**ĐẠI HỌC HUẾ**

# **KHOA KỸ THUẬT VÀ CÔNG NGHỆ**



🙠🙟🕮🙝🙢

****

**BÁO CÁO**

**ĐỒ ÁN**

**NĂM HỌC 2022-2023**

**Giảng viên hướng dẫn: Hoàng Trọng Lợi**

**Lớp: KHDL&TTNT**

|  |
| --- |
| Số phách  *(Do hội đồng chấm thi ghi)* |

**Thừa Thiên Huế, ngày …tháng…năm.....**

**ĐẠI HỌC HUẾ**



# KHOA KỸ THUẬT VÀ CÔNG NGHỆ

🙠🙟🕮🙝🙢

****

(MẪU BÌA PHỤ)

**BÁO CÁO ĐỒ ÁN**

**NĂM HỌC 2022-2023**

**Giảng viên hướng dẫn: Hoàng Trọng Lợi**

**Lớp: KHDL&TTNT**

**Sinh viên thực hiện: Phan Thế Minh Châu**

**MSV: 20E1020067**

*(ký tên và ghi rõ họ tên)*

|  |
| --- |
| Số phách  *(Do hội đồng chấm thi ghi)* |

**Thừa Thiên Huế, ngày …tháng…năm.....**

ĐẠI HỌC HUẾ

# KHOA KỸ THUẬT VÀ CÔNG NGHỆ

🙠🙟🕮🙝🙢

**PHIẾU ĐÁNH GIÁ ĐỒ ÁN**

**Học kỳ II, năm học 2022 - 2023**

|  |  |
| --- | --- |
| **Cán bộ chấm thi 1** | **Cán bộ chấm thi 2** |
| **Nhận xét:**                      **Điểm đánh giá của CBCT1:**  Bằng số:  Bằng chữ: | **Nhận xét:**                      **Điểm đánh giá của CBCT2:**  Bằng số:  Bằng chữ: |

Điểm kết luận:

Bằng số:

Bằng chữ:

*Thừa Thiên Huế, ngày tháng năm 2023*

|  |  |
| --- | --- |
| **Cán bộ chấm thi 1**  *(Ký và ghi rõ họ và tên)* | **Cán bộ chấm thi 2**  *(Ký và ghi rõ họ và tên)* |

Mục lục

[I. Giới thiệu 5](#_Toc135226519)

[A. Tổng quan về bài toán hồi quy tuyến tính 5](#_Toc135226520)

[B. Giới thiệu thuật toán Gradient Descent 5](#_Toc135226521)

[C. Mục tiêu và cấu trúc của báo cáo 6](#_Toc135226522)

[II. Cơ bản về thuật toán Gradient Descent 7](#_Toc135226523)

[A. Định nghĩa 7](#_Toc135226524)

[B. Phân loại Gradient Descent 7](#_Toc135226525)

[C. Cách hoạt động của Gradient Descent 8](#_Toc135226526)

[D. Gradient Descent đa biến 9](#_Toc135226527)

[III. Ứng dụng của thuật toán Gradient Descent cho bài toán hồi quy tuyến tính 10](#_Toc135226528)

[IV. Các phương pháp cải tiến của thuật toán Gradient Descent 12](#_Toc135226529)

[A. Momentum 12](#_Toc135226530)

[B. Nesterov Accelerated Gradient 13](#_Toc135226531)

[C. Adagrad 13](#_Toc135226532)

[D. RMSProp 14](#_Toc135226533)

[V. Ứng dụng của thuật toán Gradient Descent 15](#_Toc135226534)

[A. Ứng dụng trong Machine Learning 15](#_Toc135226535)

[B. Ứng dụng trong Computer Vision 16](#_Toc135226536)

[C. Ứng dụng trong Natural Language Processing 17](#_Toc135226537)

[VI. Triển khai thuật toán Gradient Descent bằng Python 18](#_Toc135226538)

[A. Cài đặt thuật toán Gradient Descent với Python 18](#_Toc135226539)

[VII. Kết luận 27](#_Toc135226540)

[A. Tóm tắt các kết quả của báo cáo 27](#_Toc135226541)

[B. Những hạn chế và hướng phát triển cho thuật toán Gradient Descent và các phương pháp cải tiến 28](#_Toc135226542)

[C. Tầm quan trọng của thuật toán Gradient Descent trong Machine Learning và các lĩnh vực ứng dụng khác. 28](#_Toc135226543)

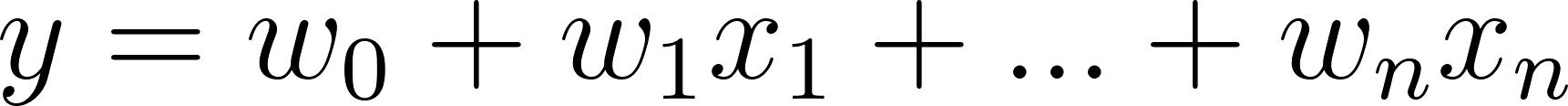
[VIII. Tài liệu tham khảo 30](#_Toc135226544)

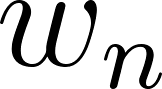
# **I. Giới thiệu**

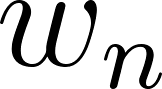
## A. Tổng quan về bài toán hồi quy tuyến tính

Bài toán hồi quy tuyến tính là một trong những bài toán quan trọng nhất trong lĩnh vực học máy, vì nó được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, từ kinh tế, tài chính, y tế, đến khoa học xã hội và kỹ thuật. Để giải quyết bài toán hồi quy tuyến tính, ta cần cố gắng tìm một mối quan hệ tuyến tính giữa các biến đầu vào và biến đầu ra. Mục tiêu là xây dựng một mô hình có thể dự đoán giá trị đầu ra cho các giá trị đầu vào mới, dựa trên quan hệ tuyến tính đã được xác định từ dữ liệu huấn luyện.

Cụ thể, hàm số dự đoán trong bài toán hồi quy tuyến tính có dạng

[](https://www.codecogs.com/eqnedit.php?latex=y%20%3D%20w_%7B0%7D%20%2B%20w_%7B1%7Dx_%7B1%7D%20%2B%20...%20%2B%20w_%7Bn%7Dx_%7Bn%7D#0)

Trong đó, y là giá trị dự đoán của biến phụ thuộc, [](https://www.codecogs.com/eqnedit.php?latex=w_%7B0%7D#0) là hệ số điều chỉnh (bias), [](https://www.codecogs.com/eqnedit.php?latex=w_%7B1%7D#0) đến [](https://www.codecogs.com/eqnedit.php?latex=w_%7Bn%7D#0) là các hệ số hồi quy (weights), và [](https://www.codecogs.com/eqnedit.php?latex=x_%7B1%7D#0) đến [](https://www.codecogs.com/eqnedit.php?latex=x_%7Bn%7D#0) là các giá trị của biến độc lập.

Tập dữ liệu huấn luyện được sử dụng để tính toán các giá trị [](https://www.codecogs.com/eqnedit.php?latex=w_%7B0%7D#0) đến [](https://www.codecogs.com/eqnedit.php?latex=w_%7Bn%7D#0), thông qua việc tối ưu hàm mất mát (loss function) của mô hình. Hàm mất mát thường được chọn là hàm bình phương sai số (mean squared error), được tính dựa trên sai số giữa giá trị dự đoán và giá trị thực tế của biến phụ thuộc trong tập dữ liệu huấn luyện.

## B. Giới thiệu thuật toán Gradient Descent

Một trong những phương pháp phổ biến nhất để tối ưu mô hình hồi quy tuyến tính là sử dụng thuật toán Gradient Descent. Đây là một trong những phương pháp phổ biến nhất để tối ưu hàm mục tiêu trong machine learning. Thuật toán này được sử dụng để tìm giá trị của bộ tham số tối ưu trong bài toán hồi quy tuyến tính và các bài toán tối ưu khác.

Ý tưởng chính của Gradient Descent là điều chỉnh bộ tham số dựa trên độ lỗi giữa đầu ra dự đoán và đầu ra thực tế của mô hình. Thuật toán cập nhật bộ tham số bằng cách di chuyển đến hướng ngược lại của đạo hàm của hàm mục tiêu. Cụ thể, thuật toán sẽ tính đạo hàm riêng của hàm mục tiêu tại điểm hiện tại, sau đó di chuyển đi một khoảng cách nhỏ trong hướng ngược lại của đạo hàm đó.

Mục tiêu của Gradient Descent là tìm giá trị của bộ tham số sao cho độ lỗi của mô hình là nhỏ nhất. Tuy nhiên, việc cập nhật bộ tham số dựa trên toàn bộ tập dữ liệu đòi hỏi chi phí tính toán lớn và không hiệu quả đối với các bộ dữ liệu lớn. Do đó, nhiều phương pháp tối ưu Gradient Descent đã được đưa ra để giải quyết vấn đề này, trong đó phương pháp Stochastic Gradient Descent (SGD) là một trong những phương pháp phổ biến nhất.

## C. Mục tiêu và cấu trúc của báo cáo

Bài báo cáo sẽ tập trung vào việc giới thiệu và phân tích chi tiết về thuật toán Gradient Descent và ứng dụng của nó trong bài toán hồi quy tuyến tính. Trước khi đi vào phần chi tiết của thuật toán và ứng dụng của nó, báo cáo sẽ bao gồm một tổng quan về bài toán hồi quy tuyến tính và thuật toán Gradient Descent.

Phần hai của báo cáo sẽ giải thích cơ bản về thuật toán Gradient Descent, bao gồm định nghĩa, phân loại và cách hoạt động.

Phần ba sẽ tập trung vào ứng dụng của thuật toán Gradient Descent trong bài toán hồi quy tuyến tính, bao gồm các bước thực hiện thuật toán, tính toán đạo hàm và cập nhật các tham số, và sử dụng thuật toán Gradient Descent để tìm nghiệm bài toán hồi quy tuyến tính.

Phần bốn tập trung giới thiệu các phương pháp cải tiến của thuật toán Gradient Descent, bao gồm Momentum, Nesterov Accelerated Gradient, Adagrad, RMSProp, và Adam.

Trong phần năm, báo cáo cung cấp một số ví dụ thực tế về ứng dụng của thuật toán Gradient Descent trong Machine Learning, Computer Vision và Natural Language Processing.

Hướng dẫn cài đặt thuật toán Gradient Descent cho hồi quy tuyến tính đa biến bằng Python cho bài toán dự báo giá nhà sẽ được nêu chi tiết trong phần sáu.

Cuối cùng, phần bảy của báo cáo sẽ tóm tắt các kết quả của báo cáo, đưa ra những hạn chế và hướng phát triển cho thuật toán Gradient Descent và các phương pháp cải tiến, và đánh giá tầm quan trọng của thuật toán Gradient Descent trong Machine Learning và các lĩnh vực ứng dụng.

# **II. Cơ bản về thuật toán Gradient Descent**

## A. Định nghĩa

Gradient Descent là một thuật toán tối ưu hóa được sử dụng phổ biến trong Machine Learning. Nó được áp dụng để tìm giá trị tối ưu của một hàm số bằng cách tìm kiếm đường dốc lớn nhất của hàm số này. Thuật toán Gradient Descent hoạt động dựa trên việc tính toán đạo hàm của hàm số tại một điểm xác định, sau đó cập nhật giá trị của điểm đó dựa trên giá trị đạo hàm tính được. Việc lặp lại quá trình này cho đến khi đạt được điểm dừng sẽ cho kết quả giá trị tối ưu của hàm số.

Gradient Descent có nhiều ứng dụng trong Machine Learning, trong đó bao gồm cả việc tìm kiếm các trọng số tối ưu cho một mô hình học máy. Thuật toán này được sử dụng rộng rãi vì tính đơn giản của nó và khả năng ứng dụng cho nhiều loại hàm số và mô hình. Tuy nhiên, Gradient Descent cũng có một số hạn chế, bao gồm độ chậm của thuật toán khi tiến tới giá trị tối ưu và việc lựa chọn learning rate phù hợp để đạt được kết quả tốt nhất. Do đó, nhiều phương pháp cải tiến của Gradient Descent đã được phát triển để cải thiện hiệu suất của thuật toán.

## B. Phân loại Gradient Descent

Phân loại Gradient Descent là một mục quan trọng trong bài toán tối ưu hóa với Gradient Descent. Có thể phân loại Gradient Descent dựa trên cách tính toán gradient và cách cập nhật các tham số.

Gradient Descent phân loại dựa trên cách tính toán gradient thành hai loại chính: Batch Gradient Descent và Stochastic Gradient Descent.

Batch Gradient Descent là phương pháp cập nhật các tham số trong hồi quy tuyến tính bằng cách tính toán đạo hàm của hàm mất mát trên toàn bộ tập dữ liệu huấn luyện. Với cách làm này, việc tính toán sẽ tốn nhiều thời gian và tài nguyên tính toán, đặc biệt là trên các tập dữ liệu lớn. Tuy nhiên, Batch Gradient Descent có thể dẫn đến một nghiệm toàn cục chính xác nếu hàm mất mát là lồi (convex).

Stochastic Gradient Descent là phương pháp cập nhật các tham số của mô hình một cách ngẫu nhiên trên một mẫu dữ liệu (một cặp dữ liệu đầu vào và đầu ra) từ tập dữ liệu huấn luyện. Cách làm này có thể giúp giảm thời gian tính toán và tài nguyên, nhưng đồng thời cũng dẫn đến việc có thể đạt được nghiệm cục bộ. Tuy nhiên, Stochastic Gradient Descent có thể được kết hợp với các phương pháp khác như Momentum hay RMSProp để cải thiện độ chính xác của mô hình.

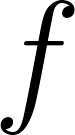
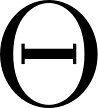
Ngoài ra, còn có một loại Gradient Descent khác, là Mini-batch Gradient Descent. Mini-batch Gradient Descent là sự kết hợp của Batch Gradient Descent và Stochastic Gradient Descent. Thay vì sử dụng toàn bộ tập dữ liệu hoặc một mẫu dữ liệu duy nhất để tính toán gradient, Mini-batch Gradient Descent sử dụng một tập con nhỏ của dữ liệu để tính toán gradient và cập nhật các tham số.

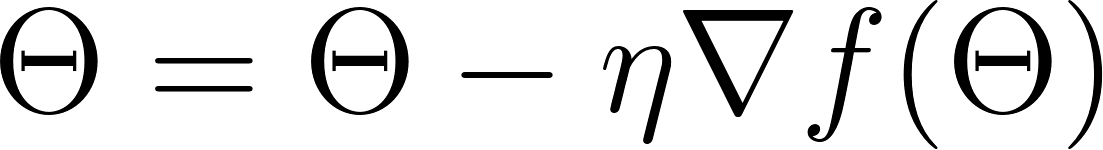
Về phân loại Gradient Descent dựa trên cách cập nhật các tham số, chúng ta có thể chia làm hai loại chính: Gradient Descent thông thường và Gradient Descent với momentum. Gradient Descent thông thường cập nhật các tham số bằng cách di chuyển theo hướng ngược với gradient của hàm mất mát. Trong khi đó, Gradient Descent với momentum sử dụng một động lực để giúp cho quá trình cập nhật tham số được ổn định hơn và tránh bị mắc kẹt ở các cực tiểu địa phương.

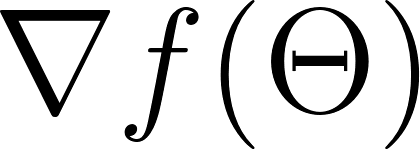
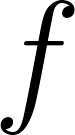
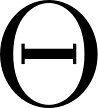
## C. Cách hoạt động của Gradient Descent

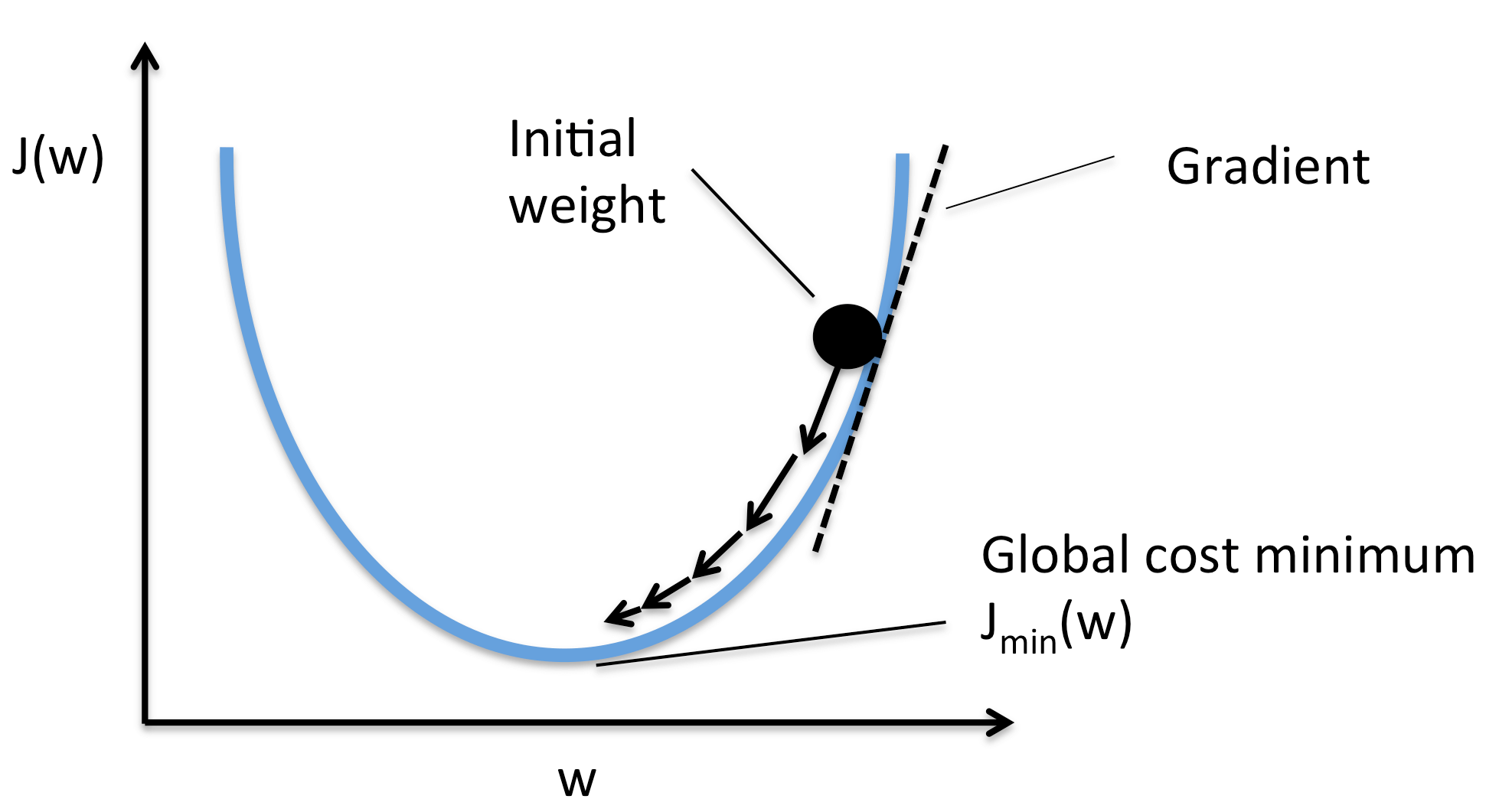
Gradient Descent là một thuật toán tối ưu hóa được sử dụng để tìm giá trị tối ưu của một hàm số có thể đo được. Thuật toán Gradient Descent sử dụng gradient (đạo hàm riêng) của hàm số tại điểm hiện tại để điều chỉnh giá trị của các tham số trong một mô hình, giúp mô hình tiến gần đến giá trị tối ưu.

Cách hoạt động của Gradient Descent được chia thành hai bước chính: tính gradient và cập nhật tham số. Bước tính gradient là quá trình tính toán đạo hàm riêng của hàm số tại điểm hiện tại. Bước cập nhật tham số sử dụng gradient để điều chỉnh giá trị của các tham số sao cho hàm số đạt giá trị tối ưu.

Giả sử [](https://www.codecogs.com/eqnedit.php?latex=f#0) là hàm số cần tối ưu,  là tập hợp các tham số trong mô hình,  là learning rate (tốc độ học). Bước cập nhật tham số được thực hiện bằng công thức:

[](https://www.codecogs.com/eqnedit.php?latex=%5CTheta%20%3D%20%5CTheta%20-%20%5Ceta%20%5Cnabla%20f(%5CTheta)#0)

Trong đó, [](https://www.codecogs.com/eqnedit.php?latex=%5Cnabla%20f(%5CTheta)#0) là gradient của hàm số [](https://www.codecogs.com/eqnedit.php?latex=f#0) tại điểm [](https://www.codecogs.com/eqnedit.php?latex=%5CTheta#0)



Hình 1: Mô phỏng thuật toán Gradient Descent

Để đạt được giá trị tối ưu của hàm số, Gradient Descent sẽ tiếp tục lặp lại quá trình tính gradient và cập nhật tham số cho đến khi đạt được điều kiện dừng, chẳng hạn như đạt đến một số lần lặp tối đa hoặc đạt đến một giá trị độ chính xác nhất định.

Thuật toán Gradient Descent có thể được áp dụng trên nhiều loại mô hình khác nhau, chẳng hạn như mô hình hồi quy tuyến tính hay mô hình mạng nơ-ron. Tuy nhiên, cách hoạt động của thuật toán vẫn giữ nguyên và sử dụng cùng một công thức để tính gradient và cập nhật tham số.

## D. Gradient Descent đa biến

Gradient Descent đơn biến và Gradient Descent đa biến là hai biến thể của thuật toán Gradient Descent, nhưng có một số khác biệt quan trọng giữa chúng:

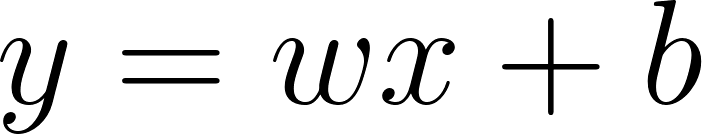
* Gradient Descent đơn biến:
* Áp dụng cho hàm mục tiêu có một biến độc lập.
* Chỉ có một hệ số (tham số) cần được cập nhật để tìm kiếm điểm tối ưu.
* Quá trình Gradient Descent đơn biến tính toán gradient (đạo hàm) của hàm mục tiêu theo biến độc lập và cập nhật hệ số dựa trên gradient đó.
* Gradient Descent đa biến:
* Áp dụng cho hàm mục tiêu có nhiều biến độc lập.
* Có nhiều hệ số (tham số) cần được cập nhật để tìm kiếm điểm tối ưu.
* Quá trình Gradient Descent đa biến tính toán gradient (đạo hàm riêng) của hàm mục tiêu theo từng biến độc lập và cập nhật các hệ số dựa trên gradient đó.

Với Gradient Descent đa biến, việc tính toán gradient và cập nhật các hệ số trở nên phức tạp hơn so với Gradient Descent đơn biến, vì phải tính toán đạo hàm riêng theo từng biến. Tuy nhiên, Gradient Descent đa biến cho phép tìm kiếm điểm tối ưu trong không gian đa chiều, mô hình hóa mối quan hệ phức tạp hơn giữa các biến độc lập và có khả năng giải quyết các bài toán phức tạp hơn.

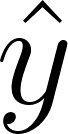
# **III. Ứng dụng của thuật toán Gradient Descent cho bài toán hồi quy tuyến tính**

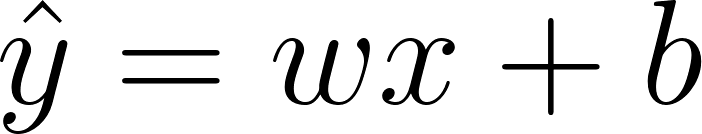
Phần lớn của đề tài này tập trung vào việc tìm hiểu về thuật toán Gradient Descent và ứng dụng của nó trong bài toán hồi quy tuyến tính. Trong phần này, báo cáo sẽ trình bày chi tiết các bước thực hiện thuật toán Gradient Descent cho bài toán hồi quy tuyến tính.

1. Khởi tạo tham số

Trước khi thực hiện thuật toán Gradient Descent, ta cần khởi tạo giá trị ban đầu cho các tham số. Thông thường, ta có thể khởi tạo giá trị của các tham số là các giá trị ngẫu nhiên hoặc bằng 0. Trong trường hợp này, ta cần khởi tạo giá trị ban đầu cho các tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0) của phương trình hồi quy tuyến tính [](https://www.codecogs.com/eqnedit.php?latex=y%20%3D%20wx%20%2B%20b#0). Ở đây, [](https://www.codecogs.com/eqnedit.php?latex=w#0) là một vectơ tham số, [](https://www.codecogs.com/eqnedit.php?latex=b#0) là giá trị bias của phương trình tuyến tính đang xét.

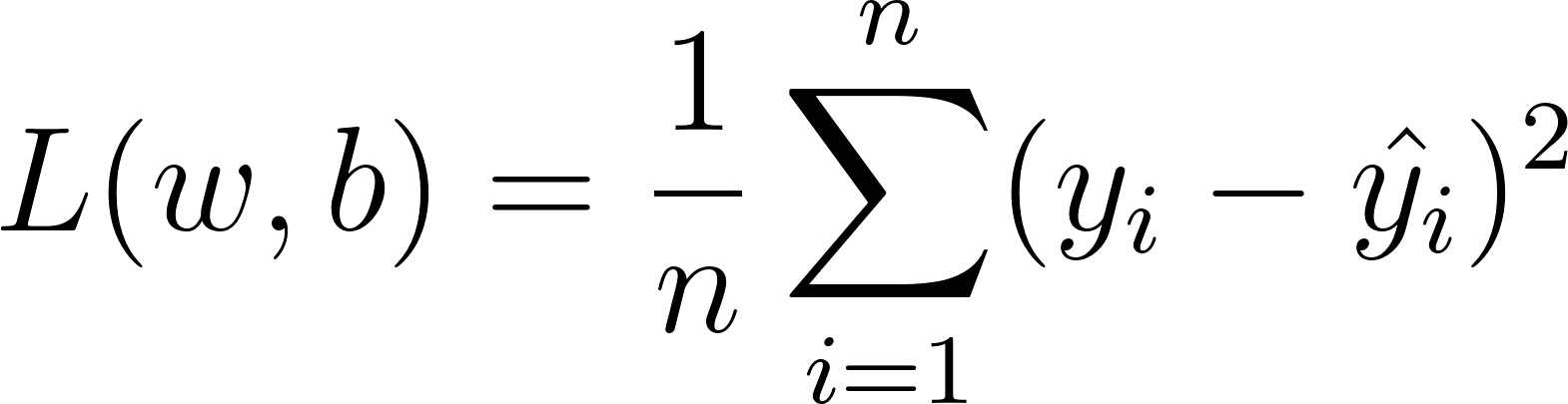
1. Tính giá trị dự đoán

Với mỗi mẫu dữ liệu, ta tính giá trị dự đoán [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7By%7D#0) bằng cách áp dụng phương trình hồi quy tuyến tính vào giá trị của biến đầu vào [](https://www.codecogs.com/eqnedit.php?latex=x#0) và các tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0) như sau:

[](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7By%7D%3Dwx%2Bb#0)

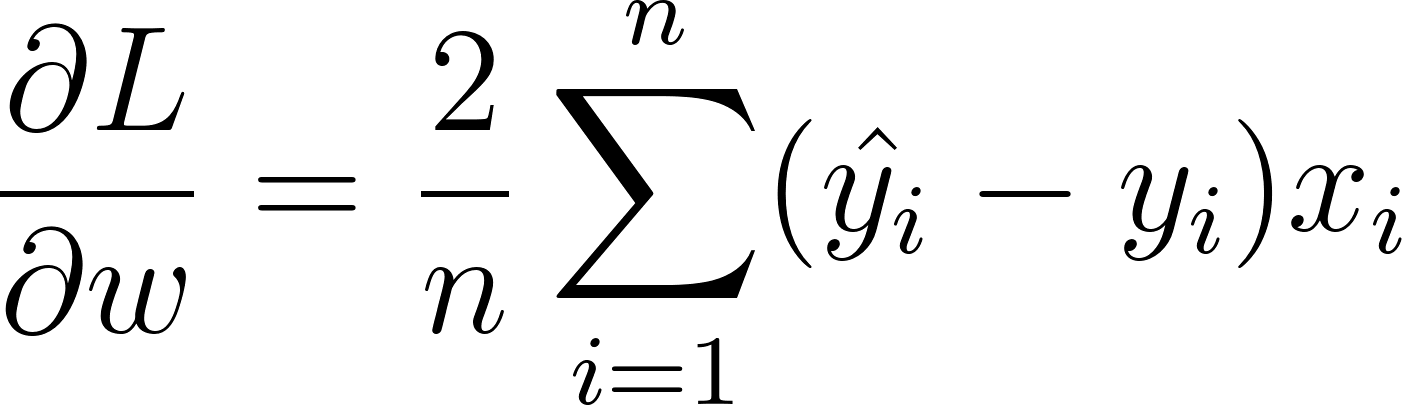
1. Tính độ lỗi của giá trị dự đoán

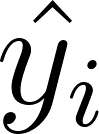
Sau khi có giá trị dự đoán được tính bằng phương trình tuyến tình với các tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0) hiện có, bước tiếp theo cần tính độ lỗi của mô hình dự đoán bằng cách sử dụng hàm mất mát (loss function). Giả sử trong trường hợp này, sử dụng hàm mất mát là Mean Squared Error (MSE) để tính độ lỗi cho bài toán hồi quy tuyến tính. Hàm MSE được định nghĩa như sau:

[](https://www.codecogs.com/eqnedit.php?latex=L(w%2Cb)%3D%5Cfrac%7B1%7D%7Bn%7D%5Csum_%7Bi%3D1%7D%5E%7Bn%7D(y_i-%5Chat%7By_i%7D)%5E2#0)

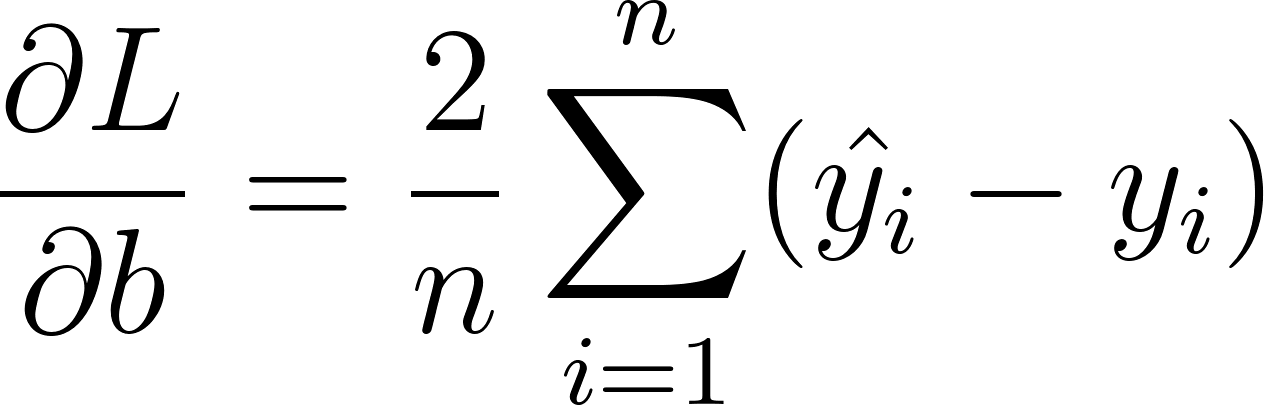
1. Tính gradient

Sau khi tính được giá trị của hàm mất mát (loss), tiếp theo chúng ta cần tính toán gradient của hàm mất mát đó đối với các tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0). Gradient được tính bằng cách lấy đạo hàm riêng của hàm mất mát đó đối với từng tham số. Trong trường hợp của bài toán hồi quy tuyến tính với hàm mất mát MSE, ta có thể tính gradient đối với tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) bằng cách lấy đạo hàm riêng của hàm MSE đối với [](https://www.codecogs.com/eqnedit.php?latex=w#0), được tính như sau:

[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B%5Cpartial%20L%7D%7B%5Cpartial%20w%7D%3D%5Cfrac%7B2%7D%7Bn%7D%5Csum_%7Bi%3D1%7D%5E%7Bn%7D(%5Chat%7By_i%7D-y_i)x_i#0)

Với [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7By_i%7D#0) là giá trị đầu ra dự đoán tương ứng với [](https://www.codecogs.com/eqnedit.php?latex=x_i#0), [](https://www.codecogs.com/eqnedit.php?latex=y_i#0) là giá trị đầu ra thực tế của dữ liệu thứ [](https://www.codecogs.com/eqnedit.php?latex=i#0), [](https://www.codecogs.com/eqnedit.php?latex=n#0) là số lượng mẫu dữ liệu, và [](https://www.codecogs.com/eqnedit.php?latex=x_i#0) là giá trị đầu vào tương ứng với mẫu dữ liệu thứ [](https://www.codecogs.com/eqnedit.php?latex=i#0)

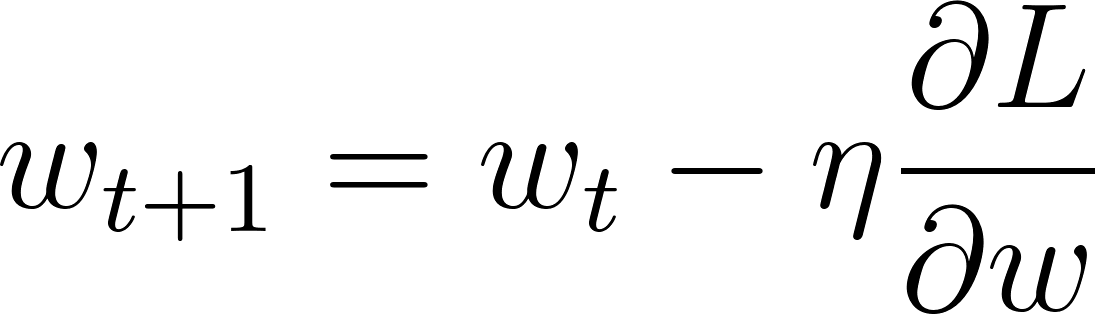
Tương tự, gradient của hàm mất mát đối với tham số [](https://www.codecogs.com/eqnedit.php?latex=b#0) có thể được tính bằng cách lấy đạo hàm riêng của hàm MSE đối với [](https://www.codecogs.com/eqnedit.php?latex=b#0), được tính như sau:

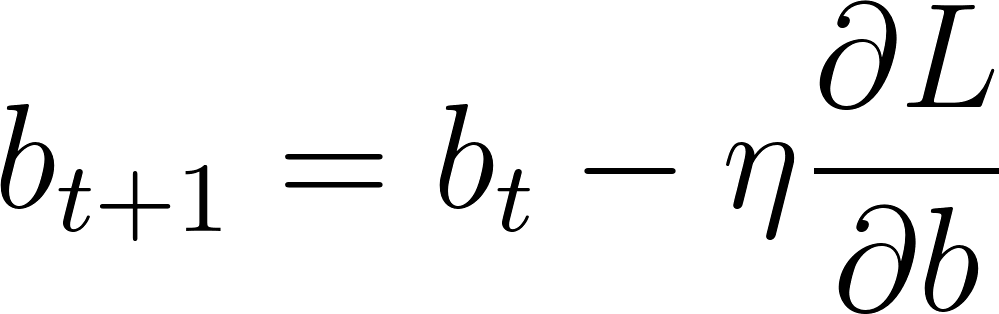
[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B%5Cpartial%20L%7D%7B%5Cpartial%20b%7D%3D%5Cfrac%7B2%7D%7Bn%7D%5Csum_%7Bi%3D1%7D%5E%7Bn%7D(%5Chat%7By_i%7D-y_i)#0)

Tại mỗi vòng lặp của thuật toán Gradient Descent, ta sẽ sử dụng các giá trị gradient này để cập nhật các tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0) cho mô hình.

1. Cập nhật tham số

Sau khi tính được gradient, bước cuối cùng cần cập nhật các tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0) mới theo công thức sau:

[](https://www.codecogs.com/eqnedit.php?latex=w_%7Bt%2B1%7D%20%3D%20w_t%20-%20%5Ceta%5Cfrac%7B%5Cpartial%20L%7D%7B%5Cpartial%20w%7D#0)

[](https://www.codecogs.com/eqnedit.php?latex=b_%7Bt%2B1%7D%20%3D%20b_t%20-%20%5Ceta%5Cfrac%7B%5Cpartial%20L%7D%7B%5Cpartial%20b%7D#0)

Trong đó, [](https://www.codecogs.com/eqnedit.php?latex=%5Ceta#0) là hệ số learning rate, giá trị này thường được chọn là một số nhỏ dương, nhằm đảm bảo thuật toán hội tụ nhanh hơn và giảm thiểu độ lớn của các bước cập nhật. Khi giá trị của learning rate quá lớn, thuật toán sẽ dễ bị phân kỳ và không hội tụ được. Tuy nhiên, nếu giá trị của learning rate quá nhỏ, thuật toán sẽ cập nhật tham số rất chậm, dẫn đến tốc độ hội tụ chậm hơn.

Việc cập nhật tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0) được lặp lại cho đến khi hàm loss function đạt được giá trị nhỏ nhất hoặc không thay đổi đáng kể trong một số vòng lặp cho trước (epoch). Sau khi huấn luyện xong mô hình, ta sử dụng các tham số [](https://www.codecogs.com/eqnedit.php?latex=w#0) và [](https://www.codecogs.com/eqnedit.php?latex=b#0) tối ưu nhất để tính toán giá trị dự đoán trên tập dữ liệu mới.

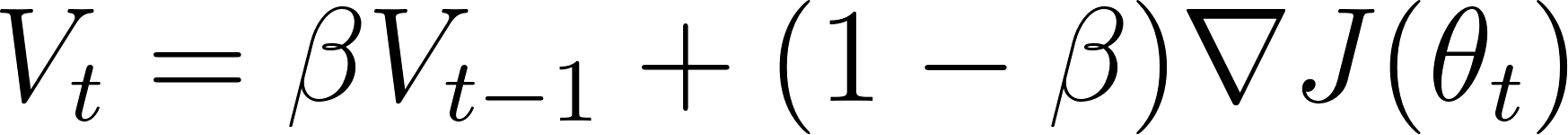
# **IV. Các phương pháp cải tiến của thuật toán Gradient Descent**

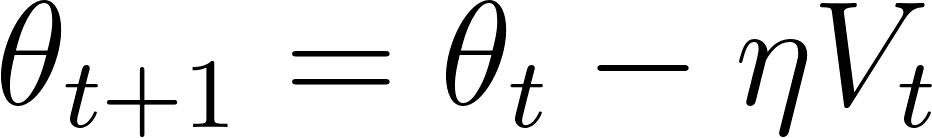
## A. Momentum

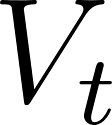
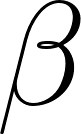
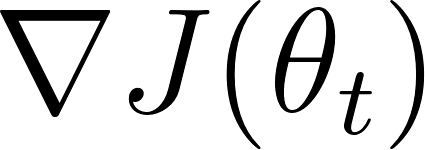
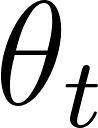
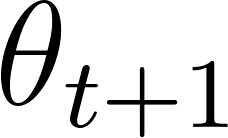
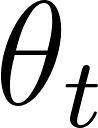
Trong phần trước, bài báo cáo đã trình bày về phương pháp Gradient Descent và các bước thực hiện Gradient Descent trong bài toán hồi quy tuyến tính. Tuy nhiên, trong quá trình tối ưu hóa, chúng ta có thể gặp phải một số khó khăn như mất mát thông tin và độ chính xác của giá trị cập nhật thấp. Để khắc phục những vấn đề này, chúng ta có thể sử dụng phương pháp Momentum - một phương pháp cái tiến của Gradient Descent.

Phương pháp Momentum sử dụng một momentum vectơ để lưu giữ thông tin về hướng của Gradient Descent trong quá khứ. Mỗi lần cập nhật tham số, hướng đi sẽ được quyết định bởi hai yếu tố: gradient hiện tại và momentum vectơ. Phương pháp Momentum cộng thêm một lượng vectơ đại diện cho quá khứ vào giá trị cập nhật hiện tại. Điều này giúp cho giá trị cập nhật tránh được việc dao động quá mức, đồng thời giúp cho quá trình tối ưu hóa được diễn ra nhanh hơn và ổn định hơn.

Công thức toán học của phương pháp Momentum được mô tả như sau:

[](https://www.codecogs.com/eqnedit.php?latex=V_%7Bt%7D%20%3D%20%5Cbeta%20V_%7Bt-1%7D%20%2B%20(1-%5Cbeta)%5Cnabla%20J(%5Ctheta_t)#0)

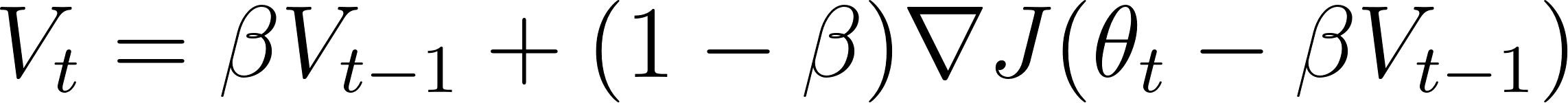
[](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_%7Bt%2B1%7D%20%3D%20%5Ctheta_t%20-%20%5Ceta%20V_%7Bt%7D#0)

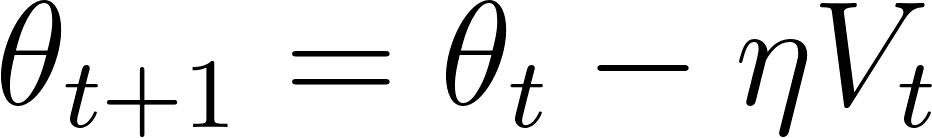
Trong đó, [](https://www.codecogs.com/eqnedit.php?latex=V_%7Bt%7D#0) là vectơ momentum tại thời điểm t, [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta#0) là hệ số giảm dần trong khoảng từ 0 đến 1, [](https://www.codecogs.com/eqnedit.php?latex=%5Cnabla%20J(%5Ctheta_t)#0) là gradient của hàm mất mát tại thời điểm t, [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_t#0) là giá trị tham số tại thời điểm t, [](https://www.codecogs.com/eqnedit.php?latex=%5Ceta#0) là learning rate. Giá trị cập nhật [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_%7Bt%2B1%7D#0) được tính bằng cách trừ vector momentum nhân với learning rate [](https://www.codecogs.com/eqnedit.php?latex=%5Ceta#0) từ giá trị tham số [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_t#0).

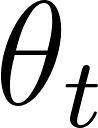
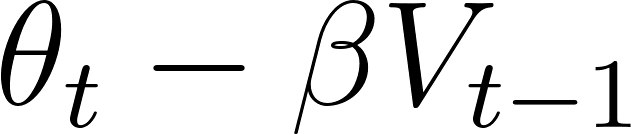
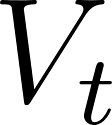
Phương pháp Momentum là một phương pháp cái tiến của Gradient Descent giúp tăng tốc quá trình tối ưu hóa và giảm độ dao động của giá trị cập nhật trong quá trình tối ưu hóa. Phương pháp này thường được sử dụng trong các bài toán tối ưu hóa với dữ liệu lớn và phức tạp, bao gồm cả bài toán hồi quy tuyến tính.

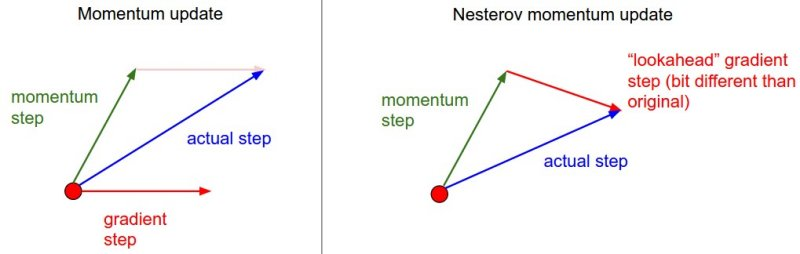
## B. Nesterov Accelerated Gradient

Phương pháp Nesterov Accelerated Gradient (NAG) là một phương pháp cái tiến của Gradient Descent, được đề xuất bởi Yurii Nesterov vào năm 1983. Phương pháp này cải tiến so với phương pháp Momentum bằng cách dự đoán trước giá trị của tham số [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0) tại thời điểm tiếp theo. Cụ thể, NAG được biểu diễn theo góc nhìn toán học như sau

[](https://www.codecogs.com/eqnedit.php?latex=V_t%20%3D%20%5Cbeta%20V_%7Bt-1%7D%20%2B%20(1-%5Cbeta)%5Cnabla%20J(%5Ctheta_t%20-%20%5Cbeta%20V_%7Bt-1%7D)#0)

[](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_%7Bt%2B1%7D%20%3D%20%5Ctheta_t%20-%20%5Ceta%20V_t#0)

Thay vì tính toán gradient của hàm mất mát tại vị trí hiện tại của tham số [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_t#0), phương pháp Nesterov sử dụng gradient của hàm mất mát tại một vị trí ước tính của [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta#0) tại thời điểm tiếp theo: [](https://www.codecogs.com/eqnedit.php?latex=%5Ctheta_t%20-%20%5Cbeta%20V_%7Bt-1%7D#0). Bằng cách này, phương pháp Nesterov có thể dự đoán được hướng của giá trị tham số sẽ di chuyển tới và điều chỉnh vector momentum [](https://www.codecogs.com/eqnedit.php?latex=V_t#0) phù hợp.



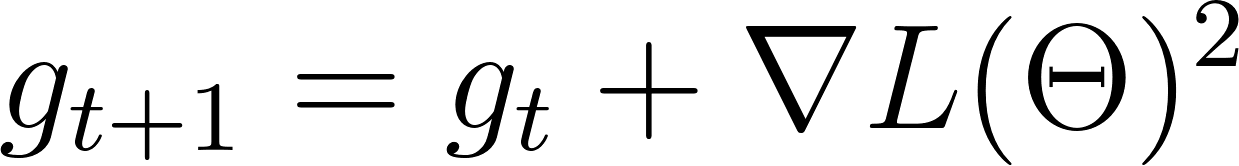
Hình 2: Ý tưởng của Momentum và Nesterov Accelerated Gradient

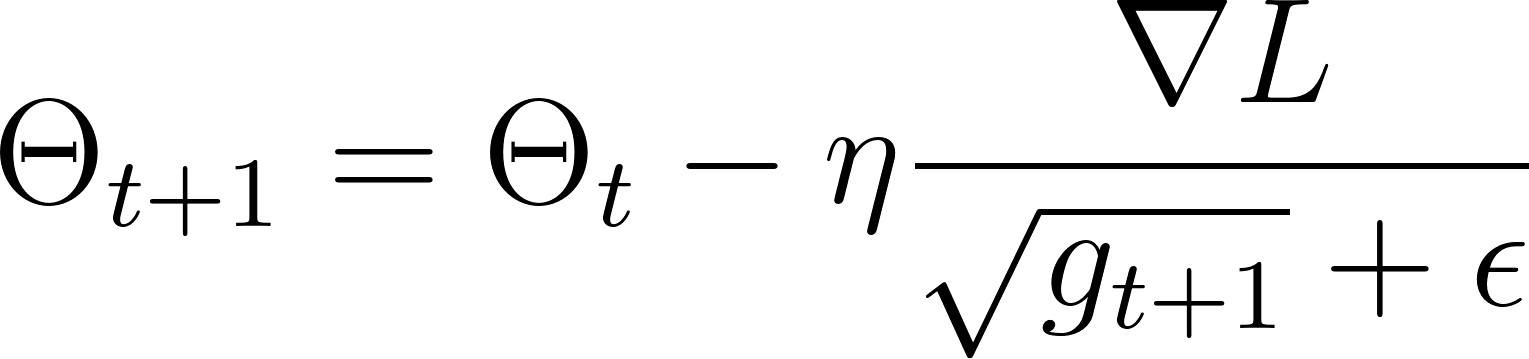
Tương tự như phương pháp Momentum, phương pháp Nesterov Accelerated Gradient giúp tăng tốc quá trình tối ưu hóa và giảm độ dao động của giá trị cập nhật trong quá trình tối ưu hóa. Đồng thời, phương pháp này còn giúp tăng tính ổn định của quá trình tối ưu hóa.

## C. Adagrad

Adagrad (Adaptive Gradient Algorithm) là một phương pháp cái tiến của Gradient Descent, được đề xuất bởi Duchi, Hazan và Singer vào năm 2011. Adagrad sử dụng một kỹ thuật điều chỉnh tốc độ học tập của từng tham số riêng biệt dựa trên lịch sử của các gradient trước đó. Cụ thể, Adagrad sử dụng một vector gradient squared tích lũy để chia nhỏ tốc độ học tập.

Công thức cập nhật của Adagrad cho bài toán tối thiểu hóa hàm mất mát L với tham số w tại bước thời gian t được cho bởi:

[](https://www.codecogs.com/eqnedit.php?latex=g_%7Bt%2B1%7D%20%3D%20g_t%20%2B%20%5Cnabla%20L(%5CTheta)%5E2#0)

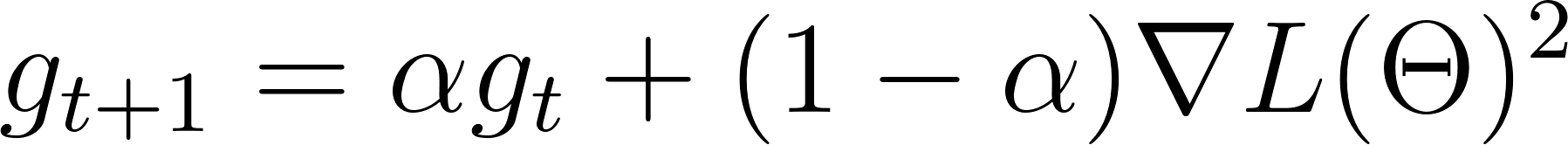
[](https://www.codecogs.com/eqnedit.php?latex=%5CTheta_%7Bt%2B1%7D%20%3D%20%5CTheta_t%20-%20%5Ceta%5Cfrac%7B%5Cnabla%20L%7D%7B%5Csqrt%7Bg_%7Bt%2B1%7D%7D%20%2B%20%5Cepsilon%20%7D#0)

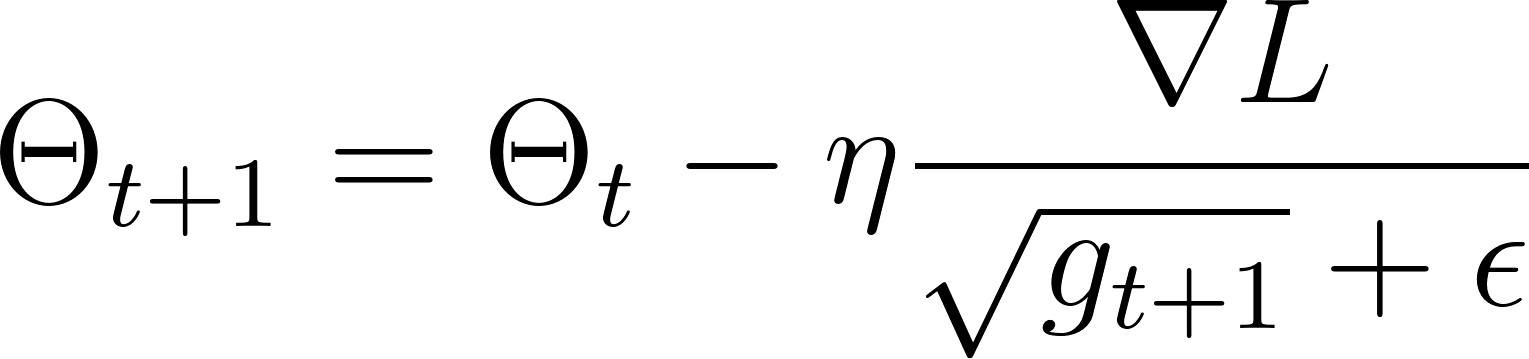
Dựa vào công thức cũng có thể thấy rằng Adagrad giảm tốc độ học tập của các tham số đã được cập nhật nhiều, vì chúng đã có xu hướng di chuyển đến điểm tối ưu và sẽ không có nhiều sự thay đổi lớn nữa. Ngược lại, các tham số chưa được cập nhật nhiều sẽ có tốc độ học tập cao hơn để giúp tìm được điểm tối ưu nhanh hơn. Phương pháp Adagrad đã được áp dụng rộng rãi trong các bài toán học máy, đặc biệt là trong các bài toán xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói. Tuy nhiên, Adagrad cũng có một số hạn chế như không thể xử lý được các feature có giá trị gradient lớn như outliers. Để khắc phục hạn chế này, một số phương pháp khác như RMSProp và Adam đã được đề xuất.

## D. RMSProp

RMSProp là viết tắt của Root Mean Square Propagation, là một phương pháp tối ưu gradient descent khắc phục các nhược điểm của Adagrad. RMSProp sử dụng đạo hàm của hàm mục tiêu như là thông tin đầu vào để cập nhật tỷ lệ học tập cho mỗi tham số của mô hình. RMSProp được thiết kế để giảm độ lớn của đạo hàm khi nó lớn và tăng độ lớn của đạo hàm khi nó nhỏ.

Cụ thể, công thức cập nhật tham số của RMSProp được tính như sau:

[](https://www.codecogs.com/eqnedit.php?latex=g_%7Bt%2B1%7D%20%3D%20%5Calpha%20g_t%20%2B%20(1-%5Calpha)%20%5Cnabla%20L(%5CTheta)%5E2#0)

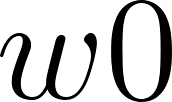
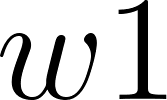
[](https://www.codecogs.com/eqnedit.php?latex=%5CTheta_%7Bt%2B1%7D%20%3D%20%5CTheta_t%20-%20%5Ceta%5Cfrac%7B%5Cnabla%20L%7D%7B%5Csqrt%7Bg_%7Bt%2B1%7D%7D%20%2B%20%5Cepsilon%20%7D#0)

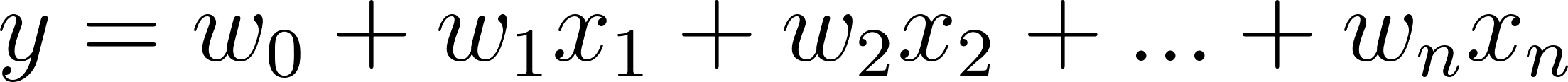
RMSProp là một phương pháp tối ưu gradient descent được sử dụng phổ biến trong bài toán hồi quy tuyến tính. Với cách cập nhật tỷ lệ học tập dựa trên giá trị độ lớn trung bình của đạo hàm bình phương của các bước cập nhật trước đó, RMSProp giúp giảm độ chậm hội tụ và tránh mắc phải vấn đề về local minimum. Việc kết hợp RMSProp với momentum có thể giúp tăng tốc độ hội tụ và cải thiện hiệu quả của quá trình tối ưu.

# **V. Ứng dụng của thuật toán Gradient Descent**

## A. Ứng dụng trong Machine Learning

Trong thực tế, thuật toán Gradient Descent được sử dụng rất phổ biến trong các bài toán Machine Learning liên quan đến hồi quy và phân loại. Một trong những ứng dụng phổ biến của Gradient Descent là trong bài toán dự đoán giá nhà. Với việc thu thập dữ liệu về giá nhà và các thông tin liên quan như số lượng phòng, diện tích, vị trí, tiện nghi, thời gian xây dựng và nhiều yếu tố khác, chúng ta có thể xây dựng một mô hình hồi quy tuyến tính để dự đoán giá nhà. Để xây dựng mô hình này, ta cần tìm các tham số phù hợp để giảm thiểu sai số giữa giá nhà dự đoán và giá nhà thực tế. Thuật toán Gradient Descent được sử dụng để tối ưu hóa các tham số này.

Cụ thể, ta có thể sử dụng thuật toán Gradient Descent để tìm ra các tham số [](https://www.codecogs.com/eqnedit.php?latex=w0#0), [](https://www.codecogs.com/eqnedit.php?latex=w1#0), [](https://www.codecogs.com/eqnedit.php?latex=w2#0),..., [](https://www.codecogs.com/eqnedit.php?latex=wm#0) trong mô hình hồi quy tuyến tính:

[](https://www.codecogs.com/eqnedit.php?latex=y%20%3D%20w_0%20%2B%20w_1x_1%20%2B%20w_2x_2%20%2B%20...%20%2B%20w_nx_n#0)

Với các tham số được tối ưu hóa, ta có thể dự đoán giá nhà dựa trên các thông tin đầu vào của căn nhà.

Không chỉ góp phần giải quyết các bài toán hồi quy tuyến tính, Gradient Descent còn là một công cụ hữu hiệu trong việc xử lý các mô hình học máy không giám sát (Unsupervised Learning), phát hiện cụm trong dữ liệu không được gán nhãn. Trong phân tích cụm, thuật toán gradient descent có thể tối ưu hóa hàm mục tiêu để tìm ra các cụm dữ liệu tương đồng với nhau. Ngoài ra, những mảng khác của học máy như học tăng cường (Reinforcement Learning), mạng nơ-ron nhân tạo cũng cần sự trợ giúp của Gradient Descent.

Trong học tăng cường, chính sách là một phương thức để quyết định cách hành động trong một môi trường, nơi mà chúng ta cần tối đa hóa phần thưởng trung bình thu được từ các hành động đó. Để tối ưu hóa chính sách, chúng ta có thể sử dụng phương pháp gradient descent. Cụ thể, chúng ta có thể sử dụng Gradient Descent để tìm ra giá trị tối ưu của hàm giá trị hành động (Q-function), cho trước chính sách hiện tại. Q-function là một hàm số có thể tính toán giá trị của một hành động trong một trạng thái cụ thể, với giả định rằng chính sách đang được sử dụng. Để áp dụng Gradient Descent vào tối ưu hóa Q-function, chúng ta cần tính toán gradient của hàm này đối với các tham số của chính sách hiện tại. Sau đó, chúng ta có thể cập nhật các tham số bằng cách di chuyển một khoảng nhỏ theo phương ngược với gradient, để tìm ra giá trị tối ưu của Q-function.

Ngoài ra, một ứng dụng quan trọng khác của gradient descent trong machine learning là trong mô hình neural network. Neural network là một loại mô hình học máy được xây dựng bằng cách kết hợp nhiều lớp các neuron và các liên kết giữa chúng. Mỗi neuron nhận đầu vào từ các neuron của lớp trước đó, tính toán và truyền đầu ra đến các neuron của lớp sau. Các liên kết giữa các neuron có trọng số, và quá trình học của mạng neural chính là điều chỉnh các trọng số này để mô hình có thể dự đoán đầu ra chính xác hơn. Gradient descent là phương pháp tối ưu hóa trọng số trong mạng neural. Để tối thiểu hóa hàm mất mát (loss function) trong quá trình huấn luyện mạng neural, gradient descent sẽ tính gradient của hàm mất mát theo các trọng số và điều chỉnh chúng theo chiều ngược lại của gradient. Quá trình này được gọi là backpropagation và được thực hiện trong mỗi vòng lặp huấn luyện.

## B. Ứng dụng trong Computer Vision

Trong lĩnh vực Computer Vision, Gradient Descent được sử dụng rộng rãi trong các bài toán xử lý ảnh, nhất là trong việc tối ưu hóa các mô hình deep learning. Những mô hình này thường có hàng triệu tham số và việc tìm ra bộ tham số tối ưu để đạt được kết quả tốt nhất là vô cùng phức tạp. Trong đó, việc tối ưu hóa các tham số của mô hình dựa trên Gradient Descent là một phương pháp hiệu quả.

Một trong những ứng dụng chính của Gradient Descent trong computer vision là tìm kiếm hình dạng. Trong quá trình này, Gradient Descent được sử dụng để tối ưu hóa một hàm mục tiêu, nhằm điều chỉnh các tham số của một mô hình, giúp chúng ta tìm ra hình dạng tốt nhất của một đối tượng trong hình ảnh. Cụ thể có thể xét ví dụ bài toán nhận diện khuôn mặt. Trong bài toán này, mục tiêu cần tìm kiếm khuôn mặt của một người trong một bức ảnh. Để giải quyết vấn đề này, Gradient Descent được sử dụng để điều chỉnh các tham số của một mô hình, giúp tìm kiếm vị trí và kích thước của khuôn mặt trong hình ảnh. Thuật toán Gradient Descent giúp mô hình đạt được kết quả chính xác hơn và giảm thiểu sai số trong quá trình tìm kiếm.

Ngoài ra, Gradient Descent còn được sử dụng để giải quyết bài toán phân tích hình ảnh, giúp xác định các đặc trưng của một hình ảnh. Trong quá trình này, Gradient Descent được sử dụng để tối ưu hóa các tham số của một mô hình, giúp phát hiện và phân tích các đặc trưng của một hình ảnh, bao gồm cạnh, điểm nổi bật, hình dạng và màu sắc.

Có rất nhiều biến thể của gradient descent được sử dụng trong mô hình neural network, chẳng hạn như stochastic gradient descent (SGD), mini-batch gradient descent, momentum gradient descent, hay Adam. Mỗi biến thể có đặc điểm riêng, tùy vào bài toán cụ thể mà sử dụng phương pháp nào cho hiệu quả nhất. Trên thực tế, gradient descent và các biến thể của nó được sử dụng rộng rãi trong các ứng dụng computer vision, từ xử lý ảnh, nhận dạng vật thể, đến phân loại và nhận diện giọng nói. Các mô hình deep learning hiện đại, chẳng hạn như Convolutional Neural Network (CNN) hay Recurrent Neural Network (RNN), đều sử dụng gradient descent để tối ưu hóa trọng số.

## C. Ứng dụng trong Natural Language Processing

Natural Language Processing (NLP) là một lĩnh vực rất quan trọng trong máy học và trí tuệ nhân tạo. NLP là sự kết hợp giữa ngôn ngữ tự nhiên và các thuật toán máy học để giúp máy tính có thể hiểu và sản xuất ra ngôn ngữ nhân tạo. Trong quá trình huấn luyện mô hình NLP, gradient descent là một trong những thuật toán quan trọng được sử dụng để tối ưu hóa hàm mất mát và tìm ra các trọng số tối ưu cho mô hình.

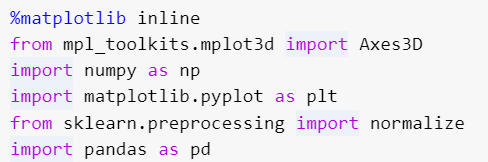
Gradient descent là một phương pháp tối ưu hóa được sử dụng để cập nhật các trọng số trong quá trình huấn luyện mô hình NLP. Phương pháp này là cách tìm kiếm trọng số tối ưu nhất bằng cách di chuyển theo hướng giảm dần của đạo hàm của hàm mất mát. Điều này đảm bảo rằng mô hình NLP được huấn luyện để dự đoán kết quả chính xác nhất. Các phương pháp tối ưu hóa khác như Stochastic Gradient Descent (SGD) hay Mini-Batch Gradient Descent cũng được sử dụng trong NLP để cập nhật trọng số và tìm kiếm nhanh hơn.

Các ứng dụng của gradient descent trong NLP rất đa dạng. Một trong số đó là xây dựng các mô hình dự đoán văn bản. Để huấn luyện các mô hình dự đoán văn bản, gradient descent được sử dụng để tối ưu hóa các tham số mô hình, bao gồm trọng số và các hệ số. Điều này giúp cho mô hình đạt được độ chính xác cao nhất có thể và dự đoán được các kết quả tốt nhất. Một ứng dụng NLP khác của gradient descent là xây dựng mô hình phân loại văn bản. Việc phân loại văn bản là một trong những thách thức lớn nhất của NLP và được áp dụng rộng rãi trong các lĩnh vực như xã hội học, kinh tế học và chính trị học. Để giải quyết vấn đề này, gradient descent được sử dụng để tối ưu hóa các tham số của mô hình, bao gồm trọng số và các hệ số.

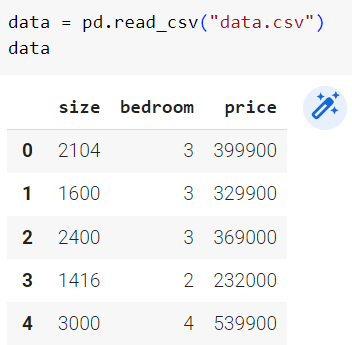
# **VI. Triển khai thuật toán Gradient Descent bằng Python**

## A. Cài đặt thuật toán Gradient Descent với Python

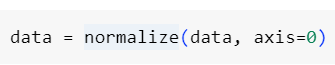
Khai báo các thư viện cần dùng



Tải dataframe từ file csv



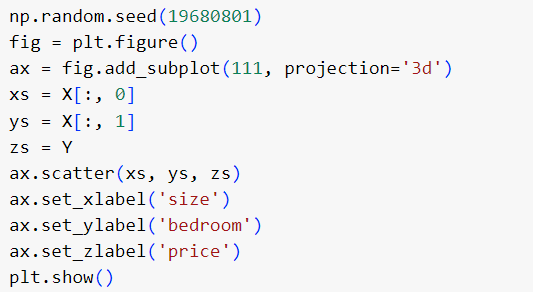
Chuẩn hóa dữ liệu

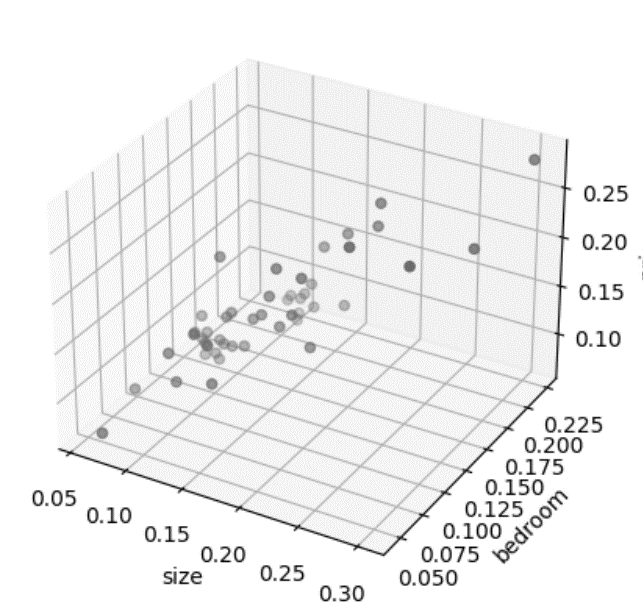


Tách dữ liệu thành X và Y



Trực quan hóa dữ liệu

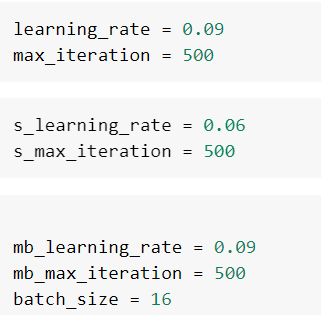




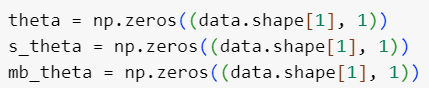
Tìm kiếm giá trị tối ưu của một hàm mục tiêu sử dụng 3 phương pháp Batch Gradient Descent, Mini-Batch Gradient Descent và Stochastic Gradient Descent (SGD):

* Batch Gradient Descent (BGD):
* BGD là phương pháp tối ưu hóa trong đó các tham số mô hình được cập nhật sau khi tính toán gradient của hàm mục tiêu trên toàn bộ tập dữ liệu huấn luyện.
* Điều này có nghĩa là BGD tính toán gradient cho tất cả các điểm dữ liệu trong một lượt (batch) và sau đó cập nhật các tham số.
* BGD đảm bảo hướng đi của cập nhật tham số được xác định dựa trên thông tin toàn cục của dữ liệu, tạo ra kết quả ổn định và hội tụ tới điểm tối ưu.
* Mini-Batch Gradient Descent:
* Mini-Batch Gradient Descent (MBGD) là biến thể trung gian giữa BGD và SGD.
* Trong MBGD, tập dữ liệu huấn luyện được chia thành các mini-batch nhỏ. Mỗi mini-batch gồm một số lượng nhỏ các điểm dữ liệu.
* Gradient được tính toán trên mỗi mini-batch và các tham số được cập nhật sau khi tính toán gradient trên mỗi mini-batch.
* MBGD kết hợp hiệu quả tính toán của BGD và khả năng tổng quát hóa của SGD. Nó thường được sử dụng để huấn luyện các mô hình trên tập dữ liệu lớn hơn so với SGD, nhưng vẫn tiết kiệm tài nguyên tính toán so với BGD.
* Stochastic Gradient Descent (SGD):
* Stochastic Gradient Descent là phương pháp tối ưu hóa trong đó các tham số mô hình được cập nhật sau khi tính toán gradient của hàm mục tiêu trên từng điểm dữ liệu riêng lẻ.
* SGD tính toán gradient cho một điểm dữ liệu duy nhất và sau đó cập nhật các tham số. Quá trình này được lặp lại cho tất cả các điểm dữ liệu trong quá trình huấn luyện.
* SGD có thể cho kết quả nhanh chóng và hiệu quả tính toán cho các tập dữ liệu lớn, nhưng có thể làm dao động và không ổn định hơn so với BGD và MBGD.

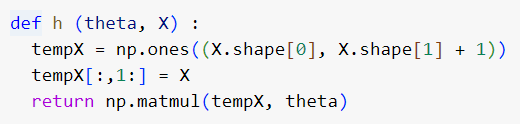
Thiết lập các tham số:



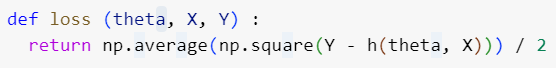
Parameters



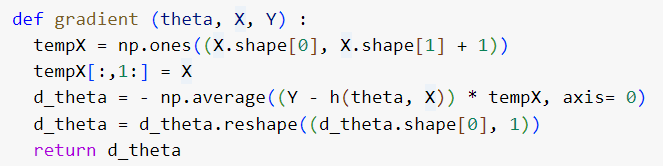
Hypothesis



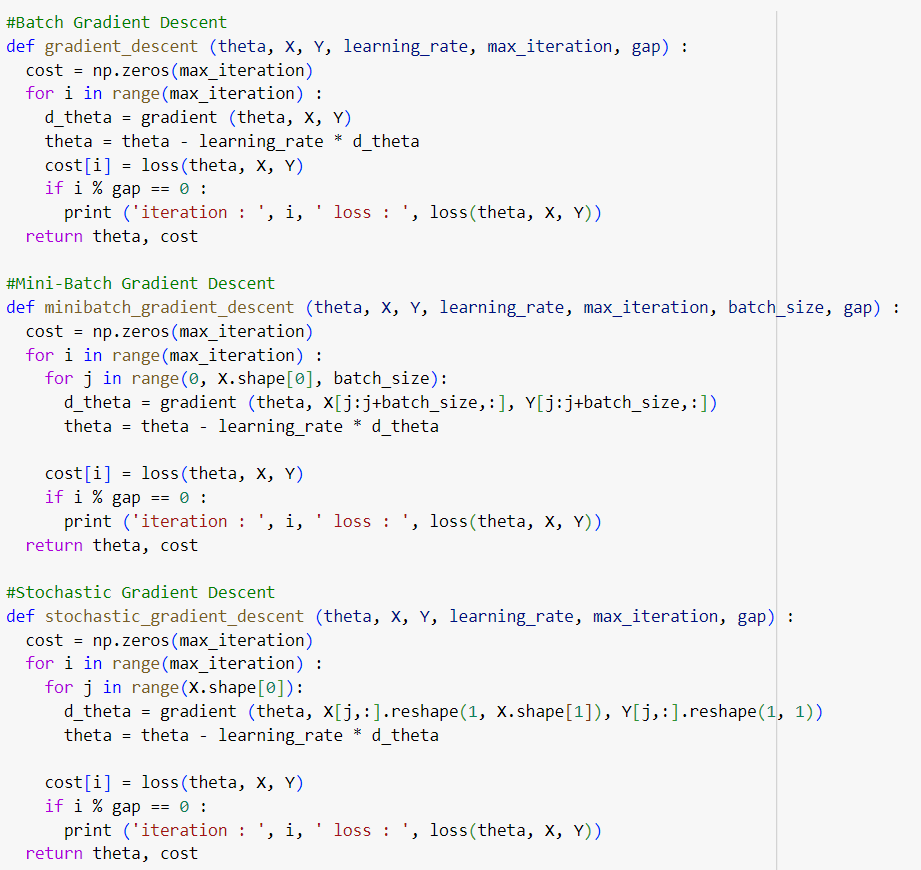
Tính Loss Function



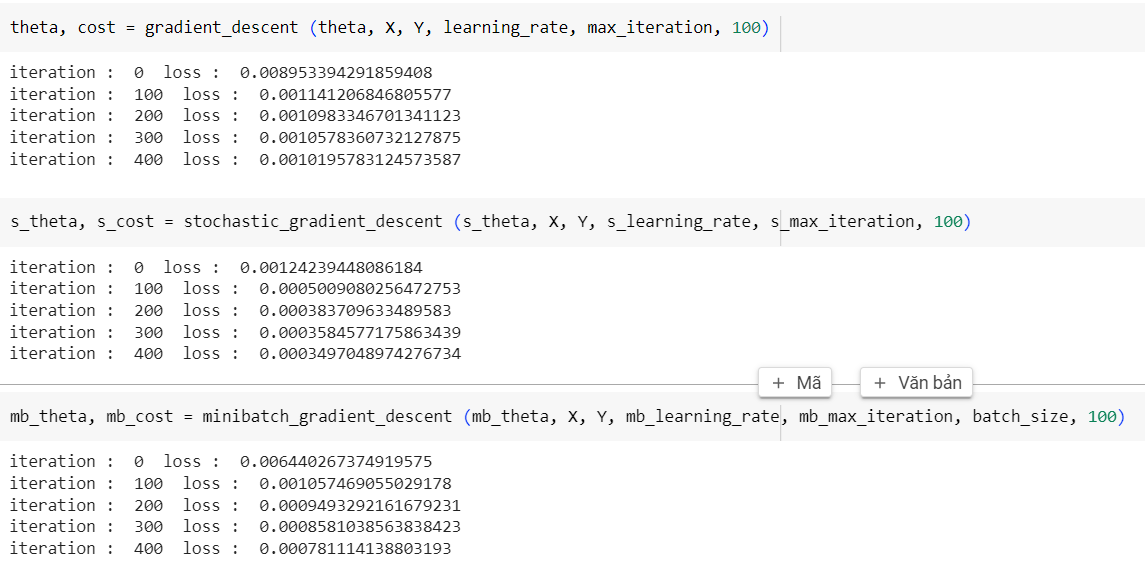
Gradient Descent



Chạy Gradient Desent bằng cả 3 phương pháp



Train Model

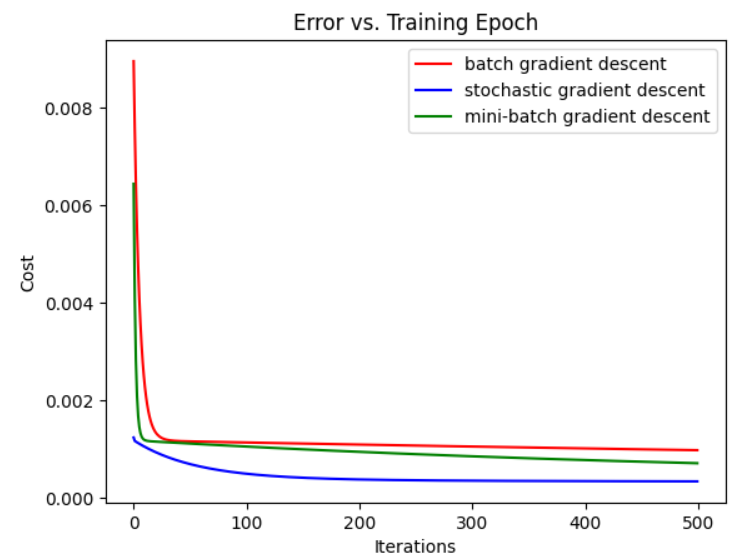


Giá trị tối ưu của từng mô hình

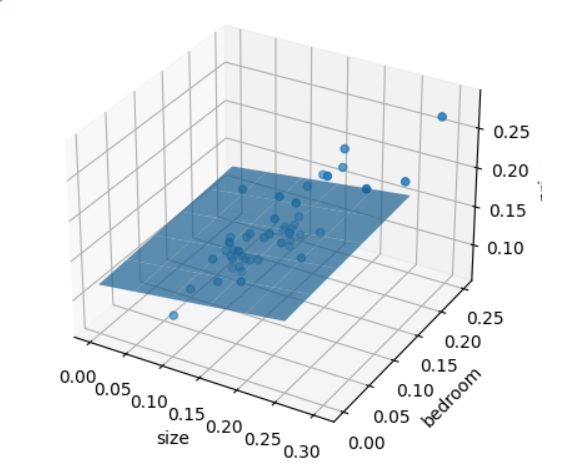


Đồ thị hàm cost

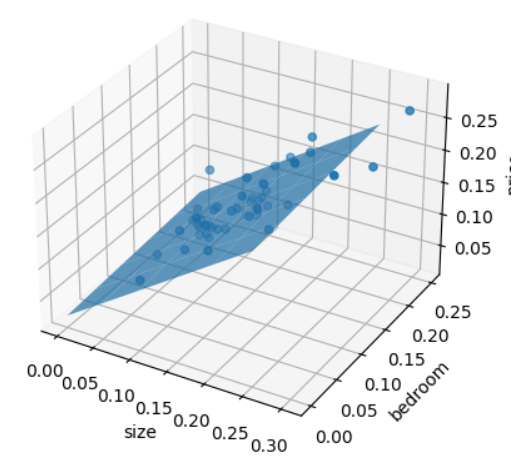




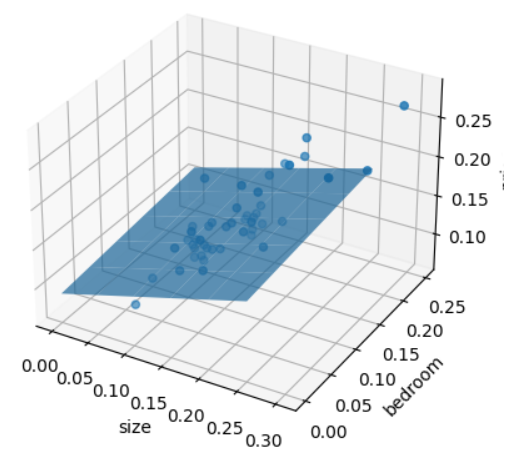
Kết quả



Batch Gradient Descent



Mini-Batch Gradient Descent



Stochastic Gradient Descent

# **VII. Kết luận**

## A. Tóm tắt các kết quả của báo cáo

Báo cáo đã tập trung trình bày về thuật toán Gradient Descent và ứng dụng trong bài toán hồi quy tuyến tính. Sau phần giới thiệu tổng quan về bài toán hồi quy tuyến tính và thuật toán Gradient Descent, báo cáo đi sâu vào cơ bản và phân loại của thuật toán Gradient Descent, cách hoạt động của nó và ứng dụng của nó cho bài toán hồi quy tuyến tính. Ngoài ra, báo cáo cũng trình bày các phương pháp cải tiến của thuật toán Gradient Descent như Momentum, Nesterov Accelerated Gradient, Adagrad và RMSProp. Đồng thời giới thiệu các bước để triển khai thuật toán Gradient Descent bằng Python, kết hợp thư viện Scikit-learn.

Dựa trên những nội dung trình bày trong báo cáo, có thể tổng kết được một số kết quả như sau:

* Gradient Descent là một thuật toán quan trọng và được sử dụng rộng rãi trong nhiều lĩnh vực ứng dụng, đặc biệt là trong Machine Learning. Thuật toán cho phép tối ưu hóa hàm mất mát và tìm ra bộ tham số tốt nhất cho mô hình hồi quy tuyến tính.
* Gradient Descent có hai phương pháp cập nhật tham số là Batch Gradient Descent và Stochastic Gradient Descent, cùng với một số phương pháp cải tiến như Momentum, Nesterov Accelerated Gradient, Adagrad và RMSProp.
* Các phương pháp cải tiến giúp cho Gradient Descent hoạt động nhanh hơn, tránh được tình trạng bị rơi vào cực tiểu địa phương và có thể giúp tăng độ chính xác của mô hình hồi quy tuyến tính.
* Thuật toán Gradient Descent có thể được triển khai bằng nhiều ngôn ngữ lập trình khác nhau, trong đó Python là một trong những ngôn ngữ phổ biến nhất. Chúng ta có thể sử dụng thư viện Scikit-learn để cài đặt Gradient Descent một cách đơn giản và nhanh chóng.
* Thuật toán Gradient Descent đóng vai trò rất quan trọng trong Machine Learning và các lĩnh vực ứng dụng khác, và còn có nhiều tiềm năng để phát triển và tối ưu hơn trong tương lai.

## B. Những hạn chế và hướng phát triển cho thuật toán Gradient Descent và các phương pháp cải tiến

Mặc dù Gradient Descent là một thuật toán rất mạnh và hiệu quả, nhưng nó cũng tồn tại một số hạn chế. Một trong những hạn chế của thuật toán này là khả năng rơi vào điểm cực tiểu địa phương (local minimum), điều này có thể khiến cho kết quả của thuật toán không tốt. Để giải quyết vấn đề này, người ta đã phát triển nhiều phương pháp cải tiến cho thuật toán Gradient Descent. Một trong những phương pháp cải tiến phổ biến nhất là Momentum đã được giới thiệu trong phần IV của báo cáo. Momentum giúp tăng tốc độ của quá trình hội tụ và giảm khả năng rơi vào điểm cực tiểu địa phương. Ngoài ra, các phương pháp cải tiến khác như Nesterov Accelerated Gradient, Adagrad và RMSProp cũng đã được phát triển và được sử dụng rộng rãi trong các mô hình học máy.

Ngoài những phương pháp cải tiến hiện tại, vẫn còn nhiều hướng phát triển cho thuật toán Gradient Descent. Một trong số đó là sử dụng các kỹ thuật học tăng cường để cải thiện hiệu suất của thuật toán, hoặc phát triển các phương pháp tối ưu hóa đa nhiệm cho việc giải quyết các bài toán phức tạp hơn. Ngoài ra, việc kết hợp các thuật toán tối ưu hóa khác như Quasi-Newton hoặc Conjugate Gradient cũng có thể cải thiện hiệu suất của thuật toán Gradient Descent.

Tổng quan lại, Gradient Descent là một trong những thuật toán quan trọng nhất trong Machine Learning và các lĩnh vực ứng dụng khác. Nó cung cấp cho chúng ta một công cụ mạnh mẽ để tối ưu hóa các mô hình học máy. Tuy nhiên, để sử dụng thuật toán này một cách hiệu quả, chúng ta cần hiểu rõ về cách hoạt động của nó và các phương pháp cải tiến hiện có để giải quyết các vấn đề liên quan đến thuật toán.

## C. Tầm quan trọng của thuật toán Gradient Descent trong Machine Learning và các lĩnh vực ứng dụng khác.

Thuật toán Gradient Descent là một trong những thuật toán quan trọng nhất trong Machine Learning và được sử dụng rộng rãi trong các lĩnh vực ứng dụng khác. Nó cung cấp cho chúng ta một công cụ mạnh mẽ để tìm kiếm giá trị tối ưu của hàm mất mát và học các mô hình phức tạp dựa trên các tập dữ liệu lớn.

Một trong những ứng dụng chính của Gradient Descent là trong bài toán hồi quy tuyến tính, nơi nó được sử dụng để tìm nghiệm tối ưu cho các mô hình hồi quy tuyến tính. Ngoài ra, Gradient Descent cũng được sử dụng rộng rãi trong các bài toán phân loại, gom nhóm và các mô hình Deep Learning phức tạp.

Bên cạnh đó, thuật toán Gradient Descent cũng được áp dụng trong nhiều lĩnh vực ứng dụng khác, chẳng hạn như trong các ứng dụng tối ưu hóa và tối ưu mô hình trong các lĩnh vực kỹ thuật, tài chính và kinh doanh. Thuật toán này cũng được sử dụng trong xử lý ảnh, xử lý ngôn ngữ tự nhiên và các ứng dụng khác trong lĩnh vực trí tuệ nhân tạo.

Tóm lại, thuật toán Gradient Descent đóng vai trò rất quan trọng trong việc xây dựng các mô hình Machine Learning và trong nhiều lĩnh vực ứng dụng khác. Việc hiểu rõ về thuật toán này và các phương pháp cải tiến của nó sẽ giúp cho chúng ta có được những mô hình chính xác và hiệu quả hơn, đồng thời giải quyết được các vấn đề thực tế trong nhiều lĩnh vực khác nhau.

# **VIII. Tài liệu tham khảo**

1. Vu, T. (2017) *Bài 7: Gradient Descent (phần 1/2)*, *Tiep Vu’s blog*. Available at: https://machinelearningcoban.com/2017/01/12/gradientdescent/ (Accessed: 10 May 2023).
2. Ruder, S., 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.
3. Botev, A., Lever, G. and Barber, D., 2017, May. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In 2017 International joint conference on neural networks (IJCNN) (pp. 1899-1903). IEEE.
4. Trực, T.T. (2023) Optimizer- Hiểu Sâu về Các Thuật Toán Tối ưu (GD, SGD, adam,..), Viblo. Available at: https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8 (Accessed: 10 May 2023).
5. Hinton, G., Srivastava, N. and Swersky, K., 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on, 14(8), p.2.
6. Van, K.N. (2021) Ai Club Tutorials. Available at: http://tutorials.aiclub.cs.uit.edu.vn/index.php/2020/06/07/machine-learning-gradient-descent-la-gi-phan-1-5/ (Accessed: 10 May 2023).
7. Ruder, S. (2020) An overview of gradient descent optimization algorithms, ruder.io. Available at: https://www.ruder.io/optimizing-gradient-descent/ (Accessed: 10 May 2023).