumen*—Este artículo investiga los métodos de optimización en aprendizaje automático, con enfoque en el Gradiente Descendente y Newton-Raphson. Analizamos teóricamente las condiciones de convergencia, estabilidad y tasas de error para ambos algoritmos, demostrando que el Gradiente Descendente alcanza una convergencia lineal $\mathcal{O}(1/\epsilon)$ para funciones convexas con gradiente Lipschitz continuo, mientras que para funciones fuertemente convexas la convergencia mejora a $\mathcal{O}(c^k)$. Presentamos las condiciones de estabilidad ($\eta < 2/L$) y caracterizamos los regímenes inestables mediante análisis del índice de progreso relativo. Comparativamente, el método de Newton-Raphson muestra ventajas en velocidad de convergencia pero requiere condiciones de diferenciabilidad más estrictas. Los resultados teóricos se complementan con análisis de sensibilidad a hiperparámetros, escalabilidad y aplicaciones prácticas en problemas de alta dimensionalidad. Concluimos con recomendaciones para la selección de algoritmos según las propiedades de la función objetivo y requerimientos computacionales.**

***Index Terms*—optimización convexa, gradiente descendente, Newton-Raphson, tasa de convergencia, aprendizaje automático, condiciones Lipschitz**

I. Introducción a Machine learning

En El machine learning es la ciencia de desarrollo de algoritmos y modelos estadísticos que utilizan los sistemas de computación con el fin de llevar a cabo tareas sin instrucciones explícitas, en vez de basarse en patrones e inferencias. Los sistemas de computación utilizan algoritmos de machine learning para procesar grandes cantidades de datos históricos e identificar patrones de datos. Esto les permite generar resultados con mayor precisión a partir de un conjunto de datos de entrada. Por ejemplo, los científicos de datos pueden entrenar una aplicación médica para diagnosticar el cáncer con imágenes de rayos X a partir del almacenamiento de millones de imágenes escaneadas y diagnósticos correspondientes.

I-A. ¿Como funciona?

En La idea central del machine learning es la existencia de una relación matemática entre cualquier combinación de datos de entrada y salida. El modelo de machine learning no conoce de antemano esta relación, pero puede adivinarla si se le dan suficientes conjuntos de datos. Esto significa que cada algoritmo de machine learning se crea en torno a una función matemática modificable. El principio subyacente puede entenderse así:

1. Entrenamos el algoritmo al darle las siguientes combinaciones de entrada y salida (e,s): (2,10), (5,19) y (9,31)
2. El algoritmo calcula que la relación entre la entrada y la salida es: o=3*i+4
3. A continuación, le damos la entrada 7 y le pedimos que prediga la salida. Puede determinar automáticamente la salida como 25.

En Si bien se trata de conocimientos básicos, el machine learning se centra en el principio de que los sistemas de computación pueden relacionar matemáticamente todos los puntos de datos complejos, siempre y cuando tengan suficientes datos y potencia de computación para procesarlos. Por lo tanto, la precisión de la salida está relacionada directamente con la magnitud de la entrada dada.

II. Gradiente Descendente

II-A. Descripcion y/o deduccion del metodo

En El Gradiente Descendente es un algoritmo de optimización fundamental en el aprendizaje automático, utilizado para minimizar la función de pérdida o error de un modelo. Funciona calculando el gradiente (la derivada) de la función de pérdida con respecto a los parámetros del modelo, como los pesos en una red neuronal. Este gradiente indica la dirección en la que la función de pérdida aumenta más rápidamente.

En el Gradiente Descendente, se ajustan los parámetros del modelo en la dirección opuesta al gradiente, es decir, "descendiendo" hacia el valor mínimo de la función de pérdida. La magnitud del ajuste se controla por la tasa de

aprendizaje. Este proceso se repite iterativamente, actualizando los parámetros paso a paso, hasta que se alcanza un mínimo local o global en la función de pérdida, lo que idealmente corresponde a un modelo bien entrenado y optimizado.

EnEn otras palabras La idea de este tipo de métodos es generar una secuencia de valores xk, de modo que, la función objetivo disminuya en cada avance de una iteración $f(xk+1) < f(xk)$, Permite espacios más flexibles hasta alcanzar eventualmente un mínimo. Las propiedades de convexidad de la función objetivo f, y el dominio del problema $\mathbb{R}^n$, aseguran las condiciones necesarias para la convergencia de la sucesión de valores $\{x_k\}$ generados por el algoritmo de gradiente descendente.

En general, los algoritmos de descenso escogen una dirección dk: Rn que satisface la regla $h\nabla f(x), dki < 0$. Esta regla se puede interpretar geométricamente, como la condición donde el vector dk forma un ángulo de menos de 90 grados con la dirección de máximo descenso. En el caso del método de gradiente, la dirección corresponde al vector de gradiente negativo

$dk = -\nabla f(xk)$. Por consiguiente, el vector de gradiente proporciona la dirección de máximo descenso o mayor tasa de descenso en la zona de evaluación local. No obstante, dependiendo del tipo de problema y función, existen otras opciones para el vector de dirección, por ejemplo, subgradientes (cuando la función es no diferenciable) o métodos que utilizan información de segundo orden como el método de Newton que hace uso del Hessiano $(\nabla 2f(xk)) - 1$ y en cada iteración minimiza una aproximación cuadrática de la función

El método de gradiente de máximo descenso genera una secuencia de puntos xk: Rn mediante la siguiente expresión de recurrencia

$xk + 1 = xk - tk\nabla f(xk), k = 0, 1, 2, ...$

II-B. Demostracion de orden convergencia

La tasa de convergencia proporciona los márgenes teóricos de complejidad donde los algoritmos se desempeñan. Estos bordes se modifican de acuerdo a la estructura del problema donde son aplicados. El peor desempeño (límite superior) queda delimitado por la condición Lipschitz del gradiente, mientras que el mejor desempeño se obtiene, suponiendo, el algoritmo evalúa una función fuertemente convexa con parámetro > 0. Ambos limites, pueden ser determinados mediante una aproximación cuadrática de la función objetivo f, empleando las constantes L y μ como parámetros del factor cuadrático (como se observa en las desigualdades (2.9), (2.10) de abajo). El desempeño de la tasa de convergencia para una función convexa, entonces, se caracteriza por situarse entre ambas condiciones de aproximación cuadrática,

$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}|y + x|^{\frac{2}{2}}$ (2.9)

$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}|y + x|^{\frac{2}{2}}$ (2.10)

De manera equivalente usando el Hessiano para todo x en $\mathbb{R}^n$

$$\mu I \preccurlyeq \nabla^2 f(x) \preccurlyeq LI \qquad ((2.11))$$

En este sentido, para una función f convexa, suave con gradiente Lipschitz continuo de constante L, se demuestra que la tasa de convergencia del método de gradiente de máximo descenso es lineal o logarítmica. La demostración se inicia con la desigualdad (2.7), que asegura decrecimiento de la función en cada iteración

$$f(y) \leq f(x) - \left(1 - \frac{1}{2}tL\right) t\|\nabla f(x)\|^2$$

$$= f(x) - \frac{(2 - tL)t}{2}\|\nabla f(x)\|^2$$

Usando α como variable auxiliar para el término $(2 - tL)t$ y reemplazando se obtiene:

$$f(y) \leq f(x) - \frac{\alpha}{2}\|\nabla f(x)\|^2 \qquad ((2.12))$$

Interesa incluir $f(x_*)$ en esta desigualdad. Para esto, se puede acotar el valor de $f(x)$ en torno al valor óptimo de la función $f(x_*)$. Usando la propiedad de convexidad de la función, se tiene la siguiente aproximación lineal en torno al valor óptimo:

$$f(x_*) \geq f(x) + \nabla f(x)^T(x_* - x)$$

$$f(x) \leq f(x_*) + \nabla f(x)^T(x - x_*)$$

Reemplazando en la desigualdad anterior (2.12):

$$f(y) \leq f(x_*) + \nabla f(x)^T(x - x_*) - \frac{\alpha}{2}\|\nabla f(x)\|^2$$

$$f(y) - f(x_*) \leq \frac{1}{2\alpha}\left(2\alpha\nabla f(x)^T(x - x_*) - \alpha^2\|\nabla f(x)\|^2\right)$$

$$= \frac{1}{2\alpha}\left(2\alpha\nabla f(x)^T(x - x_*) - \alpha^2\|\nabla f(x)\|^2 + \|x - x_*\|^2 - \|x - x_*\|^2\right)$$

$$= \frac{1}{2\alpha}\left(\|x - x_*\|^2 - \|x - \alpha\nabla f(x) - x_*\|^2\right)$$

Se puede observar que $y = x - \alpha\nabla f(x)$ es la definición de un paso de gradiente:

$$f(y) - f(x_*) \leq \frac{1}{2\alpha}\left(\|x - x_*\|^2 - \|y - x_*\|^2\right)$$

Esta última desigualdad ocurre para cada paso de iteración. Sumando todos los pasos:

$$\sum_{i=1}^{k}(f(x_i)-f(x_*)) \le \sum_{i=1}^{k}\frac{1}{2\alpha}\left(\|x_{i-1}-x_*\|^2-\|x_i-x_*\|^2\right)$$
$$\le \frac{1}{2\alpha}\left(\|x_0-x_*\|^2-\|x_k-x_*\|^2\right)$$

La sumatoria del lado derecho se cancela haciendo uso de la suma telescópica. Eliminando el término positivo $\|x_k - x_*\|^2$ permite dejar la diferencia $\|x_0 - x_*\|^2$ como límite superior de la desigualdad:

$$\sum_{i=1}^{k}(f(x_i)-f(x_*)) \le \frac{1}{2\alpha}\|x_0-x_*\|^2$$

Suponiendo que la evaluación de la función $f(x_k)$ disminuye en cada iteración:

$$f(x_k)-f(x_*) \le \sum_{i=1}^{k}(f(x_i)-f(x_*)) \le \frac{1}{2\alpha}\|x_0-x_*\|^2$$

Obtenemos la tasa de convergencia del método de gradiente:

$$f(x_k)-f(x_*) \le \frac{\|x_0-x_*\|^2}{2t(2-tL)k} \quad \square \tag{2.13}$$

Se verifica entonces que para una función convexa, bajo la condición de Lipschitz continuo el método de gradiente presenta una tasa de convergencia lineal del $\mathcal{O}(\frac{1}{\epsilon})$, donde k es el número de iteraciones. Usando un valor de paso $t \le \frac{1}{L}$ se obtiene una estimación:

$$f(x_k)-f(x_*) \le \frac{L}{2k}\|x_0-x_*\|^2, \quad \forall x_k \in \mathbb{R}^n. \tag{2.14}$$

Si se quiere asegurar que $f(x_k) - f(x_*) < \epsilon$ se requiere un número de iteraciones k sea del orden de $\mathcal{O}(\frac{1}{\epsilon})$.

Una función fuertemente convexa está acotada inferiormente por una constante $\mu > 0$. Entonces, para el caso de una función fuertemente convexa, gradiente Lipschitz continua, se verifica que la tasa de convergencia es del orden $\mathcal{O}(c^k)$ con $0 < c < 1$ [**?**], [**?**]. Con un tamaño de paso $t = \frac{2}{\mu+L}$, se evidencia que $c = \frac{\kappa-1}{\kappa+1}$ es mínimo, donde $\kappa = \frac{L}{\mu}$ se denomina número de condición. El método de gradiente de descenso presenta la siguiente tasa de convergencia:

$$f(x_k)-f(x_*) \le \frac{L}{2}\left(\frac{\kappa-1}{\kappa+1}\right)^k\|x_0-x_*\|^2 \tag{2.15}$$
$$\|x_k-x_*\| \le \left(\frac{\kappa-1}{\kappa+1}\right)^k\|x_0-x_*\|$$

El número de iteraciones para $f(x_k) - f(x_*) \le \varepsilon$ es del orden $\mathcal{O}\left(\log\left(\frac{1}{\varepsilon}\right)\right)$:

$$\frac{\log\left(\frac{f(x_0)-f(x_*)}{\varepsilon}\right)}{\log\left(\frac{2}{L}\left(\frac{\kappa+1}{\kappa-1}\right)\right)}$$

El número de condición κ presenta influencia en la tasa de convergencia. Un valor de $\kappa > 1$ muestra que $f(x_k)$ converge a $f(x_*)$ cuando $k \to \infty$, pero un valor muy grande de este número incrementa enormemente el número de iteraciones, lo que podría impedir alcanzar la convergencia en la práctica.

II-C. Condiciones de estabilidad

Para que el gradiente descendente funcione de manera estable, se debe cumplir la condición de que la tasa de aprendizaje (η) sea menor que $\frac{2}{L}$, donde L es la suavidad de la función de costo. Esta condición asegura que el costo disminuya en cada iteración, lo que se conoce como el "régimen estable". Sin embargo, en la práctica, muchas veces se utiliza una tasa de aprendizaje mayor, lo que puede llevar a un comportamiento de convergencia inestable.

Comportamiento en el Régimen Inestable
En el régimen inestable, el algoritmo puede no converger de manera directa a un mínimo, sino que puede oscilar alrededor de la solución. Esto ocurre porque el gradiente descendente no puede permanecer consistentemente por encima o por debajo de cero debido a la naturaleza oscilante de los iterados del GD. Esto se formaliza a través del "índice de progreso relativo" (RP), que en este contexto puede oscilar cerca de cero.

Causas del Régimen Inestable
Las causas de la inestabilidad en el gradiente descendente pueden incluir:

- Falta de Puntos Estacionarios: La ausencia de puntos estacionarios no triviales cerca de la trayectoria del GD puede provocar que el algoritmo entre en un régimen inestable.
- Comportamiento del Gradiente: Aunque el gradiente puede no ser Lipschitz, se ha observado que alguna forma de Lipschitz para la Hessiana es válida a lo largo de la trayectoria de GD.
- Condiciones de Suavidad: En configuraciones de costos no convexos, no está claro si la condición $\frac{2}{L}$ es necesaria o razonable.

Implicaciones de la Inestabilidad
A pesar de que en el régimen inestable se espera que el GD diverja, se ha demostrado que, en el contexto del entrenamiento de redes neuronales, el algoritmo puede seguir convergiendo de manera no monótona. Esto sugiere que el comportamiento de convergencia puede ser más complejo de

lo que tradicionalmente se ha asumido.

Experimentación y Observaciones
Experimentos han demostrado que, incluso con tasas de aprendizaje que no cumplen con la condición de estabilidad, el GD puede seguir disminuyendo el costo a largo plazo, aunque de manera oscilante. Esto ha llevado a una reevaluación de las condiciones bajo las cuales se entiende la convergencia en el aprendizaje profundo.

II-D. Comparacion de los metodos

La primera diferencia radica en el hecho de que el **descenso de gradiente** es paramétrico según la tasa de aprendizaje α. El **método de Newton** no es paramétrico, lo que significa que podemos aplicarlo sin preocuparnos por la optimización de hiperparámetros. Existe, en verdad, una versión paramétrica del método de Newton, pero solo se aplica en casos donde trabajamos con funciones polinómicas con múltiples raíces.

La segunda diferencia tiene que ver con la función de costo sobre la cual aplicamos el algoritmo. El método de Newton tiene restricciones más fuertes en términos de diferenciabilidad de la función que el descenso de gradiente. Si la segunda derivada de la función no está definida en la raíz de la función, entonces podemos aplicar descenso de gradiente pero no el método de Newton.

La tercera diferencia consiste en el comportamiento alrededor de puntos estacionarios. Si el descenso de gradiente encuentra un punto estacionario durante la iteración, el programa continúa ejecutándose, aunque los parámetros no se actualicen. Sin embargo, el método de Newton requiere calcular $\frac{f'(x)}{f''(x)}$ cuando $f'(x) = f''(x) = 0$. El programa que lo ejecuta terminaría entonces con un error de división por cero.

II-E. Ventajas y/o desventajas

- **Ventajas del Gradiente Descendente**
  - Simplicidad y facilidad de implementación: El gradiente descendente es un algoritmo sencillo de entender y programar, lo que lo hace ideal para quienes se inician en el aprendizaje automático.
  - Eficiencia computacional por iteración: Cada paso del algoritmo es computacionalmente ligero, ya que no requiere el cálculo de derivadas de segundo orden ni el almacenamiento de grandes matrices.
  - Escalabilidad: Las variantes del gradiente descendente, como el descenso estocástico y el mini-batch, permiten manejar grandes conjuntos de datos y problemas de alta dimensionalidad.
  - Flexibilidad: Se puede adaptar a diferentes problemas mediante ajustes en la tasa de aprendizaje y otras técnicas como el momentum o la regularización.
- **Desventajas del Gradiente Descendente**
  - Sensibilidad a la tasa de aprendizaje: Una tasa de aprendizaje demasiado alta puede hacer que el algoritmo diverja, mientras que una demasiado baja puede ralentizar la convergencia.
  - Posibilidad de quedar atrapado en mínimos locales: En funciones no convexas, el algoritmo puede converger a un mínimo local en lugar del mínimo global.
  - Convergencia lenta en ciertas regiones: Cuando la superficie de la función de pérdida es plana o tiene valles estrechos, el algoritmo puede tardar mucho en converger.
  - Dependencia de la escala de las características: Si las características del conjunto de datos no están correctamente escaladas, el algoritmo puede tener dificultades para converger.

II-F. Elementos ilustrativos

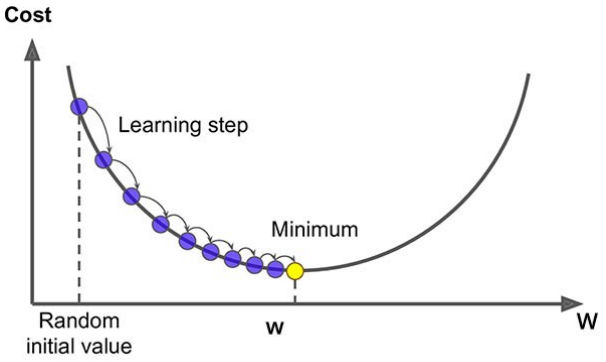

Figura 1. Visualización del Gradiente descendente, función de coste (Cost) y pesos (W).

II-G. Ejemplos

En la figura de abajo vemos una ilustración del funcionamiento del algoritmo. En primer lugar, el punto inicial "0" es seleccionado aleatoriamente:

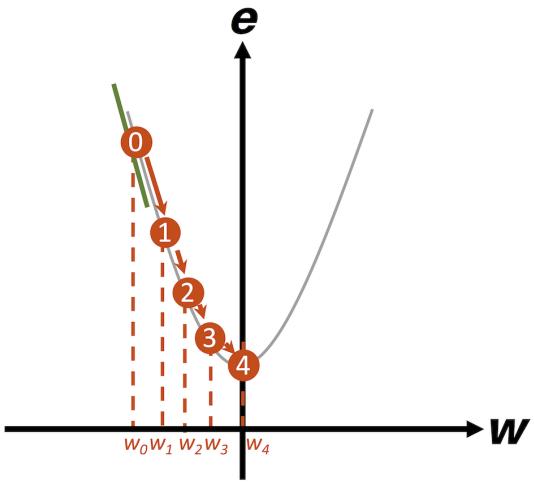

Figura 2. Descripción de la figura.

En la primera iteración se calcula el gradiente en el punto ω_0, que corresponde a la línea recta de color verde en la gráfica. Se observa que este gradiente es negativo, y por tanto la cantidad $-\alpha \cdot$gradiente es un número positivo, lo cual quiere decir que al valor inicial de ω_0 se sumará el término positivo $-\alpha \cdot$gradiente, y por tanto ω_1 estará ubicado a la derecha del punto inicial.

En la segunda iteración el gradiente seguirá siendo negativo, y por tanto el término $-\alpha \cdot$ gradiente será nuevamente positivo, lo cual quiere decir que el nuevo valor ω_2 estará ubicado a la derecha de ω_1.

Si se repite el procedimiento podemos observar que en cada iteración el nuevo valor de ω calculado está cada vez más cerca del valor mínimo buscado ($\omega = 0$).

Algo similar ocurre cuando el punto inicial (ω) es positivo, como se muestra en la figura de abajo:

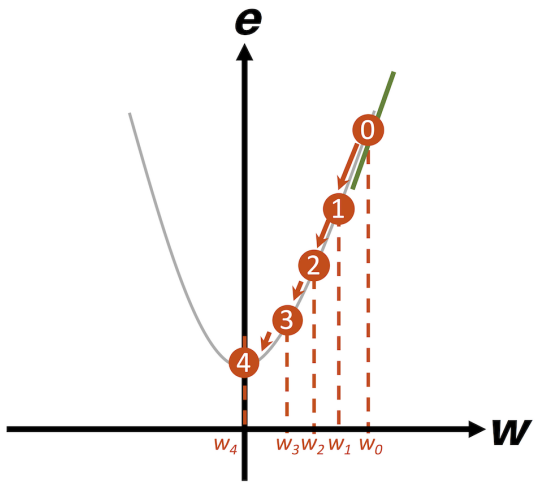

Figura 3. Ejemplo gráfico del gradiente descendente cuando el punto inicial (w) es positivo.

El gradiente o derivada de la función $e = \omega^2 + 1$ es igual a 2ω. Para simular el comportamiento del algoritmo, supongamos inicialmente un valor $\omega_0 = 5$ y una tasa de aprendizaje $\alpha = 0{,}25$.

Así, en la primera iteración del algoritmo tendremos:

- $\omega_0 = 5$
- Gradiente $\nabla e(\omega_0) = 2 \times 5 = 10$
- $\omega_1 = \omega_0 - \alpha \cdot \nabla e(\omega_0) = 5 - 0{,}25 \times 10 = 2{,}5$

Al repetir el procedimiento obtenemos los siguientes resultados en las iteraciones 2 a 6:

- $\omega_2 = 1{,}25$
- $\omega_3 = 0{,}625$
- $\omega_4 = 0{,}3125$
- $\omega_5 = 0{,}156$
- $\omega_6 = 0{,}078$

Podemos observar que en este caso se requieren aproximadamente 6 iteraciones para que el valor de ω sea casi cero.

Si ahora usamos una tasa de aprendizaje $\alpha = 0{,}4$, y repetimos el mismo procedimiento anterior, veremos que en lugar de 6 se requieren tan solo 3 iteraciones para que ω alcance el valor de 0.04:

- $\omega_1 = 1{,}0$
- $\omega_2 = 0{,}2$
- $\omega_3 = 0{,}04$

Si consideramos la situación extrema $\alpha = 1{,}0$, observamos el siguiente comportamiento oscilatorio:

- $\omega_1 = -5$
- $\omega_2 = 5$

- $\omega_3 = -5$
- $\omega_4 = 5$
- $\omega_5 = -5$
- $\omega_6 = 5$

Así, el valor de w alterna entre 5.0 y -5.0 y nunca disminuye. En este caso decimos que el algoritmo no converge y por tanto nunca encontrará un valor mínimo.

Los anteriores ejemplos nos permiten concluir que la tasa de aprendizaje α determina la cantidad de iteraciones requeridas para que el algoritmo del gradiente descendente alcance el valor mínimo, y el valor seleccionado dependerá de la función a minimizar (sin embargo siempre debe ser menor a 1).

III. Newton-Raphson

Describa los métodos utilizados en su investigación. Incluya ecuaciones cuando sea necesario:

III-A. Descripcion y/o deduccion del metodo

Sea $f : [a, b] \to \Re, f \in C^2[a, b]$ y x_k una estimación del cero en $f : [a, b]$

Deducción del método

Considere $\Delta y_k = f'(x_k) * \Delta y_k$ con Δy_k apropiado $y_k = f(x_k)$ y además $\Delta y_k = f(x_{k+1}) - f(x_k)$

Si x_k es una aproximación de la solución de $f(x) = 0$ en $[a, b]$ y x_{k+1} una mejor aproximación, $f(x_{k+1}) \approx 0$ entonces:

$$\Delta y_k = -f(x_k)y - f(x_k) = f'(x_k) * \Delta x_k$$

así se obtiene:

$$\Delta x_k = -[f'(x_k)]^{-1} * f(x_k)$$

con

$$\Delta x_k = x_{k+1} - x_k$$

III-B. Demostracion de orden convergencia

Analicemos cuantitativamente el orden de convergencia. Dado un error pequeño δ_n en la n-ésima iteración, determinaremos cuánto más pequeño es el error δ_{n+1} en la siguiente iteración.

En particular, definimos $x_n = x(1 + \delta_n)$, donde $x = \sqrt{a}$ es la solución exacta. Esto corresponde a definir $|\delta_n|$ como el **error relativo**:

$$|\delta_n| = \frac{|x_n - x|}{|x|},$$

también llamado **error fraccional**. El error relativo es típicamente la forma más útil de cuantificar el error porque es una cantidad *adimensional* (independiente de las unidades o escala de x). El logaritmo $(-\log_{10} \delta_n)$ del error relativo representa aproximadamente el número de **dígitos significativos exactos** en la respuesta x_n.

Sustituyendo esta definición de x_n (y x_{n+1}) en términos de δ_n (y δ_{n+1}) en nuestra fórmula de iteración de Newton, y usando que $a/x = x$, podemos dividir ambos lados por x:

$$1 + \delta_{n+1} = \frac{1}{2}\left(1 + \delta_n + \frac{1}{1 + \delta_n}\right) =$$

$$\frac{1}{2}\left[1 + \delta_n + 1 - \delta_n + \delta_n^2 + O(\delta_n^3)\right]$$

donde hemos expandido $(1 + \delta_n)^{-1}$ en series de Taylor. La notación $O(\delta_n^3)$ significa aproximadamente "términos de orden δ_n^3 o superiores". Como la sucesión converge, podemos asumir que $|\delta_n|^3 \ll 1$ para n suficientemente grande, por lo que los términos δ_n^3 y de orden superior son eventualmente despreciables frente a δ_n^2. Obtenemos:

$$\delta_{n+1} = \frac{\delta_n^2}{2} + O(\delta_n^3),$$

lo que significa que el **error aproximadamente se cuadra** (y se reduce a la mitad) en cada iteración una vez cerca de la solución. Elevar al cuadrado el error relativo corresponde precisamente a duplicar el número de dígitos significativos, lo que explica el fenómeno observado. Esto se conoce como **convergencia cuadrática**, por lo tanto el método de Newton Raphson tiene convergencia cuadrática.

III-C. Condiciones de estabilidad

Según Burden (Sección 2.4, pp. 68–71), el método de Newton-Raphson es estable bajo las siguientes condiciones:

1. Condiciones Básicas de Convergencia

1. **Regularidad de f**:

$$f \in C^2([a, b]), \quad \text{donde } [a, b] \text{ contiene a la raíz } p. \tag{1}$$

2. **No anulación de la derivada**:

$$f'(p) \neq 0. \tag{2}$$

3. **Inicialización adecuada**:

$$|p_0 - p| < \delta, \quad \text{para } \delta \text{ suficientemente pequeño.} \tag{3}$$

2. Teorema de Convergencia Cuadrática (Teorema 2.6)

Si se cumplen las condiciones anteriores, entonces:

- Existe una constante $M > 0$ tal que:

$$|p_{n+1} - p| \leq M|p_n - p|^2, \tag{4}$$

  lo que garantiza **convergencia cuadrática**.
- La cota del error depende de:

$$M = \frac{\max_{x\in[a,b]} |f''(x)|}{2 \min_{x\in[a,b]} |f'(x)|}. \tag{5}$$

3. Casos de Inestabilidad

- **Raíces múltiples** ($f'(p) = 0$):

$$\text{Convergencia se reduce a lineal.} \tag{6}$$

- **Derivadas cercanas a cero**:

$$\text{Si } |f'(x)| \approx 0 \text{ cerca de } p, \text{ el método puede diverger.} \tag{7}$$

- **Elección inadecuada de** p_0:

$$\text{Si } p_0 \text{ está lejos de } p, \text{ no hay garantía de convergencia.} \tag{8}$$

III-D. Comparacion de los metodos

Diferencias entre el método de Newton y el Gradiente Descendiente

Parametrización

- **Descenso de gradiente:** Es paramétrico según la tasa de aprendizaje α.
- **Método de Newton:** No es paramétrico, lo que significa que podemos aplicarlo sin preocuparnos por la optimización de hiperparámetros.
- **Nota:** Existe una versión paramétrica del método de Newton, pero solo se aplica cuando operamos con una función polinómica con múltiples raíces.

Diferenciabilidad de la función

- **Método de Newton:** Presenta restricciones más estrictas en cuanto a la diferenciabilidad de la función.
- **Caso particular:** Si la segunda derivada $f''(x)$ no está definida en la raíz:
  - Se puede aplicar descenso de gradiente
  - No se puede aplicar el método de Newton

Comportamiento en puntos estacionarios

- **Descenso de gradiente:**
  - Si encuentra un punto estacionario durante la iteración, el programa continúa ejecutándose
  - Los parámetros no se actualizan en este caso
- **Método de Newton:**
  - Requiere calcular $\frac{f'(x)}{f''(x)}$
  - Para $f'(x) = f''(x) = 0$ se produce un error de división por cero
  - El programa finalizaría con error en este caso

III-E. Ventajas y/o desventajas

Ventajas del Método de Newton-Raphson

El método de Newton-Raphson ofrece varias ventajas distintivas sobre otros métodos de búsqueda de raíces, especialmente en términos de velocidad, eficiencia y aplicabilidad a funciones suaves. A continuación, se presentan algunas ventajas clave:

A. Convergencia rápida (convergencia cuadrática)

- El método de Newton-Raphson converge cuadráticamente cerca de la raíz, lo que significa que la cantidad de dígitos correctos aproximadamente se duplica con cada iteración, siempre que la estimación inicial esté cerca de la raíz verdadera.

B. Eficiente para funciones de buen comportamiento

- Cuando la función $f(x)$ es suave y su derivada $f'(x)$ es fácilmente calculable, el método de Newton-Raphson proporciona una búsqueda de raíces muy eficiente y rápida.

C. No es necesario el uso de corchetes

- A diferencia de métodos como la bisección y la falsa posición, el método de Newton-Raphson no requiere un intervalo de horquillado inicial. Solo se necesita una única aproximación inicial, lo cual puede resultar mucho más práctico.

D. Uso directo de información derivada

- El método de Newton-Raphson utiliza directamente la derivada $f'(x)$, lo que significa que puede incorporar más información sobre el comportamiento local de la función para converger más rápidamente.

E. Ampliamente aplicable a funciones suaves

- Cuando la función $f(x)$ es diferenciable, el método de Newton-Raphson es muy versátil y se puede aplicar a una amplia variedad de ecuaciones (incluidas ecuaciones trascendentales, algebraicas y trigonométricas).

Desventajas del Método de Newton-Raphson

A. Dependencia de la estimación inicial

- El método es muy sensible a la estimación inicial. Una mala elección de x_0 puede provocar:
  - Divergencia del método
  - Convergencia lenta
  - Convergencia a una raíz local distinta a la deseada
- Por lo tanto, el método puede requerir:
  - Cuidadosa consideración de la estimación inicial
  - Múltiples intentos con diferentes puntos de partida

B. Cálculo de derivadas

- El método requiere el cálculo de la primera derivada $f'(x)$ de la función
- Problemas asociados:
  - Para algunas funciones, el cálculo puede ser computacionalmente costoso
  - En algunos casos la derivada puede ser difícil de obtener analíticamente
- Alternativas:

- Métodos sin derivada (como el método de la secante) pueden ser más adecuados
- Uso de derivadas numéricas (aunque introduce error adicional)

C. Falla en puntos críticos

- Casos problemáticos:
  - Cuando $f'(x) = 0$ en la raíz o cerca de ella (división por cero)
  - Cuando la función tiene singularidades cerca de la raíz
  - Cuando existen puntos de inflexión cerca de la raíz
- Consecuencias:
  - El método puede fallar completamente
  - Puede producir convergencia errática o no converger

D. Raíces múltiples

- Problema fundamental:
  - En funciones con múltiples raíces, el método puede converger a diferentes raíces dependiendo de la estimación inicial
- Soluciones posibles:
  - Técnicas de deflación para eliminar raíces encontradas
  - Uso combinado con algoritmos de búsqueda global
  - Aplicación del método desde múltiples puntos iniciales
- Limitación adicional:
  - Convergencia más lenta en raíces múltiples (orden de convergencia se reduce)

III-F. Elementos ilustrativos

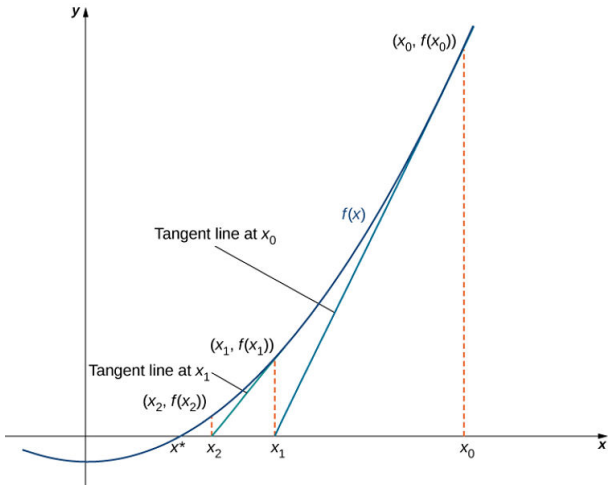

Figura 4. Visualización del método de newton

III-G. Ejemplos

$$f(x) = x^x e^{-x}\sqrt{2\pi x} - 120$$

$$\Rightarrow f'(x) = (f(x) + 120)\left[\ln x + \frac{1}{2x}\right]$$

$$\Rightarrow g(x) = x - \frac{f(x)}{(f(x) + 120)\left[\ln x + \frac{1}{2x}\right]}$$

El primer término de la función f se denomina fórmula de Stirling, y es una aproximación a la función factorial. La solución de esta ecuación es aproximadamente 5 (cuyo factorial es precisamente 120).

En este ejemplo tomamos $x_0 = 6$ y $TOL = 10^{-6}$. En la tabla puede verse la evolución de la solución, y que converge en 6 iteraciones.

En este caso, el cero es simple y se observa la convergencia cuadrática esperada por la teoría.

t	x_i	$\lvert x_i - x_{i-1}\rvert$
0	6.0000000000	–
1	5.5568195217	0.0797543409
2	5.2144604956	0.0656556949
3	5.0438222496	0.0338311379
4	5.0107677553	0.0065966925
5	5.0097329283	0.0002065633
6	5.0097319546	0.0000001944

[Fuente: Mantilla, Ignacio, "Análisis Numérico", Ejemplo 2.53]

IV. Conclusiones

Los métodos de Gradiente Descendiente y Newton-Raphson son herramientas fundamentales en optimización numérica, cada una con sus ventajas y limitaciones. Mientras que el Gradiente Descendiente destaca por su simplicidad y eficiencia en problemas de alta dimensión, como los encontrados en aprendizaje automático, el método de Newton-Raphson ofrece una convergencia más rápida cerca del óptimo gracias al uso de información de segundo orden, aunque a un mayor costo computacional. La elección entre ambos depende del equilibrio entre precisión, velocidad y recursos disponibles, siendo clave considerar la naturaleza del problema, la dimensionalidad de los datos y la disponibilidad de derivadas. En muchos casos, técnicas híbridas o aproximaciones como los métodos quasi-Newton pueden ofrecer un balance óptimo entre rendimiento y eficiencia.

Referencias

[1] "*¿Qué es el machine learning?* - Explicación sobre el machine learning empresarial - AWS". Amazon Web Services, Inc. Accedido el 16 de abril de 2025. [En línea]. Disponible: Link

[2] "¿Qué es el Gradiente Descendente?" CodificandoBits.com. Accedido el 16 de abril de 2025. [En línea]. Disponible: Link

[3] IBM. "*¿Qué es el descenso del gradiente?* | IBM". IBM - United States. Accedido el 16 de abril de 2025. [En línea]. Disponible: Link

[4] J. A. SEPÚLVEDA ORTEGA, "*IMPLEMENTACIÓN Y EXPERIMENTACIÓN CON MÉTODOS NUMÉRICOS DE OPTIMIZACIÓN DE PRIMER ORDEN EN SISTEMAS INERCIALES CON FACTOR DE AMORTIGUAMIENTO HESSIANO*", TESIS PARA OPTAR AL GRADO MAGISTER EN CIENC. ING., MENCION MAT. APL., 2021. Accedido el 16 de abril de 2025. [En línea].
Link

[5] BRAVO BOLÍVAR, J. E., BOTERO ARANGO, A. J., and BOTERO ARBELÁEZ, M. (2005). "*EL METODO DE NEWTON-RAPHSON - LA ALTERNATIVA DEL INGENIERO PARA RESOLVER SISTEMAS DE ECUACIONES NO LINEALES*". Scientia Et Technica, XI(27), 221-224.[En linea]
Link

[6] G. De Luca y M. Aibin. "gradient-descent-vs-newtons-gradient-descent". www.baeldung.com. Accedido el 18 de abril de 2025. [En línea]. Disponible: link

[7] V. Patel, S. Zhang y B. Tian. "Global Convergence and Stability of Stochastic Gradient Descent". List of Proceedings. Accedido el 18 de abril de 2025. [En línea]. Disponible: link

[8] K. Ahn, J. Zhang y S. Sra. "Understanding the Unstable Convergence of Gradient Descent". proceedings.mlr.press. Accedido el 18 de abril de 2025. [En línea]. Disponible: link

[9] Johnson, S. G. (s.f.). Newton's Method and the Sqrt Example [Notas de curso]. Departamento de Matemáticas, MIT. Link

[10] Burden, R. L., & Faires, J. D. (2011). *Numerical Analysis* (9th ed.). Cengage Learning.

[11] Ijraset. (s. f.). The Newton-Raphson Method: A Detailed analysis. IJRASET. Link

[12] De Luca, G., & De Luca, G. (2024, March 18). Gradient Descent vs. Newton's Gradient Descent Baeldung on Computer Science. Baeldung on Computer Science. Link