A white background with blue and grey dots

Description automatically generated

Mục Lục

[I. Các biến thể của Gradient Descent 2](#_Toc147251334)

[1. Batch Gradient Descent (Batch GD): 2](#_Toc147251335)

[2. Stochastic Gradient Descent (SGD) 3](#_Toc147251336)

[3. Mini-batch Gradient Descent (Mini-batch GD) 4](#_Toc147251337)

[II. Chứng minh sự hội tụ 5](#_Toc147251338)

[1. Nhắc lại 5](#_Toc147251339)

[2. Các cách chứng minh 6](#_Toc147251340)

[3.Bài toán tìm learning rate tối ưu 7](#_Toc147251341)

[4. Các điều kiện để chứng minh hội tụ bằng đạo hàm 7](#_Toc147251342)

[5.Các bước chứng minh gradient descent hội tụ 8](#_Toc147251343)

[6. Tài liệu tham khảo 10](#_Toc147251344)

## Các biến thể của Gradient Descent

**Epoch** là một vòng lặp qua toàn bộ tập dữ liệu huấn luyện trong quá trình huấn luyện mô hình.

### 1. Batch Gradient Descent (Batch GD):

Batch Gradient Descent (Batch GD) là biến thể cơ bản nhất của thuật toán Gradient Descent.

***Cách hoạt động:***

- Trong Batch GD, tất cả các mẫu dữ liệu trong tập huấn luyện được sử dụng để tính gradient của hàm mất mát tại mỗi bước cập nhật tham số.

- Gradient là vector chứa đạo hàm riêng của hàm mất mát theo từng tham số của mô hình. Đạo hàm này được tính bằng cách sử dụng toàn bộ tập dữ liệu huấn luyện.

- Sau khi tính toán gradient, các tham số của mô hình được cập nhật một lần duy nhất sau mỗi epoch (lặp qua toàn bộ dữ liệu huấn luyện).

***Ưu điểm:***

- Hội tụ đến giá trị tối thiểu toàn cục nếu tỷ lệ học tập (learning rate) được đặt đúng. Vì Batch GD sử dụng toàn bộ tập dữ liệu, nó có khả năng tìm ra giải pháp tối ưu tốt nhất.

- Được tích hợp trong các mô hình huấn luyện trên tập dữ liệu nhỏ với hiệu suất cao.

***Nhược điểm:***

- Không hiệu quả trên tập dữ liệu lớn. Việc tính toán gradient cho toàn bộ tập dữ liệu là đắt kinh tế về thời gian và bộ nhớ.

- Batch GD yêu cầu lưu trữ toàn bộ tập dữ liệu trong bộ nhớ, điều này là không thực tế cho các tập dữ liệu lớn.

***Ứng dụng:***

- Batch GD thường được sử dụng trong các tình huống mà tập dữ liệu huấn luyện có kích thước nhỏ hoặc có thể xử lý được trong bộ nhớ của máy tính.

- Nó được ưa chuộng trong các bài toán có dữ liệu nhỏ như hồi quy tuyến tính và phân loại đơn giản.

- Batch GD cũng thường được sử dụng trong quá trình pre-training của các mô hình Deep Learning trước khi chuyển sang các biến thể khác như Mini-batch GD để cải thiện tốc độ huấn luyện.

Tóm lại, Batch Gradient Descent là biến thể cơ bản của Gradient Descent, được sử dụng khi tập dữ liệu huấn luyện có kích thước nhỏ hoặc có thể xử lý được trong bộ nhớ. Nó đảm bảo hội tụ đến giá trị tối thiểu toàn cục, nhưng không hiệu quả trên các tập dữ liệu lớn.

### 2. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) là một biến thể của thuật toán Gradient Descent. Điểm mạnh chính của SGD là khả năng cập nhật tham số mô hình nhanh chóng và khả năng thoát ra khỏi các điểm tối ưu cục bộ. Dưới đây là chi tiết về SGD:

***1. Cách hoạt động:***

- Tại mỗi epoch (lần lặp), chúng ta duyệt qua từng điểm dữ liệu một trong tập huấn luyện, thay vì tính đạo hàm của hàm mất mát dựa trên toàn bộ dữ liệu.

- Với mỗi điểm dữ liệu, chúng ta tính đạo hàm của hàm mất mát và cập nhật trọng số sau mỗi lần tính toán.

- Thuật toán lặp lại quá trình trên cho đến khi hoàn thành một số epoch cố định hoặc khi điều kiện dừng được thỏa mãn.

Điểm chính của SGD là tính toán đạo hàm dựa trên từng điểm dữ liệu, giúp giảm thiểu mức tiêu thụ tài nguyên tính toán. Điều này đặc biệt hữu ích khi có cơ sở dữ liệu lớn hoặc trong các bài toán online learning, trong đó dữ liệu mới liên tục được thêm vào và mô hình cần được cập nhật thường xuyên.

Thứ tự lựa chọn điểm dữ liệu:

Trong quá trình lặp, một điểm quan trọng là thứ tự lựa chọn các điểm dữ liệu. Sau mỗi epoch, chúng ta nên xáo trộn (shuffle) lại thứ tự các điểm dữ liệu để đảm bảo tính ngẫu nhiên. Nếu không xáo trộn, thuật toán có thể bị hiệu ứng của dữ liệu nếu dữ liệu đã được sắp xếp theo một cách cụ thể. Điều này giúp thoát khỏi các điểm local minimum và có thể dẫn đến tìm nghiệm tối ưu toàn cục

Hàm lấy ngẫu nhiên:

Trong SGD, công thức cập nhật tham số (trọng số) thông thường được viết như sau:

θ = θ - η ∇θ J(θ; xi; yi)

SGD cho phép áp dụng các kỹ thuật tăng tốc GD như Momentum, AdaGrad, và RMSprop để tối ưu hóa quá trình cập nhật tham số và cải thiện tính ổn định của thuật toán.

***2. Ưu điểm:***

- Cập nhật nhanh: Do việc tính toán gradient chỉ dựa trên một ví dụ, nên việc cập nhật tham số mô hình nhanh chóng.

- Đa dạng: SGD có khả năng thoát ra khỏi các điểm tối ưu cục bộ và tìm kiếm không gian tham số một cách hiệu quả hơn.

***3. Hạn chế:***

- Dao động: SGD thường dao động mạnh trong quá trình tối ưu hóa và có thể gây nhiễu cho quá trình hội tụ, khiến cho hàm mất mát không giảm một cách liên tục.

- Không đảm bảo hội tụ đến giá trị tối thiểu toàn cục: Do sự dao động và tính ngẫu nhiên, SGD không đảm bảo hội tụ đến giá trị tối thiểu toàn cục. Tuy nhiên, nó có thể hội tụ đến một điểm gần tối thiểu toàn cục.

**4. Ứng dụng**

- SGD thường được sử dụng trong các mô hình học máy và mạng nơ-ron sâu (deep learning) với dữ liệu lớn, nơi Batch GD không thể sử dụng do yêu cầu tài nguyên tính toán và bộ nhớ lớn.

- SGD thường kết hợp với các biến thể như Momentum, Adagrad, RMSprop và Adam để tối ưu hóa hiệu suất hội tụ và giảm thiểu dao động.

Trong các mô hình deep learning, SGD và các biến thể của nó được sử dụng rộng rãi để huấn luyện mạng nơ-ron, vì tính nhanh chóng và khả năng thoát ra khỏi các điểm tối ưu cục bộ giúp cải thiện quá trình hội tụ.

### 3. Mini-batch Gradient Descent (Mini-batch GD)

Mini-batch Gradient Descent (Mini-batch GD) là một biến thể phổ biến của thuật toán Gradient Descent, kết hợp lợi ích của cả Batch GD và SGD. Dưới đây là chi tiết về Mini-batch GD:

***1. Cách hoạt động:***

* Chia tập dữ liệu huấn luyện: Tập dữ liệu huấn luyện được chia thành các mini-batch có kích thước nhỏ hơn so với toàn bộ tập dữ liệu, nhưng lớn hơn so với một ví dụ đơn lẻ. Kích thước của mini-batch thường là một tham số được xác định trước .
* Chọn một mini-batch ngẫu nhiên: Trong mỗi epoch (lần lặp), một mini-batch ngẫu nhiên được chọn từ tập dữ liệu. Điều này đảm bảo tính ngẫu nhiên trong việc tính toán gradient, giúp tránh các điểm tối ưu cục bộ và cải thiện tính đa dạng của quá trình tối ưu hóa.
* Tính toán gradient Gradient của hàm mất mát (ví dụ: hàm mất mát bình phương trung bình cho mạng nơ-ron) được tính dựa trên các mẫu trong mini-batch. Cụ thể, gradient là trung bình của gradient của từng mẫu trong mini-batch.
* Cập nhật tham số:Tham số của mô hình được cập nhật bằng cách trừ gradient đã tính toán từ giá trị hiện tại của tham số.

θ=θ−η∇θJ(θ;xi:i+n;yi:i+n)

Note: giá trị n thường được chọn là khoảng từ 50 đến 100.

***2. Ưu điểm:***

- Hiệu suất cao: Mini-batch GD thường nhanh hơn Batch GD vì việc tính toán gradient được thực hiện trên một lượng nhỏ dữ liệu.

- Ổn định: So với SGD, Mini-batch GD thường ổn định hơn trong quá trình hội tụ và giảm thiểu dao động.

- Linh hoạt: Kích thước mini-batch có thể điều chỉnh, cho phép kiểm soát tốc độ học tập và sự ổn định của thuật toán.

***3. Hạn chế:***

- Cần điều chỉnh kích thước mini-batch: Lựa chọn kích thước mini-batch có thể ảnh hưởng đến hiệu suất của thuật toán. Kích thước này phụ thuộc vào bài toán cụ thể và phải được điều chỉnh thử nghiệm.

***4. Ứng dụng:***

- Mini-batch GD là biến thể phổ biến trong huấn luyện mô hình học máy và mạng nơ-ron sâu (deep learning).

- Trong deep learning, Mini-batch GD được sử dụng để cải thiện tốc độ huấn luyện mô hình và giảm thiểu tài nguyên tính toán và bộ nhớ cần thiết so với Batch GD.

- Nó thường kết hợp với các biến thể như Momentum, Adagrad, RMSprop, và Adam để cải thiện hiệu suất và tốc độ hội tụ.

Mini-batch GD là sự lựa chọn phổ biến cho nhiều bài toán trong machine learning và deep learning do tính linh hoạt và hiệu suất của nó. Kích thước mini-batch được điều chỉnh để đạt hiệu suất tốt nhất cho từng bài toán cụ thể.

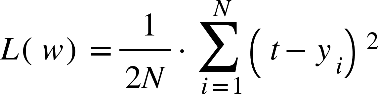
Mỗi biến thể của Gradient Descent có ưu điểm và hạn chế riêng, và lựa chọn phụ thuộc vào bài toán cụ thể và kích thước của tập dữ liệu. Batch GD đảm bảo hội tụ đến giá trị tối thiểu toàn cục nhưng có thể chậm trên dữ liệu lớn, trong khi SGD và Mini-batch GD có thể hội tụ nhanh hơn nhưng có thể dao động.

## Chứng minh sự hội tụ

### 1. Nhắc lại

Hàm mất mát( Loss Function):

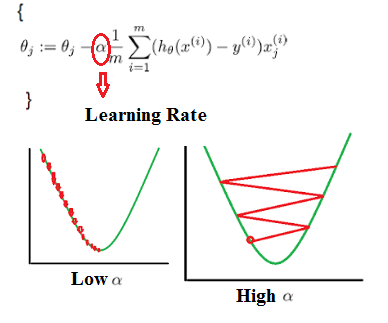
Linear Regression là một bài toán tối ưu hóa trong đó mục tiêu là tìm ra các hệ số của một mô hình tuyến tính sao cho hàm mục tiêu được tối thiểu hóa. Hàm mục tiêu trong Linear Regression thường được biểu diễn bằng Mean Squared Error (MSE), có thể viết như sau:



Gradient Descent Update Rule:

Trong mỗi vòng lặp, thuật toán cập nhật {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>w</mi><mrow><mi>t</mi><mo>+</mo><mn>1</mn></mrow></msub></mstyle></math>","origin":"MathType for Microsoft Add-in"} bằng cách di chuyển ngược theo hướng của đạo hàm (gradient) của hàm mất mát với một bước học {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3B7;</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"} (learning rate) .

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>w</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><msub><mi>w</mi><mi>k</mi></msub><mo>&#xA0;</mo><mo>&#x2212;</mo><mo>&#xA0;</mo><mi>&#x3B7;</mi><mo>&#x2207;</mo><mi>L</mi><mo>(</mo><msub><mi>w</mi><mi>k</mi></msub><mo>)</mo></mstyle></math>","origin":"MathType for Microsoft Add-in"}



### 2. Các cách chứng minh

Sự hội tụ của thuật toán Gradient Descent là một khía cạnh quan trọng trong việc tối ưu hóa hàm mục tiêu trong các bài toán tối ưu hóa. Chứng minh sự hội tụ của Gradient Descent có thể được thực hiện dựa trên các điều kiện và tiêu chuẩn khác nhau tùy thuộc vào biến thể cụ thể của thuật toán. Dưới đây là một số ví dụ về cách chứng minh sự hội tụ của Gradient Descent:

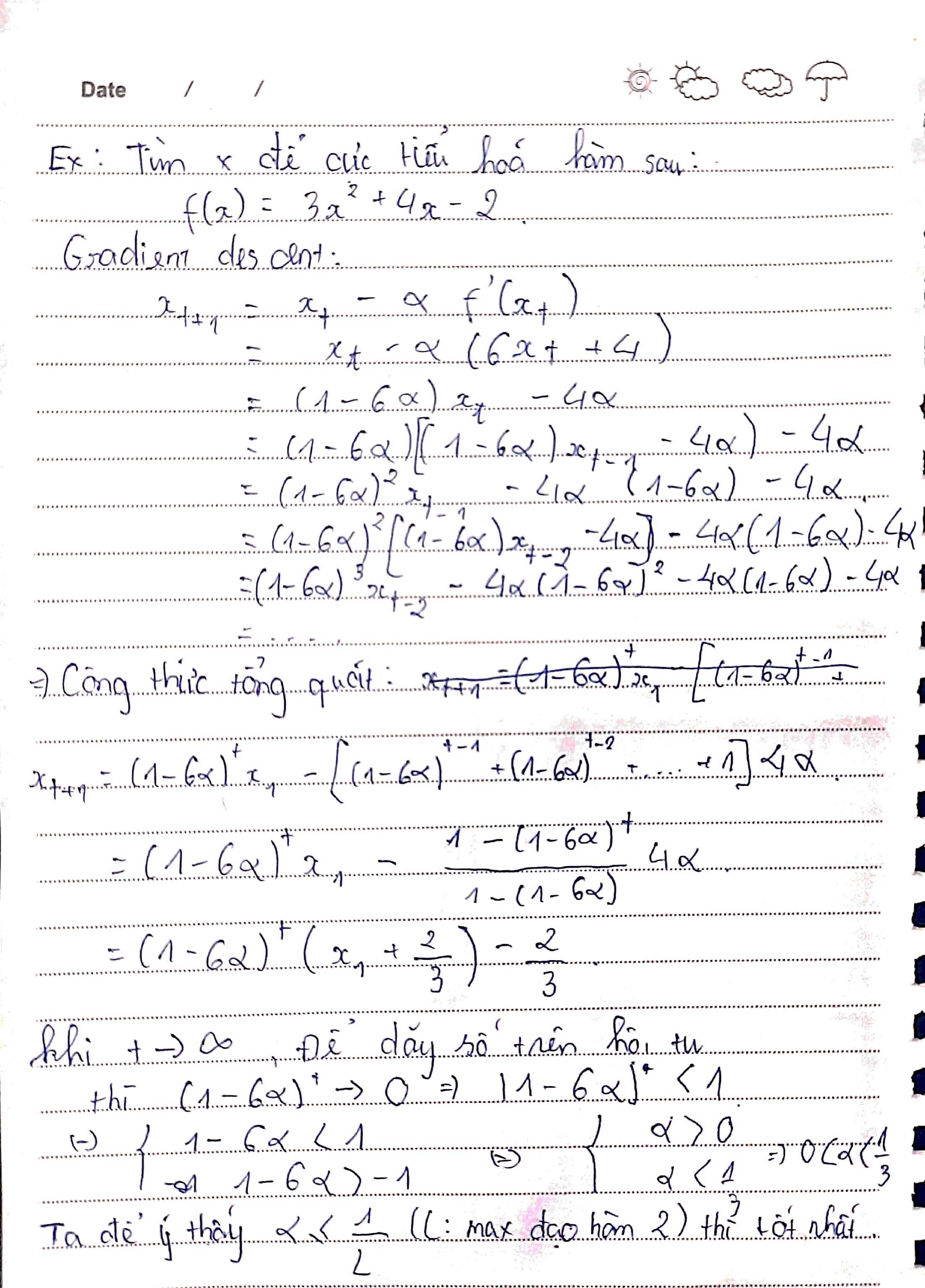
**1. Chứng minh sự hội tụ bằng đạo hàm**: Một trong những cách thông thường để chứng minh sự hội tụ của Gradient Descent là sử dụng tính toán đạo hàm của hàm mục tiêu. Nếu hàm mục tiêu là lồi (convex), và đạo hàm của nó liên tục và Lipchitz liên quan đến norm của gradient, thì có thể sử dụng điều kiện Lipschitz để chứng minh sự hội tụ.

**2. Chứng minh bằng dãy số**: Một cách tiếp cận phổ biến là sử dụng dãy số do Gradient Descent tạo ra. Chứng minh sự hội tụ bằng cách chứng minh rằng dãy số này là giảm dần và hội tụ đến điểm cực tiểu.

**3. Chứng minh sự hội tụ về điểm tối ưu gần đúng( Sử dụng hàm Taylor)**: Trong nhiều trường hợp thực tế, Gradient Descent không hội tụ chính xác đến điểm tối ưu, mà chỉ hội tụ đến một điểm gần đúng của nó. Chứng minh sự hội tụ trong trường hợp này thường liên quan đến xác định khoảng cách giữa điểm gần đúng và điểm tối ưu thực sự.

Chứng minh sự hội tụ của Gradient Descent là một lĩnh vực phức tạp và tùy thuộc vào bài toán cụ thể và biến thể của thuật toán. Việc sử dụng các nguyên tắc toán học và kiến thức về tối ưu hóa là quan trọng để chứng minh sự hội tụ một cách chính xác.

### 3.Bài toán tìm learning rate tối ưu



### 4. Các điều kiện để chứng minh hội tụ bằng đạo hàm

Chúng ta có thể chứng minh sự hội tụ của thuật toán Gradient Descent (GD) dựa trên điều kiện "**Lipschitz continuity**" của gradient (điều kiện này tương đương với "smoothness") và tính chất "**strong convexity**" của hàm mục tiêu. Điều này cho thấy GD là một phương pháp hiệu quả để tối ưu hóa hàm mục tiêu.

**4.1 Lipschitz continuity**

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mo>&#x2225;</mo><mo>&#x2207;</mo><mi>f</mi><mo>(</mo><mi>x</mi><mo>)</mo><mo>-</mo><mo>&#x2207;</mo><mi>f</mi><mo>(</mo><mi>y</mi><mo>)</mo><msub><mo>&#x2225;</mo><mn>2</mn></msub><mo>&#x2264;</mo><mi>L</mi><mo>&#x2225;</mo><mi>x</mi><mo>-</mo><mi>y</mi><msub><mo>&#x2225;</mo><mn>2</mn></msub><mo>,</mo><mo>&#x2200;</mo><mi>x</mi><mo>,</mo><mi>y</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"}

- Điều kiện "Lipschitz gradient" đề cập đến mức độ biến đổi (độ biến thiên) của gradient của hàm f(x) khi chuyển từ một điểm sang một điểm khác trong không gian. Nó thể hiện rằng sự biến đổi của gradient được kiểm soát bởi một hằng số L. Nghĩa rằng cho mọi cặp điểm x và y thì khoảng cách giữa gradient tại x và gradient tại y không vượt quá L lần khoảng cách giữa x và y theo một norm bất kỳ (trong trường hợp này là norm 2).

**4.2 Smoothness**

- Điều kiện Lipschitz gradient cũng được gọi là điều kiện về tính "smoothness" của hàm f(x). Nó cho biết rằng gradient của hàm không biến đổi quá nhanh khi chuyển từ một điểm sang một điểm khác.

Về cơ bản, điều kiện Lipschitz gradient/smoothness cho phép chúng ta kiểm soát tốc độ hội tụ của Gradient Descent. Khi f(x) thỏa mãn điều kiện này, chúng ta có thể chứng minh rằng dãy các điểm {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>x</mi><mi>t</mi></msub></mstyle></math>","origin":"MathType for Microsoft Add-in"}tạo ra bởi thuật toán sẽ hội tụ tới giá trị tối thiểu của f với một tốc độ được kiểm soát bởi hằng số L.

Ví dụ: Nếu hàm f thỏa mãn điều kiện Lipschitz gradient với L=10, điều này có nghĩa rằng gradient của L không thay đổi nhanh hơn 10 lần khoảng cách giữa hai điểm bất kỳ. Điều này giúp đảm bảo rằng thuật toán Gradient Descent hội tụ với một tốc độ chậm hơn và ổn định hơn.

**4.3 Quadratic Approximation**:

Trong trường hợp hàm *f* là hàm lồi (convex), chúng ta có thể giới hạn hàm *f* từ dưới bằng một biểu diễn tuyến tính (linear approximation) tại điểm *x*. Ngoài ra, nếu hàm *f* có Lipschitz gradient (điều kiện L-smooth), chúng ta cũng có thể giới hạn nó từ trên bằng một biểu diễn bậc hai (quadratic approximation).

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mo>&#xA0;</mo><mi>f</mi><mo>(</mo><mi>y</mi><mo>)</mo><mo>&#xA0;</mo><mo>&#x2264;</mo><mo>&#xA0;</mo><mi>f</mi><mo>&#xA0;</mo><mo>(</mo><mi>x</mi><mo>)</mo><mo>&#xA0;</mo><mo>+</mo><mo>&#x3008;</mo><mi>y</mi><mo>&#xA0;</mo><mo>&#x2212;</mo><mo>&#xA0;</mo><mi>x</mi><mo>,</mo><mo>&#xA0;</mo><mo>&#x2207;</mo><mi>f</mi><mo>&#xA0;</mo><mo>(</mo><mi>x</mi><mo>)</mo><mo>&#x3009;</mo><mo>&#xA0;</mo><mo>+</mo><mfrac><mrow><mo>&#xA0;</mo><mi>L</mi><mo>&#xA0;</mo></mrow><mn>2</mn></mfrac><mo>&#xB7;</mo><mo>|</mo><mo>|</mo><mi>y</mi><mo>-</mo><mi>x</mi><mo>|</mo><msubsup><mo>|</mo><mn>2</mn><mn>2</mn></msubsup></mstyle></math>","origin":"MathType for Microsoft Add-in"}

Phương trình trên cung cấp một loại ước tính cho hàm *f*. Nó nói rằng giá trị của hàm *f* tại *y* không vượt quá giá trị của hàm f tại *x* cộng với một giá trị tương tự của inner product giữa y-x và gradient tại x, cộng thêm một giá trị bị chặn bởi L.

### 5.Các bước chứng minh gradient descent hội tụ

**Điều kiện**

**1. Hàm mục tiêu f(x)là một hàm lồi**: Điều này có nghĩa là f thỏa mãn bất đẳng thức Jensen, tức là đoạn thẳng giữa hai điểm bất kỳ trên đồ thị của f không cắt đồ thị đó.

Định nghĩa qua đạo hàm hai lần: Một hàm số f(x) được coi là lồi trên một tập con X nếu đạo hàm hai lần của nó f''(x) không âm trên tập X. Nghĩa là, với mọi x thuộc X, ta có f''(x) ≥ 0.

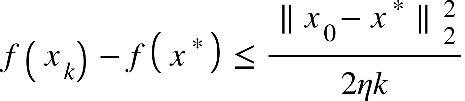
**2. Gradient của f liên tục (Lipschitz continuous):** Nghĩa là tồn tại một hằng số L sao cho:

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mo>&#x2225;</mo><mo>&#x2207;</mo><mi>f</mi><mo>(</mo><mi>x</mi><mo>)</mo><mo>-</mo><mo>&#x2207;</mo><mi>f</mi><mo>(</mo><mi>y</mi><mo>)</mo><msub><mo>&#x2225;</mo><mn>2</mn></msub><mo>&#x2264;</mo><mi>L</mi><mo>&#x2225;</mo><mi>x</mi><mo>-</mo><mi>y</mi><msub><mo>&#x2225;</mo><mn>2</mn></msub><mo>,</mo><mo>&#x2200;</mo><mi>x</mi><mo>,</mo><mi>y</mi></mstyle></math>","origin":"MathType for Microsoft Add-in"}

**Chứng minh**

**Trường hợp 1: Sự hội tụ của gradient descent với learning rate đã được tối ưu**.

Giả sử hàm f(x) đã thỏa 2 điều kiện trên. Nếu ta chạy gradient descent cho k lần với learning rate . Ta sẽ chứng minh được phương trình sau:



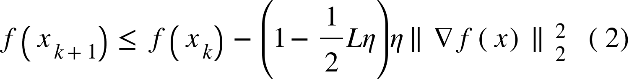
Trong đó, x\* là giá trị cực tiểu, giá trị tối ưu nhât mà ta cần tìm, {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><mi>&#x3B7;</mi></math>","origin":"MathType for Microsoft Add-in"} là learning rate và k là số vòng lặp. Khi chứng minh được công thức này thì có nghĩa là gradient descent sẽ hội tụ tại x\*.

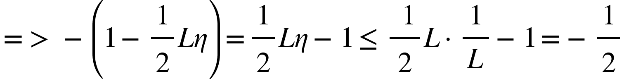
Nếu hàm có đủ 2 điều kiện trên, ta có thể biểu diễn Quadratic Approximation như sau:

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mrow><mo>&#xA0;</mo><mi>f</mi><mo>(</mo><msub><mi>x</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub><mo>)</mo><mo>&#xA0;</mo><mo>&#x2264;</mo><mo>&#xA0;</mo><mi>f</mi><mo>(</mo><msub><mi>x</mi><mi>k</mi></msub><mo>)</mo><mo>&#xA0;</mo><mo>+</mo><mo>&#x3008;</mo><msub><mi>x</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub><mo>&#xA0;</mo><mo>&#x2212;</mo><mo>&#xA0;</mo><msub><mi>x</mi><mi>k</mi></msub><mo>,</mo><mo>&#xA0;</mo><mo>&#x2207;</mo><mi>f</mi><mo>&#xA0;</mo><mo>(</mo><mi>x</mi><mo>)</mo><mo>&#x3009;</mo><mo>&#xA0;</mo><mo>+</mo><mfrac><mrow><mo>&#xA0;</mo><mi>L</mi><mo>&#xA0;</mo></mrow><mn>2</mn></mfrac><mo>&#xB7;</mo><mo>|</mo><mo>|</mo><msub><mi>x</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub><mo>-</mo><msub><mi>x</mi><mi>k</mi></msub><mo>|</mo><msubsup><mo>|</mo><mn>2</mn><mn>2</mn></msubsup></mrow><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mfenced><mn>1</mn></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"}

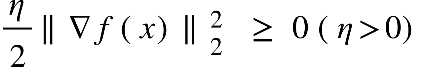
Với {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>x</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub><mo>=</mo><mo>&#xA0;</mo><msub><mi>x</mi><mi>k</mi></msub><mo>-</mo><mi>&#x3B7;</mi><mo>&#x2207;</mo><mi>f</mi><mfenced><msub><mi>x</mi><mi>k</mi></msub></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"}

Ta chứng minh được công thức:



Mà ta có giá trị learning rate , 

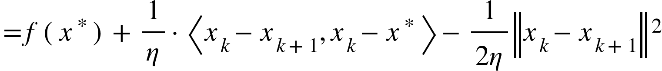
Từ đó, ta có {"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>f</mi><mfenced><msub><mi>x</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub></mfenced><mo>&#x2264;</mo><mo>&#xA0;</mo><mi>f</mi><mfenced><msub><mi>x</mi><mi>k</mi></msub></mfenced><mo>-</mo><mfrac><mi>&#x3B7;</mi><mn>2</mn></mfrac><mo>&#x2225;</mo><mo>&#x2207;</mo><mi>f</mi><mfenced><mi>x</mi></mfenced><msubsup><mo>&#x2225;</mo><mn>2</mn><mn>2</mn></msubsup><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mfenced><mn>3</mn></mfenced></mstyle></math>","origin":"MathType for Microsoft Add-in"}

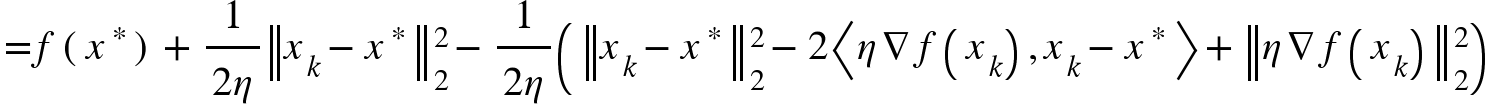
Vì nên{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>f</mi><mfenced><msub><mi>x</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub></mfenced><mo>-</mo><mi>f</mi><mfenced><msub><mi>x</mi><mi>k</mi></msub></mfenced><mo>&#x2264;</mo><mo>&#xA0;</mo><mo>-</mo><mfrac><mi>&#x3B7;</mi><mn>2</mn></mfrac><mo>&#x2225;</mo><mo>&#x2207;</mo><mi>f</mi><mfenced><mi>x</mi></mfenced><msubsup><mo>&#x2225;</mo><mn>2</mn><mn>2</mn></msubsup><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#xA0;</mo><mo>&#x2264;</mo><mn>0</mn></mstyle></math>","origin":"MathType for Microsoft Add-in"}

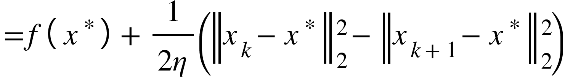
=> Vậy hàm số sau là hàm giảm=> chứng minh được sự hội tụ của hàm Gradient Descent.

Xét sự hội tụ của hàm gradient descent ở giá trị x\*, ta có:

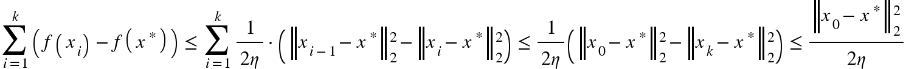
{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>f</mi><mfenced><msub><mi>x</mi><mrow><mi>k</mi><mo>+</mo><mn>1</mn></mrow></msub></mfenced><mo>&#x2264;</mo><mi>f</mi><mfenced><msup><mi>x</mi><mo>*</mo></msup></mfenced><mo>+</mo><mfenced open=\"&lt;\" close=\"&gt;\"><mrow><mo>&#x2207;</mo><mi>f</mi><mfenced><msub><mi>x</mi><mi>k</mi></msub></mfenced><mo>,</mo><msub><mi>x</mi><mi>k</mi></msub><mo>-</mo><msup><mi>x</mi><mo>*</mo></msup></mrow></mfenced><mo>-</mo><mfrac><mi>&#x3B7;</mi><mn>2</mn></mfrac><msubsup><mfenced open=\"||\" close=\"||\"><mrow><mo>&#x2207;</mo><mi>f</mi><mfenced><msub><mi>x</mi><mi>k</mi></msub></mfenced></mrow></mfenced><mn>2</mn><mn>2</mn></msubsup></mstyle></math>","origin":"MathType for Microsoft Add-in"}

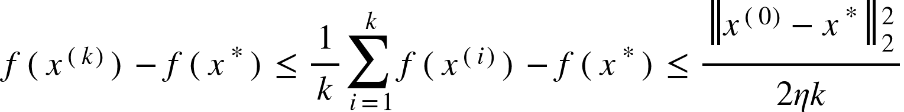






Đây là giá trị ở 1 điểm x, nếu ta tính tổng từ x1 tới ....xk thì:

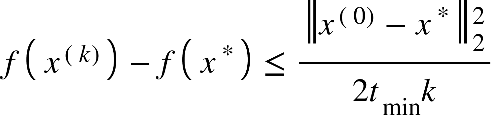


Ta có: 

Công thức này là một kết quả quan trọng trong việc hiểu sự hội tụ của thuật Gradient Descent (GD). Nó cho thấy rằng sai số của giá trị hàm mục tiêu tại bước lặp k có thể được giới hạn dưới bởi trung bình cộng của sai số tại các bước lặp trước. Điều này có nghĩa là khi ta tiến hành thêm bước lặp mới, sai số tại bước lặp mới sẽ không vượt quá một giới hạn dựa trên sai số trung bình của các bước lặp trước đó. Nó giúp chúng ta hiểu rằng khi ta tăng số lượng bước lặp (k), sai số giữa giá trị tối ưu và giá trị mục tiêu tại bước lặp tiếp theo sẽ giảm.

**Trường hợp 2: Sự hội tụ với giá trị learning rate tùy chỉnh**

Ở trường hợp này, việc sử dụng gradient descent và việc lựa chọn learning rate dựa trên backtracking line search. Điều kiện tiên quyết để áp dụng định lý này là hàm mục tiêu f(x) phải là hàm lõm (convex), có khả năng khái quát (differentiable), và đạo hàm của nó phải thỏa mãn tính chất Lipschitz liên tục với hằng số L. Tính chất này đảm bảo sự trôi chảy của thuật toán trong không gian các điểm x.



Giá trị của t(min) được tính như sau:

{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msub><mi>t</mi><mi>min</mi></msub><mo>=</mo><mi>min</mi><mo>{</mo><mn>1</mn><mo>,</mo><mi>&#x3B2;</mi><mo>/</mo><mi>L</mi><mo>}</mo></mstyle></math>","origin":"MathType for Microsoft Add-in"}

- {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><mi>&#x3B2;</mi></math>","origin":"MathType for Microsoft Add-in"} là một hằng số dương được chọn trước (có thể là một giá trị cố định) và thường được sử dụng để kiểm soát việc chọn {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><msub><mi>t</mi><mi>min</mi></msub></math>","origin":"MathType for Microsoft Add-in"}.

Do đó, để tìm giá trị của {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><msub><mi>t</mi><mi>min</mi></msub></math>","origin":"MathType for Microsoft Add-in"}, bạn cần xem xét cả hai điều kiện:

1. Nếu {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><mn>1</mn><mo>&lt;</mo><mi>&#x3B2;</mi><mo>/</mo><mi>L</mi></math>","origin":"MathType for Microsoft Add-in"}, thì {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><msub><mi>t</mi><mi>min</mi></msub><mo>=</mo><mn>1</mn></math>","origin":"MathType for Microsoft Add-in"}.

2. Nếu {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><mi>&#x3B2;</mi><mo>/</mo><mi>L</mi><mo>&lt;</mo><mn>1</mn></math>","origin":"MathType for Microsoft Add-in"}, thì {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><msub><mi>t</mi><mi>min</mi></msub><mo>=</mo><mi>&#x3B2;</mi><mo>/</mo><mi>L</mi></math>","origin":"MathType for Microsoft Add-in"}.

Lựa chọn t sẽ ảnh hưởng đến tốc độ hội tụ của thuật toán. Nếu bạn muốn tối ưu hóa tốc độ hội tụ, bạn có thể điều chỉnh giá trị của {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><mi>&#x3B2;</mi><mo>&#xA0;</mo></math>","origin":"MathType for Microsoft Add-in"}để đạt được giá trị tối ưu cho {"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><msub><mi>t</mi><mi>min</mi></msub></math>","origin":"MathType for Microsoft Add-in"}dựa trên bài toán cụ thể của bạn.

### 6. Tài liệu tham khảo

[Machine Learning cơ bản (machinelearningcoban.com)](https://machinelearningcoban.com/2017/01/16/gradientdescent2/#vi-du-voi-bai-toan-linear-regression)

<https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf>

<https://courses.cs.washington.edu/courses/cse546/15au/lectures/lecture09_optimization.pdf>

[06-grad-desc-analysis.pdf (gatech.edu)](https://mdav.ece.gatech.edu/ece-6270-spring2021/notes/06-grad-desc-analysis.pdf)

[lecture3.pdf (princeton.edu)](https://www.cs.princeton.edu/courses/archive/fall18/cos597G/lecnotes/lecture3.pdf)

<https://raghumeka.github.io/CS289ML/gdnotes.pdf>