**Yêu cầu**

Viết cơ sở toán học, mã giã và cài đặt 1 trong các thuật toán sau (về giải phương trình hồi quy phi tuyến):

- **Phương pháp Gradient Descent (Tối ưu hóa dựa trên đạo hàm)**

- Phương pháp Gauss-Newton

- Phương pháp Spline hoặc Spline Cubic

- Phương pháp hồi quy quá trình Gaussian (Gaussian Process)

- Phương pháp tối ưu hóa tổng quát (Global Optimization Techniques)

- Nhóm thuật toán AI: Neural Networks, SVM, Mô hình cây quyết định và các phương pháp kết hợp (Ensemble Methods)

**Mục Lục**

[1. Cơ sở toán học của thuật toán Gradient Descent 2](#_Toc195890079)

[1.1. Tối ưu phi tuyến 2](#_Toc195890080)

[1.2. Phương pháp Gradient Descent 3](#_Toc195890081)

[2. Mã giả 4](#_Toc195890082)

[3. Cài đặt thuật toán trên python 4](#_Toc195890083)

[3.1. Chọn tập dữ liệu về Việt Nam 4](#_Toc195890084)

[3.2. Cài đặt thuật toán với python 5](#_Toc195890085)

[4. So sánh với ChatGPT 6](#_Toc195890086)

[4.1. So sánh cài đặt thuật toán bằng python với ChatGPT 6](#_Toc195890087)

[4.2. Kiểm tra mã giả với ChatGPT 7](#_Toc195890088)

[5. Tài liệu tham khảo 8](#_Toc195890089)

**Bảng phân công công việc**

|  |  |  |
| --- | --- | --- |
| **Thành viên** | | **Công việc phân công** |
| **Mã học viên** | **Họ và Tên** |
| 240104026 | Huỳnh Quốc Bữu | * Cơ sở toán học * Mã giả |
| 240104039 | Nguyễn Đức Linh | * Cơ sở toán học * Cài đặt thuật toán với python và viết so sánh với ChatGPT |

# Cơ sở toán học của thuật toán Gradient Descent

## Tối ưu phi tuyến

Giả sử có mô hình phi tuyến:

(1.1.1)

Trong nhiều bài toán học máy, đặc biệt là hồi quy phi tuyến (nonlinear regression), mục tiêu là tìm bộ tham số tối ưu để mô hình , khớp tốt nhất với dữ liệu huấn luyện . Điều này thường được thực hiện bằng cách tối thiểu hóa một hàm mất mát, chẳng hạn như tổng bình phương sai số [1]:

(1.1.2)

Bài toán lúc này là cần tối ưu với điều kiện tổng quát là: tìm sao cho nhỏ nhất

Với

là đạo hàm cấp 1 của F theo (Ma trận *Jacobian*)

là đạo hàm cấp 2 của F theo (Ma trận *Hessian*)

Điều kiện cần cho điểm cực tiểu cục bộ là:

Điều kiện đủ nếu tại là:

(ma trận **Hessian là ma trận xác định dương**, thì là cực tiểu địa phương duy nhất. [2])

Có nhiều phương pháp tìm tìm . Trong bài tập này Nhóm chúng em chọn phương pháp **Gradient Descent** (Tối ưu hóa dựa trên đạo hàm)

## Phương pháp Gradient Descent

Đa thức Taylor với khai triển bậc 1 và bước lặp của hàm F là:

(1.2.1)

Để với một số cần:

Ta gọi là hướng giảm (descent direction) nếu

Gọi - là lượng giảm của của hàm F khi di chuyển từ đến

Khi nhỏ từ (1.2.1) ta có - =

Tốc độ giảm chuẩn hóa là (1.2.2)

Với là góc giữa 2 vector h và

Nhận thấy tốc độ giảm chuẩn hóa lớn nhất khi

Điều này chứng tỏ là hướng giảm tốt nhất

Gọi . Khi đó là độ dốc lớn nhất tại .

Cũng có nghĩa là giảm nhanh nhất với độ dốc

Khi đó ta tìm bằng cách tìm kiếm với số lần lặp tối đa (bắt đầu = 0) sao cho

Với là sai số kỳ vọng cho trước

và được tìm qua line search sao cho:

# Mã giả

**Input:**

←

for k = 0 to do

g ←

if then

return

end if

h ←

α ← LineSearch(, h)

←

end for

**return**

# Cài đặt thuật toán trên python



A computer screen shot of a black screen

AI-generated content may be incorrect.

## Chọn tập dữ liệu về Việt Nam

1. Chọn tập dữ liệu về giá nhà tại Việt Nam để phân tích
2. Nguồn dữ liệu: <https://www.kaggle.com/datasets/nguyentiennhan/vietnam-housing-dataset-2024/data> [3]
3. File dữ liệu: Github-Code-Team11/datasets/vietnam\_housing\_dataset.csv
4. Giới thiệu tập dữ liệu:

Bộ dữ liệu này chứa thông tin về các bất động sản nhà ở tại Việt Nam. Nó bao gồm các đặc điểm chi tiết của từng bất động sản như vị trí, đặc điểm vật lý, tình trạng pháp lý và nội thất, cùng với giá bán. Bộ dữ liệu này rất hữu ích cho việc phân tích bất động sản, xây dựng mô hình dự đoán giá và phân tích xu hướng thị trường.

Nguồn thu thập dữ liệu: Dữ liệu được thu thập (crawled) từ trang web batdongsan.vn.

1. Mô tả các cột dữ liệu:

* Address (Địa chỉ): Địa chỉ đầy đủ của bất động sản, bao gồm tên dự án (nếu có), đường, phường, quận và thành phố.
* Area (Diện tích): Tổng diện tích của bất động sản, đơn vị tính là mét vuông (m²).
* Frontage (Mặt tiền): Chiều rộng mặt trước của bất động sản, đơn vị tính là mét (m).
* Access Road (Đường vào): Chiều rộng của con đường dẫn vào bất động sản, đơn vị tính là mét (m).
* House Direction (Hướng nhà): Hướng chính mà mặt trước của căn nhà quay về (ví dụ: Đông, Tây, Nam, Bắc).
* Balcony Direction (Hướng ban công): Hướng mà ban công quay về.
* Floors (Số tầng): Tổng số tầng của căn nhà.
* Bedrooms (Số phòng ngủ): Số lượng phòng ngủ trong bất động sản.
* Bathrooms (Số phòng tắm): Số lượng phòng tắm trong bất động sản.
* Legal Status (Tình trạng pháp lý): Trạng thái pháp lý của bất động sản, ví dụ như có sổ hồng, đang trong hợp đồng mua bán, v.v.
* Furniture State (Tình trạng nội thất): Tình trạng trang bị nội thất, ví dụ như đầy đủ nội thất, nội thất cơ bản hoặc chưa có nội thất.
* Price (Giá bán): Giá của bất động sản, đơn vị tính là **tỷ đồng Việt Nam (VND)**.

## Cài đặt thuật toán với python

|  |
| --- |
| import pandas as pd  import numpy as np  # 1) Đọc dữ liệu  CSV\_PATH = "C:\Users\ducli\Downloads\archive (1)/vietnam\_housing\_dataset.csv"  df = pd.read\_csv(CSV\_PATH)  # 2) Giữ lại 3 đặc trưng + đích  features = ["Area", "Bedrooms", "Floors"]  target   = "Price"  data = df[features + [target]].dropna()        # bỏ hàng có NaN  # 3) Chuẩn hoá đặc trưng (z‑score) cho tốc độ hội tụ  X\_raw = data[features].values  y     = data[target].values.reshape(-1, 1)  mu    = X\_raw.mean(axis=0)  sigma = X\_raw.std(axis=0)  Xn    = (X\_raw - mu) / sigma                  # X chuẩn hoá  m     = Xn.shape[0]  # 4) Thêm cột bias  X = np.hstack([np.ones((m, 1)), Xn])          # (m × 4)  theta = np.zeros((X.shape[1], 1))             # khởi tạo 0  # 5) Các hàm tiện ích  def compute\_cost(X, y, theta):      m = len(y)      return (1/(2\*m)) \* np.sum((X @ theta - y) \*\* 2)  def gradient\_descent(X, y, theta, alpha=0.01, iters=1000):      cost\_history = []      m = len(y)      for \_ in range(iters):          error  = X @ theta - y          grad   = (1/m) \* (X.T @ error)          theta -= alpha \* grad          cost\_history.append(compute\_cost(X, y, theta))      return theta, cost\_history  # 6) Huấn luyện  theta\_opt, J\_hist = gradient\_descent(X, y, theta,                                       alpha     = 0.01,                                       iters     = 1000)  # 7) In kết quả  print("Thông số đã học (theta):")  print(f"θ0 (bias)      = {theta\_opt[0,0]:.4f}")  print(f"θ1 (Area)      = {theta\_opt[1,0]:.4f}")  print(f"θ2 (Bedrooms)  = {theta\_opt[2,0]:.4f}")  print(f"θ3 (Floors)    = {theta\_opt[3,0]:.4f}")  print(f"J cuối cùng     = {J\_hist[-1]:.4f}") |

Kết quả thực thi:

A screenshot of a computer program

AI-generated content may be incorrect.

# So sánh với ChatGPT

## So sánh cài đặt thuật toán bằng python với ChatGPT

A computer screen with white text

AI-generated content may be incorrect.

Về cơ bản, cả ChatGPT và nhóm đều tính gradient và cập nhật tham số theo cùng một công thức vectorized; độ phức tạp tính toán giống nhau (O(n·d) mỗi vòng, với n số mẫu, d số chiều).

Điểm khác biệt nhỏ: ChatGPT gọi hàm compute\_loss mỗi vòng; cách này thì chia nhỏ các hàm dễ bảo trì ngược lại nếu xét về tối ưu trong tính toán thì cần gọi đến hàm tính toán, sẽ phí ít thời gian so với code inline.

## Kiểm tra mã giả với ChatGPT

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | C. Bishop, Pattern Recognition and Machine Learning, springer. |
| [2] | S. P. B. a. L. Vandenberghe, Convex optimization. |
| [3] | N. T. Nhan, “Kaggle,” [Trực tuyến]. Available: https://www.kaggle.com/datasets/nguyentiennhan/vietnam-housing-dataset-2024/data. |