TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ MÔN**

**PHÂN TÍCH XÁC SUẤT VÀ GIẢI THUẬT NGẪU NHIÊN**

**APPLICATION OF**

**GAUSSIAN PROCESS REGRESSION IN TENSORFLOW PROBABILITY**

*Người hướng dẫn*: **TS. Nguyễn Chí Thiện**

*Người thực hiện:* **Phạm Dương Thành Long**

*MSSV :* **51603190**

*Khoá :* **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ MÔN**

**PHÂN TÍCH XÁC SUẤT VÀ GIẢI THUẬT NGẪU NHIÊN**

**APPLICATION OF**

**GAUSSIAN PROCESS REGRESSION IN TENSORFLOW PROBABILITY**

*Người hướng dẫn*: **TS. Nguyễn Chí Thiện**

*Người thực hiện:* **Phạm Dương Thành Long**

*MSSV :* **51603190**

*Khoá :* **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

LỜI CẢM ƠN

Đầu tiên tôi xin gửi lời cảm ơn chân thành nhất đến các bạn bè trong lớp và thầy Nguyễn Chí Thiện nói riêng đã tận tình giúp đỡ tôi trong suốt quá trình thực hiện đề tài cuối kì lần này. Chính điều đó đã thực sự góp một phần không nhỏ trong quá trình hoàn thành báo cáo cuối kỳ môn phân tích xác suất và giải thuật ngẫu nhiên lần này. Tuy bài báo cáo cuối kỳ lần này vẫn chưa được xem là bài báo phân tích và ứng dụng đầy đủ các mặt của lý thuyết và cả bài toán mà chỉ dừng ở mức tổng quan về công nghệ và bài cuối kỳ lần này tôi tập chung nhiều phần kỹ thuật hơn là lý thuyết. Nên mong người đọc nếu cần tìm hiểu thêm về các công nghệ hoặc lý thuyết được đề cập trong bài có thể tham khảo thêm ở phần tham khảo được trích dẫn. Ngoài ra, các đánh giá một cách chân thành từ bạn đọc dù ít hay nhiều đều sẽ được tôi tiếp thu và rút kinh nghiệm cho tất cả các bài báo cáo về sau, một lần nữa xin cảm ơn.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của TS. Nguyễn Chí Thiện. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Phạm Dương Thành Long*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Báo cáo cuối kì lần này sẽ tập trung vào hai nội dung chính đó là tổng quan và đi sâu vào phân tích các công thức toán học trong nội dung lý thuyết của thuật toán. Phần tiếp theo là ứng dụng mô hình Gaussian Process Regression (GPR) vào giải quyết bài toán trên tập dữ liệu thực tế.

Về mặt lý thuyết, đầu tiên sẽ bàn sâu chi tiết về Gaussian Process là gì? Sẽ đi từ công thức tổng quát toán học, sau đó sẽ giải thích các khái niệm đã trình bày bên trên. Phần tiếp theo của lý thuyết sẽ đi sâu vào việc áp dụng Gaussian Process trong bài toán regression. Ngoài ra, phần cuối cũng sẽ nói về việc áp dụng Gaussian Process regression trong thư viện Tensorflow Probability được thực hiện như nào.

Phần ứng dụng mô hình GPR sẽ được xây dựng dựa trên bộ dữ liệu là nồng độ CO2 từ năm 1958-2021 của tổ chức khí tượng NOAA tại Hawaii. Sử dụng thư viện Tensorflow Probability theo Python để mô tả lại quá trình huấn luyện và dự đoán bằng từng bước theo trình tự từ Tiền xử lý dữ liệu, xây dựng các hàm kernel thủ công. Đặc biệt về hàm Covariance kernel sẽ sử dụng kết hợp tổng từ cả 3 kernel con để có thể kết hợp toàn bộ kiến thứ mà hàm kernel có thể học vào trong cùng 1 hàm. Phần huấn luyện và dự đoán cũng sẽ được mô hình hóa kèm một số nhận xét có thể giúp người đọc dễ hiểu hơn trong quá trình tìm hiều bài báo cáo này.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc87091935)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc87091936)

[TÓM TẮT iv](#_Toc87091937)

[MỤC LỤC 2](#_Toc87091938)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc87091939)

[CHƯƠNG 1: CÁC KHÁI NIỆM 5](#_Toc87091940)

[1.1 Gaussian Process: 5](#_Toc87091941)

[1.1.1 Các khái niệm bản 5](#_Toc87091942)

[1.1.2 Tổng quát hóa công thức Gaussian process: 6](#_Toc87091943)

[1.2 Tổng quan về các hàm Kernel: 6](#_Toc87091944)

[1.2.1 Thế nào là một hàm kernel hợp lệ: 6](#_Toc87091945)

[1.2.2 White noise kernel: 7](#_Toc87091946)

[1.2.3 Exponentiated quadratic kernel: 7](#_Toc87091947)

[1.2.4 Cách lựa chọn siêu tham số length scale (l) và amplitude (σ) 8](#_Toc87091948)

[1.2.5 Rational quadratic kernel: 9](#_Toc87091949)

[1.2.6 Periodic kernel: 10](#_Toc87091950)

[1.2.7 Local Periodic kernel: 11](#_Toc87091951)

[1.2.8 Các phương pháp kết hợp các hàm kernel: 12](#_Toc87091952)

[1.2.8.1 kết hợp các hàm kernel bằng phương pháp tích (AND): 12](#_Toc87091953)

[1.2.8.1 kết hợp các hàm kernel bằng phương pháp bổ sung (OR): 12](#_Toc87091954)

[1.3 Gaussian Process trong Regression: 12](#_Toc87091955)

[1.3.1 Dự đoán với các quan sát không nhiễu (Noise-free Observations): 12](#_Toc87091956)

[1.3.1.1 Gaussian process prior 12](#_Toc87091957)

[1.3.1.2 Gaussian process posterior: 14](#_Toc87091958)

[1.3.2 Dự đoán với các quan sát có nhiễu: 14](#_Toc87091959)

[1.3.2.1 Gaussian process prior: 15](#_Toc87091960)

[1.3.2.2 Gaussian process posterior: 15](#_Toc87091961)

[1.3.3 Quy trình lấy mẫu dự đoán dựa trên các phân phối hậu nghiệm của Gaussian Process Regression 15](#_Toc87091962)

[1.3.4 Giải thích về việc Gaussian Process Regression là một hàm phi tham số: 17](#_Toc87091963)

[CHƯƠNG II: XÂY DỰNG MÔ HÌNH GAUSSIAN PROCESS REGRESSION DỰA TRÊN TENSORFLOW PROBABILITY 18](#_Toc87091964)

[2.1 Tổng quan về đề tài: 18](#_Toc87091965)

[2.1.1 Lý do chọn đề tài: 18](#_Toc87091966)

[2.1.2 Tổng quan về dataset: 18](#_Toc87091967)

[2.1.2.1 Giới thiệu tập dữ liệu: 18](#_Toc87091968)

[2.1.2.2 Lý do chọn tập dữ liệu trên: 19](#_Toc87091969)

[2.2 Đọc và xử lý dữ liệu 19](#_Toc87091970)

[2.2.1 Đọc dataset: 19](#_Toc87091971)

[2.2.2 Tiền xử lý và phân tích dữ liệu: 21](#_Toc87091972)

[2.3 Định nghĩa các kernel cần thiết cho Gaussian process regresssion: 23](#_Toc87091973)

[2.3.1 Định nghĩa hàm kernel Mean(): 23](#_Toc87091974)

[2.3.2 Xây dựng kết hợp các hàm covariance kernel: 24](#_Toc87091975)

[2.3.2.1 Định nghĩa chuẩn chung cho các hyper-parameter: 24](#_Toc87091976)

[2.3.2.2 Định nghĩa hyper-parameter cho hàm Exponentiated Quadratic Kernel: 24](#_Toc87091977)

[2.3.2.3 Định nghĩa hyper-parameter cho hàm Rational Quadratic kernel: 24](#_Toc87091978)

[2.3.2.4 Định nghĩa hyper-parameter cho hàm local periodic kernel: 25](#_Toc87091979)

[2.3.2.4 Lập kernel tổng: 26](#_Toc87091980)

[2.4 Xây dựng mô hình: 27](#_Toc87091981)

[2.4.1 Tổng hợp các tham số huấn luyện: 27](#_Toc87091982)

[2.4.2 Định nghĩa hàm loss: 28](#_Toc87091983)

[2.4.3 Tiến hành huấn luyện mô hình: 28](#_Toc87091984)

[2.4.4 Xây dựng hàm dự đoán hậu nghiệm: 30](#_Toc87091985)

[2.5 Nhận xét kết quả thu được: 32](#_Toc87091986)

[TÀI LIỆU THAM KHẢO 33](#_Toc87091987)

[TỰ ĐÁNH GIÁ 33](#_Toc87091988)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

[Hình 1: Đồ thị biểu diễn các mẫu của white noise kernel và ma trận hiệp phương sai của nó. 7](#_Toc87091989)

[Hình 2: Sơ đồ biểu diễn các tham số length scale và amplitude () có trong Exponentiated quadratic kernel 8](#_Toc87091990)

[Hình 3: Sơ đồ biểu diễn sự lan truyền của hàm Exponentiated quadratic kernel với length scale =1 và amptitude =1 8](#_Toc87091991)

[Hình 4: Sơ đồ biểu diễn sự lan truyền của hàm Exponentiated quadratic kernel khi amptitude =1 và giảm length scale xuống 0.3 9](#_Toc87091992)

[Hình 5: Sơ đồ biểu diễn sự lan truyền của hàm Exponentiated quadratic kernel khi length scale =1 và tăng amptitude lên 10 9](#_Toc87091993)

[Hình 6: Sơ đồ biểu diễn sự lan truyền hàm Periodic kernel với amptitude =1, và 3 giá trị tăng dần. 11](#_Toc87091994)

[Hình 7: Sơ đồ biểu diễn sự lan truyền hàm Periodic kernel với amptitude =1, và 3 giá trị tăng giảm. 11](#_Toc87091995)

[Hình 8: Sơ đồ động biểu diễn quá trình cập nhật đường dự đoán của dựa trên hàm phân phối hậu nghiệm theo Gaussian process regression 16](#_Toc87091996)

[Hình 9: Sơ đồ mô phỏng quá trình huấn luyện Gaussian process regression 16](#_Toc87091997)

[Hình 10: Sơ đồ mô thống kê nồng độ khí CO2 trên khí quyển bào các mùa trong năm (2016 - 2022) theo [1] 19](#_Toc87091998)

[Hình 11: Sơ đồ biễu diễn tập dữ liệu từ năm 1958-2021 22](#_Toc87091999)

[Hình 12: Sơ đồ loss của quá trình huấn luyện theo tape.gradient 29](#_Toc87092000)

[Hình 13: Sơ đồ biểu diễn kết quả dự đoán của Gaussian process regression dựa trên dataset đã huấn luyện cho tập test sau năm 2012 32](#_Toc87092001)

CHƯƠNG 1: CÁC KHÁI NIỆM

1.1 Gaussian Process:

1.1.1 Các khái niệm bản

Theo lý thuyết xác suất thống kê, Một Gaussian Process (GP) là một quá trình lấy mẫu ngẫu nhiên (*stochastic process) có đánh số thứ tự index dựa theo thời gian hoặc không gian*. Theo đó, mọi tập hợp con hữu hạn thuộc các biến ngẫu nhiên đều là một phân phối log chuẩn đa biến *(multivariate normal distribution).*

Mỗi Gaussian process là một phân phối đồng thời *(joint distribution)* của tất cả các biến ngẫu nhiên (vô hạn). Giả sử, ta có là một Gaussian process với bất kỳ vector đầu vào nào thuộc sao cho: . Thì lúc này các vector đầu ra là phân phối Gaussian.

Một Gaussian process được xác định bởi 2 hàm kernel:

* sao cho: là trung bình của :

Hàm Covariance kernel sao cho: là hiệp phương sai giữa và .

|  |  |  |
| --- | --- | --- |
| Hoặc |  | (1.2) |
|  | (1.3) |

Trong đó:

* **:** Là Tổng phương sai (hay biên độ lỗi - Amplitude) Có thể được hiểu là độ cao của độ nhiễu tại các vị trí quan sát.
* Là khoảng cách (length scale) giữa 2 vector và

1.1.2 Tổng quát hóa công thức Gaussian process:

Tổng quát lại công thức của Gaussian process ta có 2 trường hợp:

* Trường hợp 1: Công thức tổng quát của Gaussian process cho một quan sát bất kỳ:

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

* *Hay:*

*:* Là phân phối gaussian với meanvà ma trận kích thước :

* Trường hợp 2: Ta có tập hợp con hữu hạn với . Lúc này, phân phối cận biên *(marginal distribution)* được xây dựng dưới dạng một một phân phối Gaussian đa biến ***N****()* :

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

Có thể thấy, trong khi phân phối Gaussian đa biến nhận đầu vào là một số lượng hữu hạn các phân phối đồng thời , thì phân phối Gaussian process đơn biến GP() lại không có giới hạn nào được hiểu như sau:

Giá trị trung bình và hiệp phương sai của *GP()* đều được xác định bởi một hàm *(với đầu vào là một biến tương quan với các biến có cùng miền giá trị).* Do đó, các hàm này có miền đầu vào là vô hạn => *Gaussian process* là một biến ngẫu nhiên gaussian có chiều vô hạn.

1.2 Tổng quan về các hàm Kernel:

1.2.1 Thế nào là một hàm kernel hợp lệ:

Để trở thành một hàm , thì kết quả được trả về phải là một ma trận có giá trị dương. Có nghĩa là, đó phải là một ma trận đối xứng *(symmetric matrix).* Ma trận kernel dương, cũng có nghĩa là các ma trận này khả nghịch. Do vậy, các kernel thường được định nghĩa một điều kiện là luôn dương trước khi được sử dụng để mô hình hóa cho nhiều quá trình khác nhau.

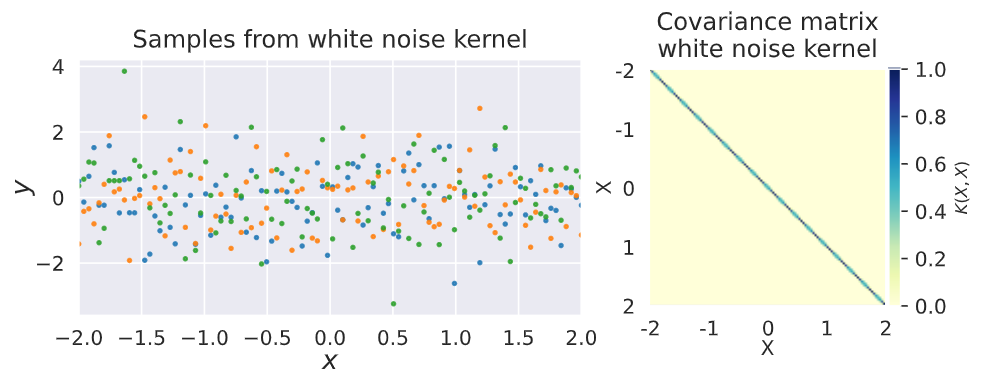
1.2.2 White noise kernel:

đại diện cho độ nhiễu giống hệt nhau và phân bố độc lập được thêm vào phân phối *Gaussian process*, như sau:

|  |  |
| --- | --- |
|  | (1.7) |

Trong đó:

* : Là phương sai của độ nhiễu.
* : Là identity matrix. (là ma trận với đường chéo là 1 và xung quoanh là các số 0).



Hình 1: Đồ thị biểu diễn các mẫu của white noise kernel và ma trận hiệp phương sai của nó.

1.2.3 Exponentiated quadratic kernel:

*Exponentiated quadratic* kernel / *exponential* kernel / *Gaussian* kernel / Kernel lũy thừa bậc 2 đều là tên gọi chỉ kernel trên. Đây là một trong những kernel phổ biến nhất được sử dụng trong việc mô hình hóa Gaussian process. Nó có thể được tính như sau:

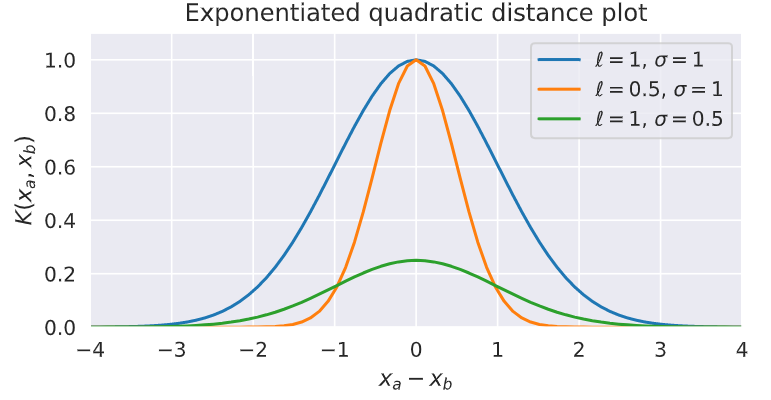
|  |  |
| --- | --- |
|  | (1.8) |

Trong đó:

* **:** Là Tổng phương sai *(hay biên độ lỗi - Amplitude)* có thể được hiểu là độ cao của độ nhiễu tại các vị trí quan sát.
* *(length scale):* Là khoảng cách giữa 2 vector và

Tác dụng của kernel lũy thừa bậc hai là giúp làm mượt *(smooth) các* hàm phân phối tiền nghiện (prior) được lấy mẫu theo Gaussian processs.

Để minh họa cho kernel trên ta có thể lấy ví dụ cho 2 điểm quan sát a và b như hình sau:

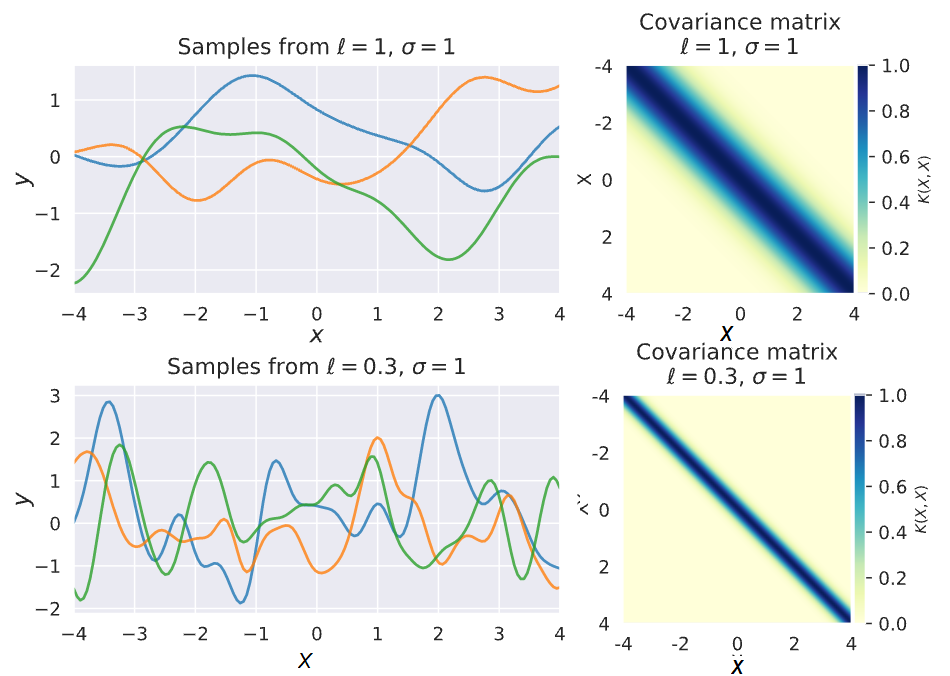


Hình 2: Sơ đồ biểu diễn các tham số length scale và amplitude () có trong Exponentiated quadratic kernel

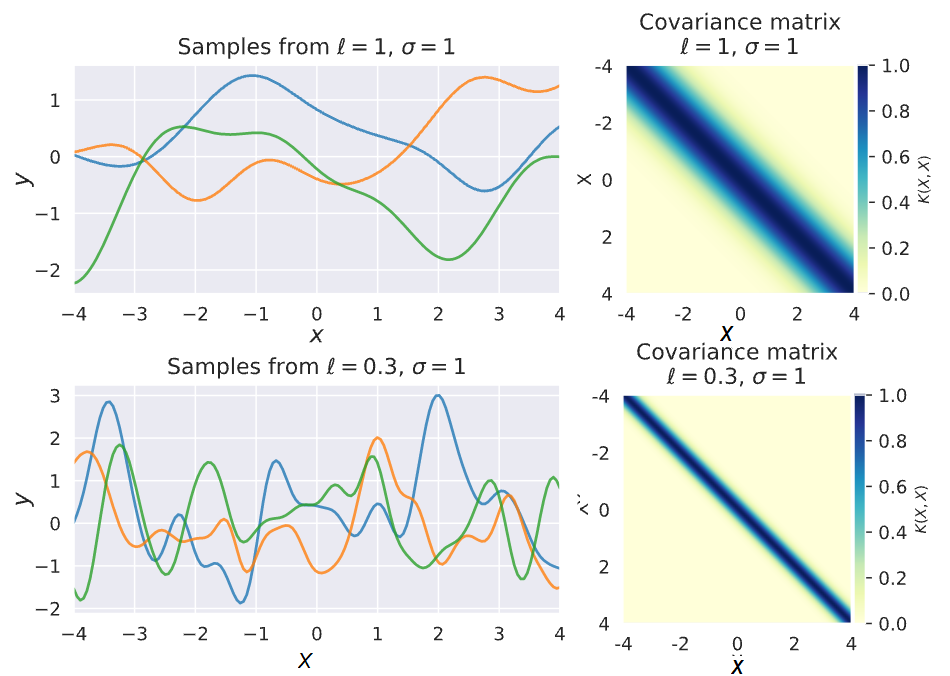
1.2.4 Cách lựa chọn siêu tham số length scale (l) và amplitude (σ)

Khi quan sát hình 2 ta có thể nhận định được rằng:

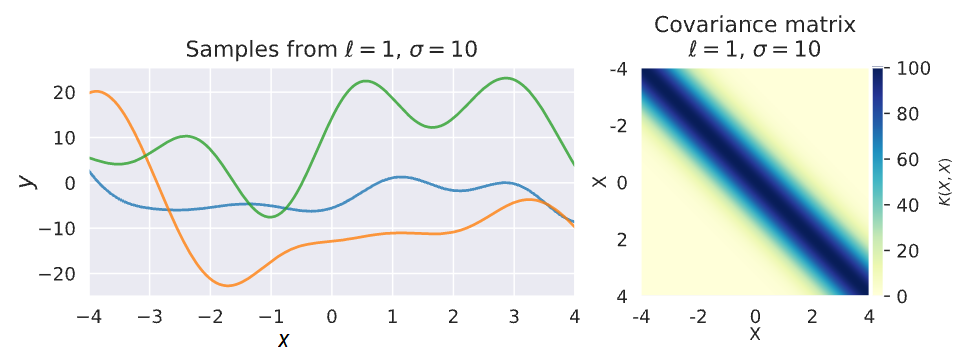
* Kết quả đầu ra từ hàm của *exponential* *kernel* giảm theo cấp số nhân khi quan sát thấy một điểm dữ liệu mới xa trung tâm và độ tương tự là cực đại khi nó nằm gần điểm trung tâm trục X (o).



Hình 3: Sơ đồ biểu diễn sự lan truyền của hàm Exponentiated quadratic kernel với length scale =1 và amptitude =1



Hình 4: Sơ đồ biểu diễn sự lan truyền của hàm Exponentiated quadratic kernel khi amptitude =1 và giảm length scale xuống 0.3



Hình 5: Sơ đồ biểu diễn sự lan truyền của hàm Exponentiated quadratic kernel khi length scale =1 và tăng amptitude lên 10

Thông qua hình ta có thể nhận định một số nội dung sau:

* Khi ***tăng*** giá trị sẽ làm ***tăng sự lan truyền*** của Covariance hay bước sóng sẽ dài hơn.
* Khi ***tăng*** giá trị amplitude sẽ làm ***tăng giá trị lớn nhất*** của của hàm *K()* Covariance.

1.2.5 Rational quadratic kernel:

Ta có công thức của Rational quadratic kernel được tính như sau:

|  |  |
| --- | --- |
|  | (1.9) |

Trong đó:

* + **:** Là Tổng phương sai *(hay biên độ lỗi - Amplitude)* có thể được hiểu là độ cao của độ nhiễu tại các vị trí quan sát.
  + *(length scale):* Là khoảng cách giữa 2 vector và
  + : Là ( > 0)

Tương tự , thì hàm bậc hữu tỉ *(Rational quadratic kernel)* cũng có nhiệm vụ làm mượt các hàm tiền nghiệm trước khi được lấy mẫu từ *Gaussian process*.

Tuy nhiên, *Rational quadratic* có thể được hiểu là một *tổng vô hạn* của các khác nhau sử dụng các khác nhau, trong đó là trọng số giữa các khoảng đó. Khi thì *Rational quadratic kernel sẽ* hội tụ thành .

1.2.6 Periodic kernel:

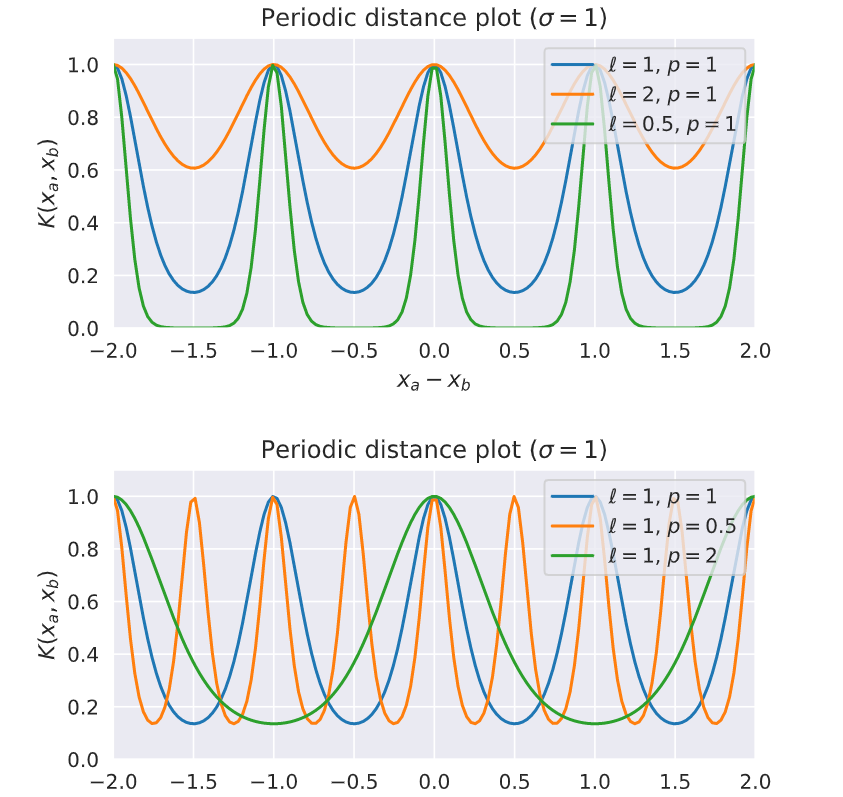
Ta có công thức của Periodic kernel được tính như sau:

|  |  |
| --- | --- |
|  | (1.10) |

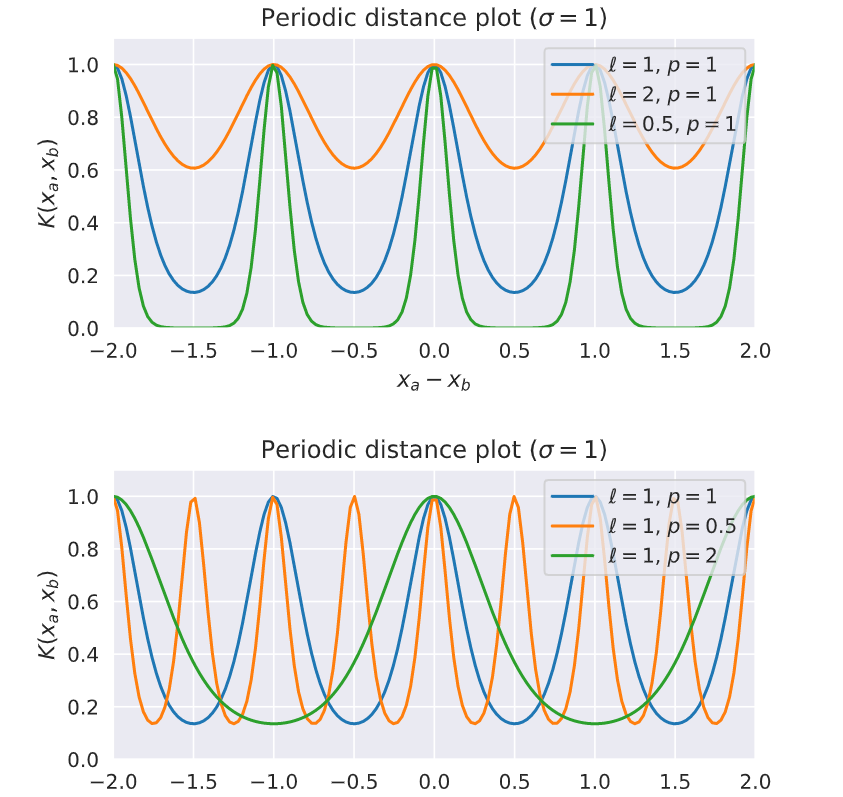
Trong đó:

* + **:** Là Tổng phương sai *(hay biên độ lỗi - Amplitude)* có thể được hiểu là độ cao của độ nhiễu tại các vị trí quan sát.
  + *(length scale):* Là khoảng cách giữa 2 vector và
  + (: Hay chu kỳ là khoảng cách giữa các lần lặp.

Để dễ hình dung hơn về kernel này ta sẽ quan sát 2 biểu đồ hình 5-6 từ quan sát **0** với một quan sát bất kỳ ký hiệu , sử dụng biên độ lỗi cố định = 1, chu kỳ =1 và 3 khoảng cách *length scale* khác nhau như sau:



Hình 6: Sơ đồ biểu diễn sự lan truyền hàm Periodic kernel với amptitude =1, và 3 giá trị tăng dần.



Hình 7: Sơ đồ biểu diễn sự lan truyền hàm Periodic kernel với amptitude =1, và 3 giá trị tăng giảm.

Quan sát *hình 6*, tăng tham số sẽ làm giảm các biến thể cục bộ trong phạm vi lặp lại, tương tự cách tăng trong giúp làm giảm các biến thể *(variations)* trong một phạm vi rộng hơn.

Đối với *hình 7*, tăng chu kỳ sẽ làm tăng khoảng cách giữa các lần lặp lại *(tăng bước sóng)*.

1.2.7 Local Periodic kernel:

Ta có công thức của là tích của 2 hàm và , cho phép các chu kỳ thay đổi trong khoảng cách xa hơn. Lưu ý rằng: các tham số như phương sai sẽ được kết khi tính toán:

|  |  |
| --- | --- |
|  | (1.11) |

Trong đó:

* + **:** Là Tổng phương sai *(hay biên độ lỗi - Amplitude)* có thể được hiểu là độ cao của độ nhiễu tại các vị trí quan sát.
  + Là lengthscale của hàm periodic.
  + (: Hay chu kỳ là khoảng cách giữa các lần lặp.
  + Là lengthscale của hàm *exponentiated quadratic*.

1.2.8 Các phương pháp kết hợp các hàm kernel:

1.2.8.1 kết hợp các hàm kernel bằng phương pháp tích (AND):

Các hàm có thể được dễ dàng kết hợp với nhau bằng cách nhân chúng lại với nhau. Tích giữa các là phép nhân từng phần tử của các ma trận hiệp phương sai tương ứng tương ứng của từng . Điều này có nghĩa là hiệp phương sai của hai tích được nhân với nhau sẽ chỉ có giá trị cao nếu cả hai hiệp phương sai đều có giá trị cao. Do đó, phép tích của nó cũng làm một phép *AND*

1.2.8.1 kết hợp các hàm kernel bằng phương pháp bổ sung (OR):

Có thể hiểu đây là phương pháp bổ sung một khác vào từng phần tử của một ma trận hiệp phương sai tương ứng của nó. Hiệp phương sai của được bổ sung vào sẽ chỉ có giá trị thấp nếu cả hai hiệp phương sai đều có giá trị thấp. Do đó, phép toán cộng có thể được hiểu là một phép toán OR

1.3 Gaussian Process trong Regression:

1.3.1 Dự đoán với các quan sát không nhiễu (Noise-free Observations):

1.3.1.1 Gaussian process prior

Đầu tiên ta sẽ tiến hành định nghĩa các tập dữ liệu:

* Một Training dataset với là số quan sát:

|  |  |
| --- | --- |
|  | (1.12) |

* Một Test dataset với số quan sát:

|  |  |
| --- | --- |
|  | (1.13) |

Trong trường hợp không có nhiễu, việc tính toán/ lấy mẫu các điểm diễn ra một cách trực tiếp. Có nghĩa là, giá trị dự đoán suy ra từ hàm là bằng chính giá trị biến mục tiêu *y (của ):*

|  |  |
| --- | --- |
|  | (1.14) |

Dựa vào các định nghĩa tập và và ta có thể xây dựng phân phối đồng thời của các hàm Gaussian process trên *train* và *test* *dataset* (tương ứng là f và f ∗):

|  |  |
| --- | --- |
|  | (1.15) |

Trong đó :

* : Là một ma trận của các kích thước
* : Là một ma trận kích thước

Với và là số lượng tập train và tập test và là số

* **Là một** vector *,* dùng để thực hiện một Gaussian process trên các
* **:** là một vector dùng để thực hiện một Gaussian process trên các
* Là hàm
  + Là giữa tất cả các điểm dữ liệu đã được huấn luyện
  + Là giữa các điểm quan sát mới *(test points)*
  + Là giữa tất cả các điểm mới *(test points)* so với tất cả các điểm đã huấn luyện và ngược lại với .

1.3.1.2 Gaussian process posterior:

Lúc này, ta sẽ quan tâm đến việc dự đoán giá trị hàm bằng cách xây dựng phân phối hậu nghiệm (posterior distribution) dựa trên việc khai triển công thức tiền nghiệm1.15 sang phân phối có điều kiện (trong đó được tính với điều kiện ) ta được :

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

Trong đó:

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

Giải thích hàm predicted *mean* và predicted covariance:

* : Giá trị trung bình dự đoán được: là hàm được ước tính dựa trên sự kết hợp tuyến tính của các quy trình Gaussian ***f***. Các hệ số *(coefficients)* của tổ hợp tuyến tính này được xác định bởi khoảng cách trong giữa các điểm và các điểm .
* :Hay Hiệp phương sai dự đoán được : Giá trị này được ước tính bằng độ khác biệt của *kernel distance* giữa các điểm trừ đi bậc hai khoảng cách *kernel* nghịch đảo của các đầu vào trước đó và tích hậu nghiệm *(post-multiplied)* với *kernel distance* giữa các điểm và

Ngoài ra, trong một số trường hợp mean được định nghĩa ban đầu khác không. Thì công thức mean sẽ được biến đổi thành:

|  |  |
| --- | --- |
|  | (1.19) |

1.3.2 Dự đoán với các quan sát có nhiễu:

1.3.2.1 Gaussian process prior:

Khác với trường hợp không có nhiếu lúc này giá trị biến mục tiêuđược tính bằng kết hợp với độ nhiệu được định nghĩa như sau:

|  |  |
| --- | --- |
|  | (1.20) |

Trong đó:

* là một có thể được tối ưu trong quá trình huấn luyện mô hình.

Từ định nghĩa công thức (1.6) ta thay ở phân phối tiền nghiệm sang

|  |  |
| --- | --- |
|  | (1.21) |

1.3.2.2 Gaussian process posterior:

Tương tự công thức phân phối hậu nghiệm có điều kiện (1.16) đã khai triển, tại trường hợp này ta sẽ thêm một phần nhiễu vào công thức như sau:

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

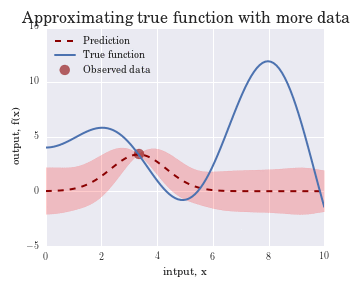
Trong đó:

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

Việc thêm nhiễu vào các phân phối Gaussian đảm bảo giúp đảm bảo cho các ma trận trong ngoặc luôn dương trong quá trình tối ưu hóa của GPR. Do đó cho phép ma trận này có thể đảo ngược.

Công thức 1.22 mô tả chính xác cách mà Gaussian process regression suy luận ra các dự đoán trong môi trường dữ liệu nhiễu. Tuy nhiên, một số phần tử trong công thức thì vẫn còn trong hộp đen( ), chẳng hạn: *kernel* **/** và .

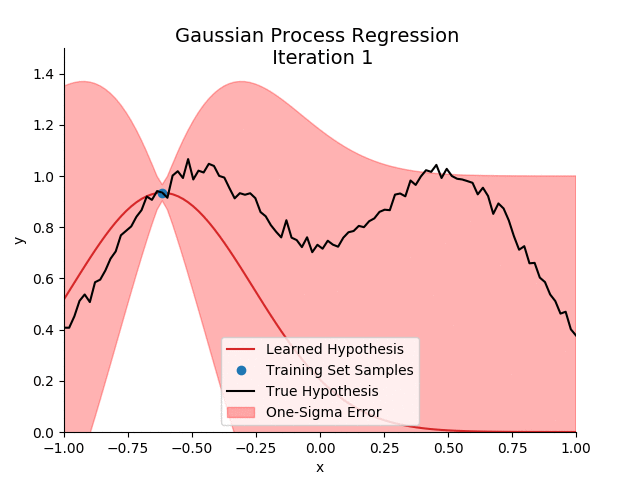
1.3.3 Quy trình lấy mẫu dự đoán dựa trên các phân phối hậu nghiệm của Gaussian Process Regression



Hình 8: Sơ đồ động biểu diễn quá trình cập nhật đường dự đoán của dựa trên hàm phân phối hậu nghiệm theo Gaussian process regression

Quan sát biểu đồ hình 8 ta thấy rằng: Quá trình trình lấy mẫu dự đoán diễn ra ngẫu nhiên trên các điểm qua sát. Tại những vị trí có nhiều điểm dữ liệu đường dự đoán được uốn theo đúng với hàm ta mo muốn và độ nhiễu thấp (vùng đỏ nhỏ dần). Mặc khác, tại các vùng không có dữ liệu thì phạm vi độ nhiễu (đỏ) sẽ rộng và nằm lệch rất xa so với đường thẳng mục tiêu.

* Để xây dựng một đường dự đoán tốt ta cần nhiều quan sát phủ đều trên miền dữ liệu. Tuy nhiên độ nhiễu dữ liệu phải hạn chế thấp nhất có thể để tránh gây rối loạn mô hình trong quá trình huấn luyện.



Hình 9: Sơ đồ mô phỏng quá trình huấn luyện Gaussian process regression

Theo Biểu đồ hình 9, ta thấy được rằng GPR chỉ hỗ trợ tốt khi cần xây dựng một đường cong uốn theo các điểm quan sát và miễn dữ liệu hẹp. Do đó, ta thấy trong trường hợp này đường thẳng mục tiêu (đúng) màu đen lại là một đường cong nhưng có gấp khúc tại nhiều điểm dữ liệu. Tuy GPR có thể uốn theo được dữ liệu mà nó quan sát nhưng chỉ phù hợp cho dữ liệu ít outlier và do đó trong trong trường hợp này mô hình dù hội tụ tốt nhưng độ chính xác sẽ không cao.

1.3.4 Giải thích về việc Gaussian Process Regression là một hàm phi tham số:

Từ các phương trình phân phối hậu nghiệm*,* ta có thể suy ra bản chất của là sự kết hợp của mô hình và các hàm để chiếu các điểm dữ liệu *input* vào không gian đặc trưng *(feature space).*

Mặc khác, để suy luận ra các dự đoán mới, mỗi đầu ra có trọng số được tính dựa theo mức độ tương tự của đầu vào so với điểm được dự đoán bằng phép đo độ tương tự được đo bởi *kernel*.

* Các giá trị dự đoán hậu nghiệm là sự kết hợp tuyến tính của các đặc trưng *(features)* và trọng số tại điểm quan sát có .

Về khái niệm, đây là một không gian tham số *vô hạn* được tổng hợp từ một *tổng hữu hạn* khi thực hiện các phép dự đoán hậu nghiệm. Từ đó, ta thấy mô hình này chỉ phụ thuộc vào đã chọn và cả tập dữ liệu đã quan sát được. Đây là lý do tại sao Gaussian Process Regression được gọi là *một mô hình phi tham số.*

CHƯƠNG II: XÂY DỰNG MÔ HÌNH GAUSSIAN PROCESS REGRESSION DỰA TRÊN TENSORFLOW PROBABILITY

2.1 Tổng quan về đề tài:

2.1.1 Lý do chọn đề tài:

Đề tài lần này mang tên dự đoán Hàm lượng/ nồng độ khí trong bầu khí quyển dựa trên phương pháp Gaussian process regression, Mục đích của bài toán này bao gồm 2 mục tiêu chính đó là:

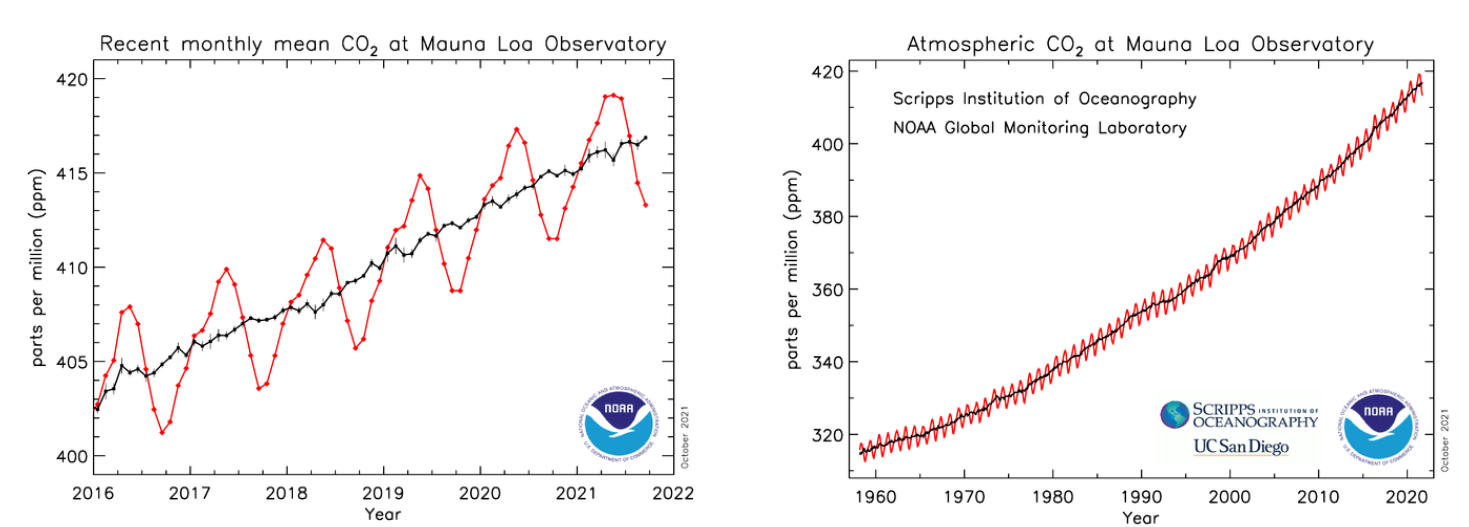
* Tìm ra quy luật tăng dần của từ năm 1958 đến nay (2021), giúp kiểm tra được các bài báo hay số liệu về việc nồng độ khí hiện nay đã tăng đột biến từ khoảng thời điểm nào.
* Do sử dụng một bộ dữ liệu sạch được cung cấp bởi tổ chức phi lợi nhuận nhưng có độ phủ rồng từ năm 1958 đến nay. Do đó giúp mô phỏng tốt cách hoạt động của thuật toán từ xây dựng các hàm trung bình, kernel cho tới xây dựng hàm tiền và hậu nghiệm.

2.1.2 Tổng quan về dataset:

2.1.2.1 Giới thiệu tập dữ liệu:

Kể từ năm 1958 đến nay, đài quan sát khí tượng thủy văn tại ***Mauna Loa*** (Hawaii), đã liên tục theo dõi và thu thập dữ liệu về các thành phần hóa học của khí quyển và đặt biệt là mật độ khí *carbon dioxide* (). Tuy nhiên khả năng hấp thụ CO2 của tự nhiên cũng thay đổi theo từng mùa.

Theo hình 10 (bên dưới), Vào mùa hè, nồng độ kh có trong khí quyển giảm xuống rõ rệt vì các loại thực vật hấp thụ nhiều carbon hơn phục vụ cho quá trình phát triển của chúng. Ngược lại, vào mùa đông chúng sử dụng ít hơn khiến cho nồng độ khí thải tại tầng khí quyển tăng rõ rệt. Rõ ràng hơn ta có thể quan sát ở hình



Hình 10: Sơ đồ mô thống kê nồng độ khí CO2 trên khí quyển bào các mùa trong năm (2016 - 2022) theo [1]

2.1.2.2 Lý do chọn tập dữ liệu trên:

Ta có một số lý do chính như sau:

* Tập dữ liệu có độ phủ rộng từ năm 1958 đến nay và được thu thập định kỳ vào mỗi tháng bởi trạm NOAA [1]. Giúp giữ liệu có độ liên tục
* Tập dữ liệu có độ nhiễu thấp và không chịu ảnh hưởng bởi yếu tố bên ngoài như hành vi con người như dữ liệu được thu thập qua khảo sát một người.
* Mặc khác, trạm khí tượng được đặt tại Hawaiin, nơi ít chịu tác động đến từ các nhà máy xí nghiệp hay khói xe từ các phương tiện Oto gây rối loạn nồng độ CO2 có trong khu vực đo đạc. Điều này giúp cho số liệu đo đạc có cùng điều kiện đo đạc dù qua các năm.

2.2 Đọc và xử lý dữ liệu

2.2.1 Đọc dataset:

Đầu tiên ta sẽ tiến hành quá trình đọc file CSV. Ngoài ra, do giá trị NAN trong dataset được trung tâm nghiên cứu sử dụng là , nên ta sẽ tiến hành replace nó trong quá trình đọc file.

Do Đây là dữ liệu dùng để miêu tả cho phần lý thuyết ở Chương 1 với , do đó ta sẽ sử dụng các giá trị cột Date làm và và cột CO2 (nồng độ) = tại các điểm dữ liệu tại mỗi index i đó. Cụ thể được miêu tả bằng đoạn code dưới đây:

#Thư viện đọc file csv

import pandas as pd

import numpy as np

#truy cập directory

import os

#Loại cảnh báo không cần thiết

import warnings

warnings.simplefilter("ignore")

PROJECT\_PATH = "/content/drive/MyDrive/RA\_n\_PA\_Project/"

df = pd.read\_csv(PROJECT\_PATH + "monthly\_in\_situ\_co2\_mlo.csv",

                  header=None,

# Bỏ qua phần credit trong file CSV

                  skiprows=57,

# Sử dụng 2 cột Date(X) và CO2(y)

                  usecols=[3, 4],

#Giá trị NaN trong file là -99.99

                  na\_values='-99.99',

#chuyển sang float64 phù hợp cho tf

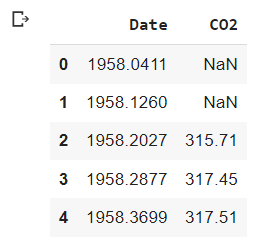
                  dtype=np.float64)

#Định nghĩa tên cột

df.columns = ["Date", "CO2"]

df.head()

Sau khi tiến hành đọc file CSV ta được :

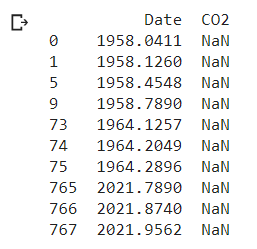
…

2.***2.2 Tiền xử lý và phân tích dữ liệu:***

Đầu tiên ta sẽ tiến hành lọc bỏ các dòng chứa giá trị NaN gây lỗi khi huấn luyện mô hình theo cách sau. Các dòng lỗi sẽ loại bỏ ngay khi phát hiện:

print(df[df.isnull().any(axis=1)])

df.dropna(inplace=True)



print("Số dòng sau khi xóa NaN: ", df.shape[0])



Tiếp theo ta sẽ thống kê sơ bộ về tập dữ liệu:

|  |  |
| --- | --- |
| df.describe() |  |

Sau khi có được giá trị min-max của 2 cột ta tiến hành vẽ sơ đồ sử dụng thư viện ***bokeh*** (cho phép tương tác với sơ đồ , nắm kéo):

from tqdm.notebook import tqdm

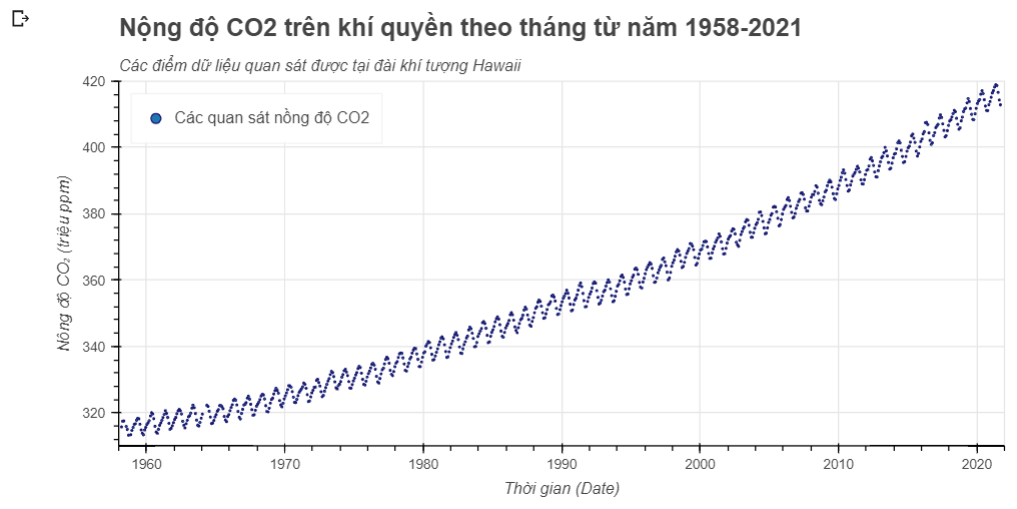
import bokeh

import bokeh.io

import bokeh.plotting

import bokeh.models

from IPython.display import display, HTML



bokeh.io.output\_notebook(hide\_banner=True)

# Vẽ Dữ liệu với X là thời gian và y là nồng độ CO2

fig = bokeh.plotting.figure(

    width=800, height=400,

    x\_range=(1958, 2022), y\_range=(310, 420))

fig.xaxis.axis\_label = 'Thời gian (Date)'

fig.yaxis.axis\_label = 'Nông độ CO₂ (triệu ppm)'

fig.add\_layout(bokeh.models.Title(

    text='Các điểm dữ liệu quan sát được tại đài khí tượng Hawaii',

    text\_font\_style="italic"), 'above')

fig.add\_layout(bokeh.models.Title(

    text='Nộng độ CO2 trên khí quyền theo tháng từ năm 1958-2021',

    text\_font\_size="15pt"), 'above')

fig.circle( df.Date, df.CO2, legend\_label='Các quan sát nồng độ CO2', size=1.5, line\_color='midnightblue')

fig.legend.location = 'top\_left'

fig.toolbar.autohide = True

bokeh.plotting.show(fig)

Hình 11: Sơ đồ biễu diễn tập dữ liệu từ năm 1958-2021

Dựa theo hình 11, ta thấy được rằng lượng ngày một tăng cao qua các năm từ . Ngoài ra lượng khí thải cũng tăng giảm có chu kỳ qua các mùa trong năm. Do đó nếu sử dụng phương thức linear regression thông thường bằng một đường thẳng để dự đoán cho dữ liệu trên sẽ gây ra nhiều sai sót trong quá trình dự đoán. Do đó, *Gaussian process regression* là lựa chọn phù hợp để biểu diễn đường cong uốn theo theo chính các quan sát trên.

Sau khi phân tích tập dữ liệu ta sẽ tiến hành chia tập quan sát Observed set (X\_train) và tập dùng để dụ đoán X\_test.

# Chia tập dataset thành 2 tập X observed và X\* để dự đoán

date\_split\_predict = 2012

X\_train = df[df.Date < date\_split\_predict]

print('Sử dụng {} dòng trước năm 2012 cho observed set'.format(len(X\_train)))

X\_test = df[df.Date >= date\_split\_predict]

print('Sử dụng {} dòng trước sau năm 2012 cho test set'.format(len(X\_test)))



2.3 Định nghĩa các kernel cần thiết cho Gaussian process regresssion:

Đầu tiên, ta import các thư viện cần thiết về *tensorflow*, *tensorflow* *probability* và đặt tên các biến viết tắt.

# Imports các thư viện hỗ trợ xác suất trong tensorflow

import tensorflow as tf

import tensorflow\_probability as tfp

# Định nghĩa tên viết tắt cho các hàm thường dùng

tfb = tfp.bijectors

tfd = tfp.distributions

tfk = tfp.math.psd\_kernels

np.random.seed(42)

tf.random.set\_seed(42)

2.3.1 Định nghĩa hàm kernel Mean():

Mean là hàm sử dụng cho sử dụng cho quá trình tính tfd.Gaussianprocess và Gaussian process regression model. Dùng để tính trung bình của tất cả các quan sát, ta sẽ định nghĩa nó bằng cách sử dụng numpy như sau:

observations\_mean = tf.constant(

[np.mean(X\_train.CO2.values)], dtype=tf.float64)

mean\_fn = lambda \_: observations\_mean

2.3.2 Xây dựng kết hợp các hàm covariance kernel:

2.3.2.1 Định nghĩa chuẩn chung cho các hyper-parameter:

Các phân phối sẽ tuân theo phân phối kiểu *float64* để phù hợp với tập dữ liệu đã đọc và các siêu tham số sẽ luôn dương giúp cho giảm lỗi khi tính toán ma trận bằng thư viện *tensorflow* *(được mô tả ở tiểu mục 1.2.1)*. Do đó ta sẽ định nghĩa một biến dùng cho .

# Định nghĩa điều kiện luôn dương cho các siêu tham số

constrain\_positive = tfb.Shift(np.finfo(np.float64).tiny)(tfb.Exp())

2.3.2.2 Định nghĩa hyper-parameter cho hàm Exponentiated Quadratic Kernel:

Đầu tiên, ta sẽ định nghĩa 2 tham số amplitude và length scale bằng hàm *tfp.util.TransformedVariable*. Sau đó, tiến hành sử dụng 2 tham số vừa khởi tạo này chuyển vào hàm Exponentiated Quadratic Kernel *(được mô tả ở tiểu mục 1.2.2*) bằng cách sử dụng hàm *tfk.ExponentiatedQuadratic, như sau:*

# Định nghĩa biên độ amplitude

exp\_amplitude = tfp.util.TransformedVariable(

    initial\_value=10., bijector=constrain\_positive,

    dtype=np.float64,name='exp\_amplitude')

# Định nghĩa khoảng cách length\_scale

exp\_length\_scale = tfp.util.TransformedVariable(

    initial\_value=10., bijector=constrain\_positive,

    dtype=np.float64,name='exp\_length\_scale')

# Xây dựng hàm Exponentiated Quadratic Kernel

exp\_kernel = tfk.ExponentiatedQuadratic(

    amplitude=exp\_amplitude, length\_scale=exp\_length\_scale)

2.3.2.3 Định nghĩa hyper-parameter cho hàm Rational Quadratic kernel:

Tương tự, ta định nghĩa tham số , và *(alpha scale-mixture, với α>0)* bằng *tfp.util.TransformedVariable.*

Sau đó, chuyển 3 tham số vừa định nghĩa vào hàm Rational Quadratic kernel *(được mô tả ở tiểu mục 1.2.5)*. thông qua hàm tfk.RationalQuadratic:

# Các hyper- parameter của Rational Quadratic kernel:

# Định nghĩa biên độ amplitude

rational\_amplitude = tfp.util.TransformedVariable(

    initial\_value=1., bijector=constrain\_positive,

    dtype=np.float64,name='rational\_amplitude')

# Định nghĩa khoảng cách length scale

rational\_length\_scale = tfp.util.TransformedVariable(

    initial\_value=1., bijector=constrain\_positive,

    dtype=np.float64,name='rational\_length\_scale')

# Định nghĩa giá trị alpha scale-mixture (alpha > 0)

rational\_scale\_mixture = tfp.util.TransformedVariable(

    initial\_value=1., bijector=constrain\_positive,

    dtype=np.float64,name='rational\_scale\_mixture')

#Xây dựng hàm Rational Quadratic kernel

rational\_kernel = tfk.RationalQuadratic(

    amplitude=rational\_amplitude,

    length\_scale=rational\_length\_scale,

    scale\_mixture\_rate=rational\_scale\_mixture)

2.3.2.4 Định nghĩa hyper-parameter cho hàm local periodic kernel:

Đối với hàm *local periodic kernel* này sẽ định nghĩa lần lượt cho 2 kernel con *(cụ thể ở tiểu mục 1.2.7)*:

* Đầu tiên, là định nghĩa tham số cho periodic kernel với 3 tham số: , và bằng hàm tfp.util.TransformedVariable

# Định nghĩa biên độ amplitude của hàm periodic.

periodic\_amplitude = tfp.util.TransformedVariable(

    initial\_value=5.0, bijector=constrain\_positive,

    dtype=np.float64,name='periodic\_amplitude')

# Định nghĩa khoảng cách lengthscale của hàm periodic.

periodic\_length\_scale = tfp.util.TransformedVariable(

    initial\_value=1.0, bijector=

    constrain\_positive, dtype=np.float64,

    name='periodic\_length\_scale')

* Tiếp theo, ta có local periodic kernel sử dụng chung nên ta chỉ cần định nghĩa tham số riêng cho hàm Exponentiated Quadratic bằng tfp.util.TransformedVariable

# Định nghĩa chu kỳ (period) của hàm periodic.

periodic\_period = tfp.util.TransformedVariable(

    initial\_value=1.0, bijector=constrain\_positive,

    dtype=np.float64,name='periodic\_period')

# Định nghĩa lengthscale của hàm Exponentiated.

periodic\_exp\_length\_scale = tfp.util.TransformedVariable(

    initial\_value=1.0, bijector=constrain\_positive,

    dtype=np.float64,name='periodic\_exp\_length\_scale')

Sau khi định nghĩa 4 tham số cần thiết,ta chuyển lần lượt các tham số vào 2 hàm kernel con. Sau đó tiến hành kết hợp 2 kernel bằng phép nhân để thu được hàm *local periodic kernel*

# Xây dựng  hàm kernel local periodic kernel

# bằng phép nhân 2 hàm kernel lại với nhau

local\_periodic\_kernel = (

    #Hàm periodic kernel

    tfk.ExpSinSquared(

        amplitude=periodic\_amplitude,

        length\_scale=periodic\_length\_scale,

        period=periodic\_period) \*

    # nhân với Hàm ExponentiatedQuadratic kernel

    tfk.ExponentiatedQuadratic(

        length\_scale=periodic\_exp\_length\_scale))

2.3.2.4 Lập kernel tổng:

# tổng tất cả 3 kernels thành một kernel:

kernel = (exp\_kernel + local\_periodic\_kernel + rational\_kernel)

2.4 ***Xây dựng mô hình:***

2.4.1 Tổng hợp các tham số huấn luyện:

Định nghĩa tham số nhiễu noise variance của các quan sát:

# Định nghĩa noise variance cho các quan sát huấn luyện

observation\_noise\_variance = tfp.util.TransformedVariable(

    initial\_value=1, bijector=constrain\_positive,

    dtype=np.float64,name='observation\_noise\_variance')

    name='observation\_noise\_variance')

    name='observation\_noise\_variance')

local\_periodic\_kernel = (

    #Hàm periodic kernel

    tfk.ExpSinSquared(

        amplitude=periodic\_amplitude,

        length\_scale=periodic\_length\_scale,

        period=periodic\_period) \*

    # nhân với Hàm ExponentiatedQuadratic kernel

    tfk.ExponentiatedQuadratic(

        length\_scale=periodic\_exp\_length\_scale))

Tổng hợp tất cả các tham số huấn luyện lại thành một list gọi là *trainable\_variables* :

#Tổng hợp lại tất cả các tham số huấn luyện đã tạo

trainable\_variables = [v.variables[0] for v in [

    #Các hyper-parameter của ExponentiatedQuadratic

    exp\_amplitude,

    exp\_length\_scale,

    #Các hyper-parameter của Rational Quadratic kernel

    rational\_amplitude,

    rational\_length\_scale,

    rational\_scale\_mixture,

    #Các hyper-parameter của local periodic kernel

    periodic\_amplitude,

    periodic\_length\_scale,

    periodic\_period,

    periodic\_exp\_length\_scale,

    #Độ nhiễu của các quan sát.

    observation\_noise\_variance

]]

Tiếp theo, Cấu hình batch size = 128 và thông tin của batch dựa trên hàm .from\_tensor\_slices dựa trên kích thước vừa định nghĩa:

#Kích thước batch

batch\_size = 128

#Định nghĩa các batch dựa trên batch size

batched\_dataset = (

    tf.data.Dataset.from\_tensor\_slices(

    (X\_train.Date.values.reshape(-1, 1),X\_train.CO2.values))

    .shuffle(buffer\_size=len(X\_train)).repeat(count=None).batch(batch\_size))

2.4.2 Định nghĩa hàm loss:

Mô hình sử dụng hàm loss quen thuộc là *negative log likelihood:*

@tf.function(autograph=False, experimental\_compile=False)

def gp\_loss\_fn(index\_points, observations):

# Hàm loss  negative-log-likelihood cho Gaussian process

    gp = tfd.GaussianProcess(

        mean\_fn=mean\_kernel,

        kernel=kernel,

        index\_points=index\_points,

        observation\_noise\_variance=observation\_noise\_variance)

    negative\_log\_likelihood = -gp.log\_prob(observations)

    return negative\_log\_likelihood

2.4.3 Tiến hành huấn luyện mô hình:

Sau khi huấn luyện mô hình ta thuật lại quá trình huấn luyện như sau:

Sơ đồ mô tả quá trình huấn luyện sau 10.000 vòng lặp:

from itertools import islice

import collections

# Sử dụng Adam optimizer với learning rate = 0.001

optimizer = tf.keras.optimizers.Adam(learning\_rate=0.001)

# Lưu Negative Log-Likelihood (NLL) của cả batch

batch\_loss = []

# Lưu Negative Log-Likelihood (NLL) của cả data

Loss\_hist = []

#số lần train

iter=10000

for i, (index\_points\_batch, observations\_batch) in tqdm(

        enumerate(islice(batched\_dataset, iter)), total=iter):

    # Tiến hàn huấn luyện cho từng batch

    with tf.GradientTape() as tape:

        loss = gp\_loss\_fn(index\_points\_batch, observations\_batch)

    grads = tape.gradient(loss, trainable\_variables)

    optimizer.apply\_gradients(zip(grads, trainable\_variables))

    batch\_loss.append((i, loss.numpy()))

    # Đánh giá trên tất cả các quan sát

    if i % 100 == 0:

        # Đánh giá trên tất cả các dữ liệu đã quan sats quan sát

        ll = gp\_loss\_fn(

            index\_points=X\_train.Date.values.reshape(-1, 1),

            observations=X\_train.CO2.values)

        Loss\_hist.append((i, ll.numpy()))

Lập sơ đồ biễu diễn quá trình huấn luyện để thuật lại quá trình hội tụ:

# Lập đồ thị loss qua các lần train

fig = bokeh.plotting.figure(

    width=600, height=350,

    x\_range=(0, iter), y\_range=(50, 200))

fig.add\_layout(bokeh.models.Title(

    text='Negative Log Likelihood(loss) trong training',

    text\_font\_size="14pt"), 'above')

fig.xaxis.axis\_label = 'số lần train'

fig.yaxis.axis\_label = 'loss của batch'

# Biểu diễn loss của các batch

fig.line(

    \*zip(\*batch\_loss), legend\_label='Batch data (loss)',

    line\_width=2, line\_color='lightblue')

# Biểu diễn loss của toàn bộ Training data

fig.extra\_y\_ranges = {

    'fig1ax2': bokeh.models.Range1d(start=130, end=250)}

fig.line(

    \*zip(\*Loss\_hist), legend\_label='Training data (loss)',

    line\_width=2, line\_color='red', y\_range\_name='fig1ax2')

# Thêm layout cho cột phải của toàn bộ loss

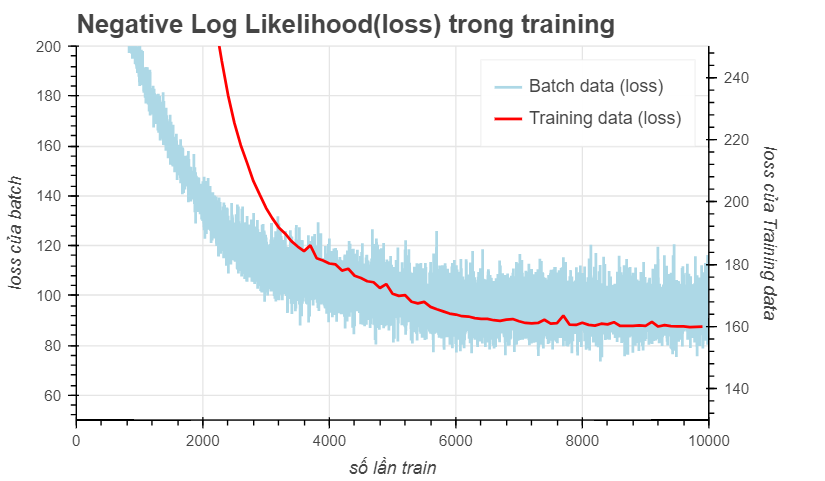
fig.add\_layout(bokeh.models.LinearAxis(

    y\_range\_name='fig1ax2', axis\_label='loss của Training data'), 'right')

fig.legend.location = 'top\_right'

fig.toolbar.autohide = True

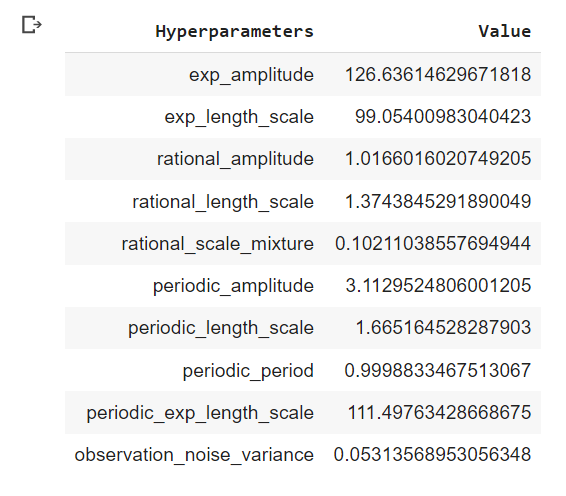
bokeh.plotting.show(fig)



Hình 12: Sơ đồ loss của quá trình huấn luyện theo tape.gradient

2.4.4 Xây dựng hàm dự đoán hậu nghiệm:

Sau khi huấn luyện mô hình ta thu được danh sách các siêu tham số đã được cập nhật như sau (ví dụ minh họa, vì sẽ thay đổ sau mỗi lần train):



Dựa vào các tham số trên đã được tối ưu hóa sau quá trình huấn luyện mô hình ta sẽ tiến hành xây dựng hàm dựa đoán hậu nghiệm sau:

#Xây dựng hàm dự đoán hậu nghiệm Posterior Gaussian process

#Sử dụng các kernel đã xây dựng và các quan sát đã học

gprm = tfd.GaussianProcessRegressionModel(

    mean\_fn=mean\_kernel,

    kernel=kernel,

    index\_points=X\_test.Date.values.reshape(-1, 1),

    observation\_index\_points=X\_train.Date.values.reshape(-1, 1),

    observations=X\_train.CO2.values,

    observation\_noise\_variance=observation\_noise\_variance)

Lấy đường dự đoán hậu nghiệm trung bình thu được từ Gaussian process regression model và độ lệch chuẩn xung quoanh đường dự đoán đó:

# Dự đoán hậu nghiệm trung bình hàm hậu nghiệm

gp\_mean = gprm.mean().numpy()

# Dự đoán độ lệch chuẩn của kết quả hàm hậu nghiệm

gp\_noise = gprm.stddev().numpy()

Vẽ biểu đồ phân bố các mẫu đữ đoán hậu nghiệm thư được từ suy luận của mô hình Gaussian process regression:

# Dự đoán hậu nghiệm trung bình hàm hậu nghiệm

gp\_mean = gprm.mean().numpy()

# Dự đoán độ lệch chuẩn của kết quả hàm hậu nghiệm

gp\_noise = gprm.stddev().numpy()

# X từ 2012 -> 2021

# y từ nồng độ 384 -> 418

fig = bokeh.plotting.figure(

    width=800, height=600,

    x\_range=(2012, 2021.3), y\_range=(384, 418))

fig.add\_layout(bokeh.models.Title(

    text='kết quả dự đoán của Gaussian process regression dựa trên dataset đã huấn luyện cho tập test sau năm 2012.',

    text\_font\_style="italic"), 'above')

#Định nghĩa các thành phần trong sơ đồ

fig.xaxis.axis\_label = 'Thời gian (Date)'

fig.yaxis.axis\_label = 'Lượng CO₂ (ppm)'

fig.add\_layout(bokeh.models.Title(text='Nồng độ CO2 trong khí quyển',

                                  text\_font\_size="14pt"), 'above')

fig.circle(df.Date, df.CO2, legend\_label='Các quan sat trên Test data',

           size=2, line\_color='darkblue')

fig.line(X\_test.Date.values, gp\_mean,

          legend\_label='Gaussian process mean function (predictions)',

          line\_width=2, line\_color='red')

# Vẽ khoảng tin cậy confidence interval (CI)

band\_x = np.append(X\_test.Date.values, X\_test.Date.values[::-1])

band\_y = np.append((gp\_mean + 2\*gp\_noise), (gp\_mean - 2\*gp\_noise)[::-1])

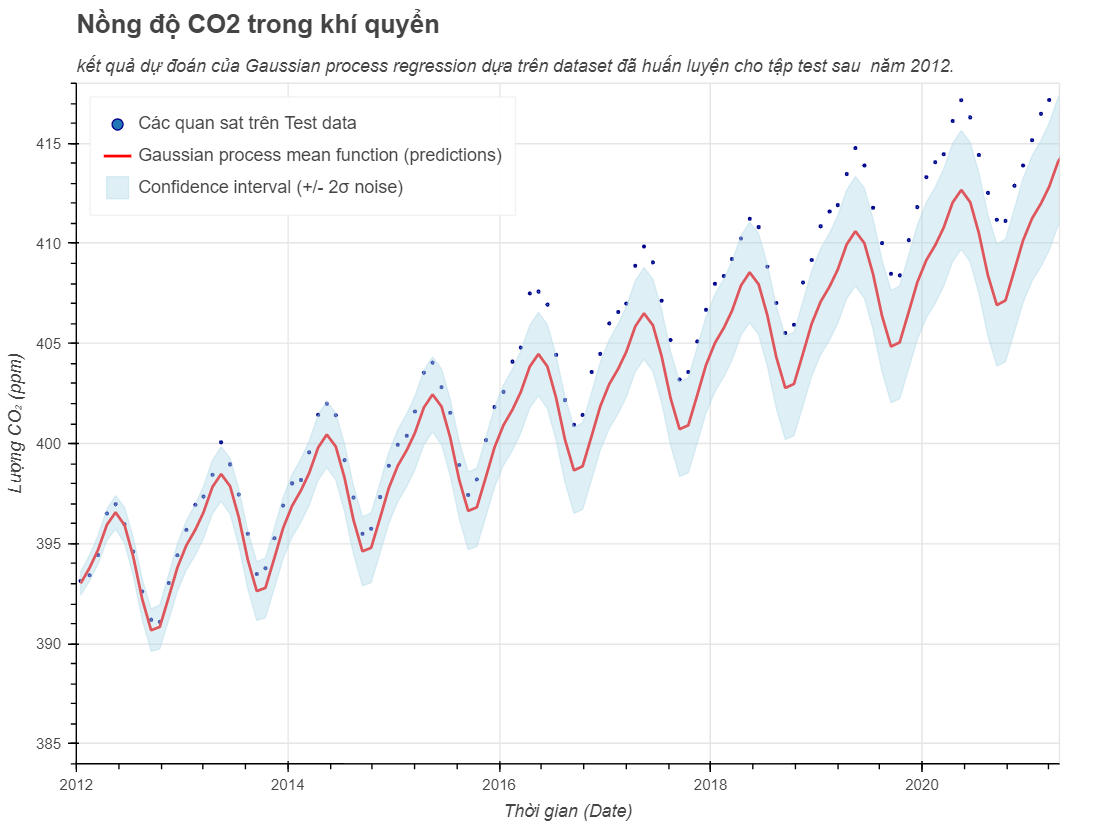
fig.patch(band\_x, band\_y, color='lightblue', alpha=0.4,

    line\_color='lightblue', legend\_label='Confidence interval (+/- 2σ noise)')

fig.legend.location = 'top\_left'

fig.toolbar.autohide = True

bokeh.plotting.show(fig)



Hình 13: Sơ đồ biểu diễn kết quả dự đoán của Gaussian process regression dựa trên dataset đã huấn luyện cho tập test sau năm 2012

2.5 Nhận xét kết quả thu được:

Theo như kết quả quan sát được từ hình 13, ta có thể thấy được rằng:

Từ khoảng sau năm 2012 khi thời kỳ công nghiệp hóa bùng nổ, lúc này lượng khí thải bắt đầu tăng lên dần. Hay từ năm 2014 là rõ ràng nhất nồng độ ở thời điểm này đã năm ở rìa của độ lệch chuẩn so với hàm dự đoán trung bình mà mô hình suy luận được. Điều đó có nghĩa là, lượng khí thải đang bắt đầu tăng lên đột ngột, điển hình là khoảng giữa năm 2018-2019 lượng đã vượt ra khỏi độ lệch chuẩn.

Mặc khác, đây cũng chỉ là việc dự đoán dựa trên 1 đặc trưng là thời gian. Nên kết quả nó có thể sẽ không khách quan và độ chính xác thấp. Tuy nhiên, mục đích của bài toán là để biểu diễn cho quá trình suy luận của phương pháp Gaussian process regression. Nên nhìn chung vẫn mang lại hiểu quả mô phỏng khá tốt.

TÀI LIỆU THAM KHẢO

1. [Earth System Research Laboratories, Global Monitoring Laboratory](https://gml.noaa.gov/ccgg/trends/)
2. [Gaussian Process Regression in TensorFlow Probability](https://www.tensorflow.org/probability/examples/Gaussian_Process_Regression_In_TFP)
3. [tfp.mcmc.sample\_chain  |  TensorFlow Probability](https://www.tensorflow.org/probability/api_docs/python/tfp/mcmc/sample_chain)
4. [Quick Start to Gaussian Process Regression | by Hilarie Sit | Towards Data Science](https://towardsdatascience.com/quick-start-to-gaussian-process-regression-36d838810319)
5. [Gaussian Processes for Dummies · (katbailey.github.io)](https://katbailey.github.io/post/gaussian-processes-for-dummies/)
6. [A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions | bioRxiv](https://www.biorxiv.org/content/10.1101/095190v3.full)