

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**🕯✡🕮🕮✡🕯**



**ĐỒ ÁN MÔN HỌC HỌC MÁY**

**TÌM HIỂU MỘT SỐ THUẬT TOÁN TRONG HỌC MÁY**

**GVHD: ThS. Trần Nhật Quang**

**SVTH: MSSV**

**Đào Xuân Thủy 16110544**

**Lâm Phước Bảo 16110016**

**Lớp thứ 6 – Tiết 1234**

**182MALE431085\_01CLC**

**Tp. Hồ Chí Minh, tháng 4 năm 2019**

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Giáo viên hướng dẫn

ThS. Trần Nhật Quang

**NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

Giáo viên phản biện

………………….....

**LỜI CẢM ƠN**

Để hoàn thành đồ án môn học này, chúng em xin gửi lời cảm ơn chân thành đến thầy Trần Nhật Minh, người đã hỗ trợ và giúp đỡ chúng em trong suốt quá trình thực hiện đồ án, nhận xét và góp ý cũng như cung cấp tài liệu tham khảo, giúp chúng em có thể hoàn thành đồ án một cách tốt nhất. Nếu không có sự hướng dẫn và kinh nghiệm thực tiễn của thầy, chúng em nghĩ rằng đồ án môn học của chúng em khó có thể hoàn thiện và hoàn thành đúng thời hạn. Một lần nữa chúng em xin cảm ơn thầy.

Chúng em xin gửi lời cảm ơn đến các bạn cùng khóa đã cung cấp nhiều thông tin và kiến thức hữu ích giúp cho chúng em hoàn thiện đồ án này.

Đồ án này được thực hiện trong thời gian có hạn, cùng với kiến thức còn hạn chế và còn nhiều bỡ ngỡ khác, do đó thiếu sót là điều không thể tránh khỏi nên chúng em rất mong nhận được những ý kiến đóng góp quý báu của mọi người để kiến thức của chúng em được hoàn thiện hơn sau này. Chúng em xin chân thành cảm ơn.

*Thành phố Hồ Chí Minh, ngày 19 tháng 05 năm 2019*

*Sinh viên thực hiện*

Thuy Bao

Đào Xuân Thủy Lâm Phước Bảo

**MỤC LỤC**

[DANH MỤC CÁC HÌNH 4](#_Toc9198326)

[DANH MỤC CÁC BẢNG 5](#_Toc9198327)

[MỘT SỐ THUẬT NGỮ SỬ DỤNG TRONG ĐỒ ÁN 6](#_Toc9198328)

[CHƯƠNG 1: TÌM HIỂU VỀ ĐẠO VĂN 8](#_Toc9198329)

[1. Đạo văn là gì? 8](#_Toc9198330)

[2. Những điều nên làm 8](#_Toc9198331)

[3. Những điều không nên làm 8](#_Toc9198332)

[4. Lời cam đoan 8](#_Toc9198333)

[CHƯƠNG 2: MULTIVARIATE LINEAR REGRESSION 10](#_Toc9198334)

[1. Đôi nét lịch sử 10](#_Toc9198335)

[2. Ứng dụng 10](#_Toc9198336)

[3. Mô tả thuật toán 10](#_Toc9198337)

[4. So sánh với các thuật toán khác 13](#_Toc9198338)

[5. Lập trình thuật toán 13](#_Toc9198339)

[CHƯƠNG 3: POLYNOMIAL REGRESSION 15](#_Toc9198340)

[1. Đặt vấn đề 15](#_Toc9198341)

[2. Ứng dụng 16](#_Toc9198342)

[3. Mô tả thuật toán 16](#_Toc9198343)

[3.1 Mô tả dữ liệu 16](#_Toc9198344)

[3.2 Phương trình biểu diễn mô hình của thuật toán 17](#_Toc9198345)

[3.3 Cách tính và đánh giá hiệu năng của mô hình 18](#_Toc9198346)

[4. So sánh với các thuật toán khác 19](#_Toc9198347)

[5. Lập trình thuật toán 20](#_Toc9198348)

[5.1 Bộ dữ liệu cho thuật toán 20](#_Toc9198349)

[5.2 Mục tiêu thuật toán 20](#_Toc9198350)

[5.3 Thư viện sử dụng 21](#_Toc9198351)

[5.3 Quá trình thực hiện 21](#_Toc9198352)

[5.4 Kết quả thu được 22](#_Toc9198353)

[CHƯƠNG 4: LOGISTIC REGRESSION 23](#_Toc9198354)

[1. Đặt vấn đề 23](#_Toc9198355)

[2. Ứng dụng 23](#_Toc9198356)

[3. Mô tả thuật toán 24](#_Toc9198357)

[4. Lập trình thuật toán 26](#_Toc9198358)

[4.1 Bộ dữ liệu cho thuật toán 26](#_Toc9198359)

[4.2 Mục tiêu thuật toán 26](#_Toc9198360)

[4.3 Thư viện sử dụng 27](#_Toc9198361)

[5.3 Quá trình thực hiện 27](#_Toc9198362)

[5.4 Kết quả thu được 28](#_Toc9198363)

[CHƯƠNG 5: NEURAL NETWORKS 30](#_Toc9198364)

[1. Đôi nét lịch sử 30](#_Toc9198365)

[2. Ứng dụng 30](#_Toc9198366)

[3. Mô tả thuật toán 31](#_Toc9198367)

[4. So sánh với các thuật toán khác 34](#_Toc9198368)

[5. Cài đặt thuật toán 34](#_Toc9198369)

[BẢNG PHÂN CÔNG CÔNG VIỆC 40](#_Toc9198370)

[PHỤ LỤC 41](#_Toc9198371)

[1. Scikit-learn 41](#_Toc9198372)

[2. Pandas 41](#_Toc9198373)

[3. Numpy 41](#_Toc9198374)

[4. Matplotlib 41](#_Toc9198375)

[5. OS 41](#_Toc9198376)

[TÀI LIỆU THAM KHẢO 44](#_Toc9198377)

# DANH MỤC CÁC HÌNH

[Hình 1. Mô hình Linear Regression 11](#_Toc9188694)

[Hình 2. Sai số thuật toán Linear Regression 11](#_Toc9188695)

[Hình 3. Vấn đề với Linear Regression 11](#_Toc9188696)

[Hình 4. So sánh Linear Regression với Polynomial Regression 12](#_Toc9188697)

[Hình 5. Kết quả so sánh Polynomial Regression và Linear Regression 15](#_Toc9188698)

[Hình 6. Overfitting với Linear Regression Hình 7. Overfitting với Polynomial Regression 16](#_Toc9188699)

[Hình 8. Mô tả dữ liệu mẫu cho thuật toán Polynomial Regression 16](#_Toc9188700)

[Hình 9. Kết quả thu được sau khi chạy thuật toán Polynomial Regression 18](#_Toc9188701)

[Hình 10. Mạng nơ ron cơ bản 20](https://d.docs.live.net/c594835c8ad49e8b/Machine%20Learning/Báo%20cáo%20học%20máy.docx#_Toc9188702)

[Hình 11. Các ký hiệu sử trong mô tả Neural Network 21](https://d.docs.live.net/c594835c8ad49e8b/Machine%20Learning/Báo%20cáo%20học%20máy.docx#_Toc9188703)

[Hình 13. Sơ đồ hiện thực Model 26](#_Toc9188704)

# DANH MỤC CÁC BẢNG

[Bảng 1. Bảng mô tả dữ liệu cho Polynomial Regression với dữ liệu đơn chiều 12](#_Toc9188705)

[Bảng 2. Bảng mô tả dữ liệu cho Polynomial Regression với dữ liệu đa chiều 12](#_Toc9188706)

# MỘT SỐ THUẬT NGỮ SỬ DỤNG TRONG ĐỒ ÁN

|  |  |
| --- | --- |
| Từ khóa | Ý nghĩa |
| fit | Định dạng / chỉnh sửa cho phù hợp |
| cost function | Sai số giữa kết quả thực tế và kết quả tìm được khi sử dụng thuật toán |
| underfitting | Mô hình tìm được bằng cách sử dụng thuật toán quá đơn giản, gây ra hiện tượng cost function lớn và khi áp dụng vào dữ liệu thực tế cho kết quả có độ sai số cao |
| overfitting | Mô hình tìm được bằng cách sử dụng thuật toán quá phức tạp, gây ra hiện tượng cost function nhỏ nhưng khi áp dụng vào dữ liệu thực tế thì cho kết quả có độ sai số cao |
| dataset | Bộ dữ liệu sử dụng cho thuật toán giúp thuật toán học được mô hình |
| learning rate | Chỉ số do lập trình viên thiết lập, giúp thuật toán chạy nhanh và tránh bị phân kỳ |
| Dense | Thư viện trong Keras, layer với full connection sử dụng như một layer neural network bình thường. Các tham số quan tâm:  Activation: dùng để chọn activatio mong muốn  Units: số lượng unit cho layer này  use\_bias: có sử dụng bias hay không (true hoặc false)  input\_dim: số lượng feature input. Cho layer đầu tiên |
| metrics | là thước đo để ta đánh giá accuracy của model. |
| loss functions | Sử dụng binary\_crossentropy: dùng trong classifier 2 class |
| optimizers | dùng để chọn thuật toán training |
| plot\_model | Giúp phác thảo cấu trúc model bằng hình ảnh ( biểu đồ ) |
| evaluate | để tính toán độ chính xác của model |
| compile | Sau khi build model xong thì compile nó có tác dụng biên tập lại toàn bộ model của chúng ta đã build. Ở đây chúng ta có thể chọn các tham số để training model như : thuật toán training thông qua tham số optimizer, function loss của model chúng ta có thể sử dụng mặc định hoặc tự build thông qua tham số loss, chọn metrics hiện thị khi model được training |

# CHƯƠNG 1: TÌM HIỂU VỀ ĐẠO VĂN

## Đạo văn là gì?

Đạo văn với mức độ nghiên cứu khoa học của sinh viên có thể được hiểu là sử dụng công trình hay tác phẩm của người khác, lấy ý tưởng của người khác, sao chép nguyên bản từ ngữ của người khác mà không ghi nguồn, sử dụng cấu trúc và cách lí giải của người khác mà không ghi nhận họ, và lấy những thông tin chuyên ngành mà không đề rõ nguồn gốc.

Đạo văn được xem là hành vi thiếu trung thực về mặt học thuật và vi phạm đạo đức rất nghiêm trọng. Ở cấp độ sinh viên, đạo văn sẽ khiến kết quả nghiên cứu bị huỷ bỏ tuỳ thuộc vào mức độ nghiêm trọng của hành vi. Ở cấp độ nghiên cứu chuyên nghiệp, người đạo văn có thể bị buộc thôi việc, thu hồi công trình đã công bố hoặc huỷ bỏ chức danh.

## Những điều nên làm

* Ghi rõ nguồn khi tham khảo tài liệu từ nguồn bên ngoài.
* Tôn trọng những sản phẩm, nội dung, công trình của người khác khi sử dụng.
* Nếu tài liệu muốn sử dụng quan trọng, cần xin ý kiến của chủ sử hữu trước khi sử dụng

## Những điều không nên làm

* Sao chép công trình, tác phẩm mà không ghi rõ nguồn.
* Lấy ý tưởng của người khác để sử dụng cho mình.
* Sao chép cấu trúc và cách lý giải của người khác.

## Lời cam đoan

Chúng em xin cam đoan đồ án này do chính chúng em thực hiện. Chúng em không sao chép, sử dụng bất kỳ tài liệu, mã nguồn của người khác mà không ghi rõ nguồn gốc. Chúng em xin chịu hoàn toàn trách nhiệm nếu vi phạm.

*Thành phố Hồ Chí Minh, ngày 19 tháng 05 năm 2019*

*Sinh viên thực hiện*

Thuy Bao

Đào Xuân Thủy Lâm Phước Bảo

# CHƯƠNG 2: MULTIVARIATE LINEAR REGRESSION

## Đôi nét lịch sử

Multivariate linear regression được trình bày bởi Legendre (1805) và Gauss (1809).

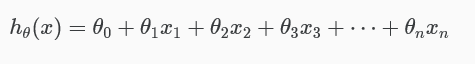
## Ứng dụng

Ứng dụng của Multivariate linear regression tương tự như ứng dụng của linear regression dùng để dự đoán các giá trị đầu ra có tính liên tục như dự đoán xu hướng, dự đoán về tài chính, kinh tế, nhà đất... Với MLR ta có thể đưa vào nhiều feature hơn dẫn đến độ chính xác tăng cao hơn.

## Mô tả thuật toán

**Định nghĩa:** Multivariate linear regression (MLR) là một dạng của Linear regression với dữ liệu đầu vào có nhiều feature.

Công thức: có dạng là một hàm hypothesis với nhiều tham số là các feature x:



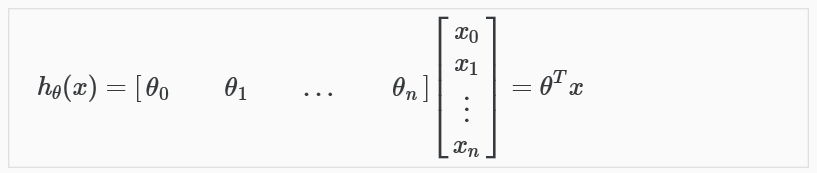
Trong đó: xj(i) là giá trị của feature j thứ i

x(i) là giá trị input (feature) thứ i

m là số lượng data traning

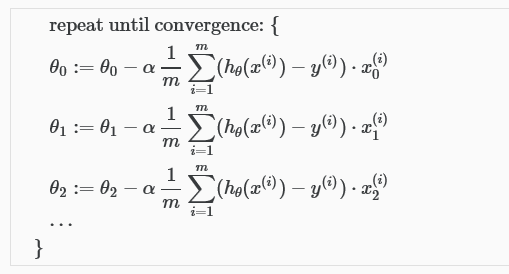
n là số lượng feature

Sử dụng định nghĩa của phép nhân ma trận, hàm hypothesis của MLR sẽ được biểu diễn như sau:



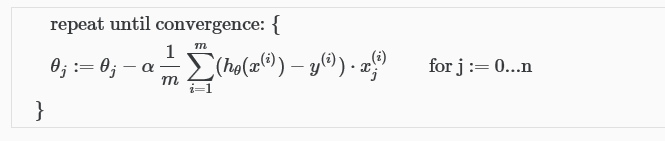
**Gradient Descent:**

Cũng tương tự như Gradient Descent trong Linear Regression. Trong Multivariate linear regression có dạng:



Chỉ cần lập lại “n” features

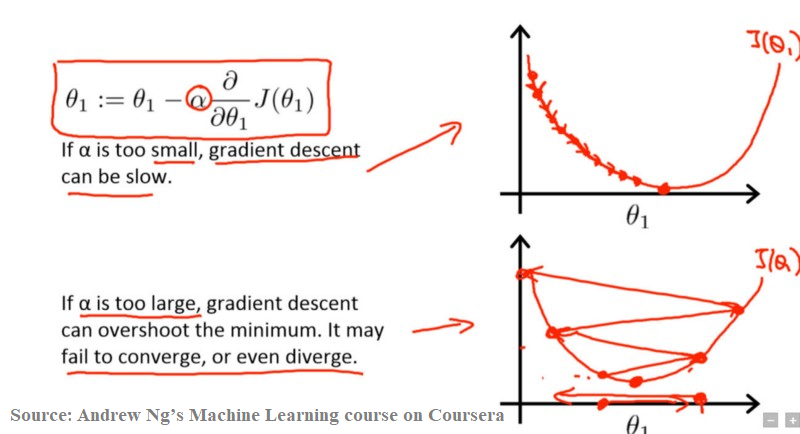
Một cách khác



**Feature Scaling**

Chúng ta có thể tăng tốc cho quá trình Gradient Descent bằng cách thay đổi giá trị input sắp xỉ cùng chung một khoảng nhỏ lại. Đó là bởi vì các giá trị θ sẽ giảm nhanh trong khoảng số nhỏ lại. Thông thường thì các giá trị hay nằm trong các khoảng -1 ≤ x(i) ≤ 1 hoặc -0.5 ≤ x(i) ≤ 0.5

Có 2 kĩ thuật để sử dụng đó là: feature scaling và mean normalization

**Learning Rate** (tốc độ học):

Hình 1. Biểu diễn tốc độ học

Nhìn vào trong hình ta có thể thấy ở hình đầu tiên khi trong quá trình lập của Gradient descent nếu hệ số alpha của learning rate quá bé sẽ làm chậm tốc độ đi về điểm min. Tuy nhiên với trường hợp ngước lại nếu hệ số learning rate lớn có thể làm cho hàm đi mãi mà khong bao giờ hội tụ.

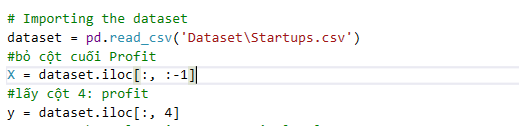
Vì vậy mà lựa chọn hệ số cho learning rate vô cùng quan trọng. thông thường learning rate thường chọn nhỏ.

## Lập trình thuật toán

**Bước 1: Load dữ liệu và xử lí dữ liệu**

Dữ liệu được load có tên Startups.csv, gồm 5 cột:

* R&D spend: chi phí R&D
* Adminstration: chi phí quản trị
* Marketing spend: chi phí marketing
* State: tên bang. Trong dataset chỉ bao gồm 3 bang: New York, California và Florida
* Profit: Lợi nhuận



Ta lấy được data train gán vào X, và label gán vào y.

Tuy nhiên vì dữ liệu ở cột State là chữ nên chúng ta tiến hành chuyển đổi về dạng số



Sau khi xử lí xong cột State về dạng số ta tiến hành gán lại vào data training là X



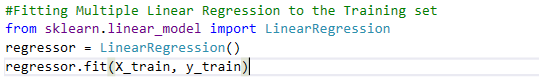
Lúc này trong X sẽ có 5 feature bao gồm: R&D spend, admin, Marketing spend, florida và new york

**Bước 2: Tiến hành tách dữ liệu training và test**

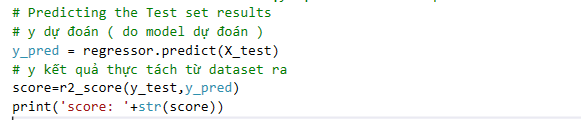


Dữ liệu đã được tách ra thành train và test từ X và y lúc nãy. Với hàm phân chia ngẫu nhiên và tập dữ liệu test chỉ 20% trong X

**Bước 3: Traing**



**Bước 4: Đánh giá Model**

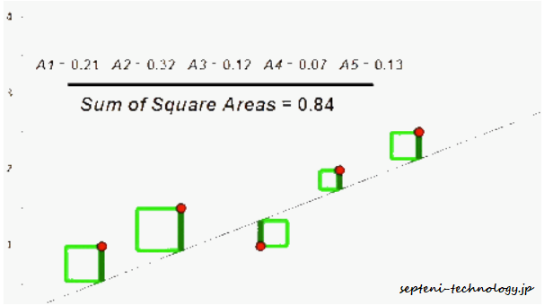


Sử dụng dữ liệu X test lúc nãy đã tách là input cho hàm Predict của model. Sau khi model đã dự đoán xong ta sẽ lấy kết quả đó đi so sánh với label test đã lấy ra lúc trước từ đó mà thu được độ chính xác của model đã train

# CHƯƠNG 3: POLYNOMIAL REGRESSION

## Đặt vấn đề

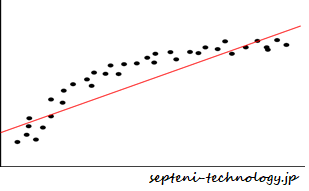
Để biểu diễn một đồ thị học được, Linear Regression sử dụng hàm số tuyến tính (được biểu diễn dưới dạng một đường thẳng trên đồ thị) làm hàm tiên đoán (ký hiệu h(x)).

Hình . Mô hình Linear Regression

Từ đó, sai số của h(x) được tính toán dựa trên bình phương sai khác giữa h(x) và y thực tế như hình sau:

Hình . Sai số thuật toán Linear Regression

Nhưng nếu ta áp dụng mô hình Linear Regression trong trường hợp sau:

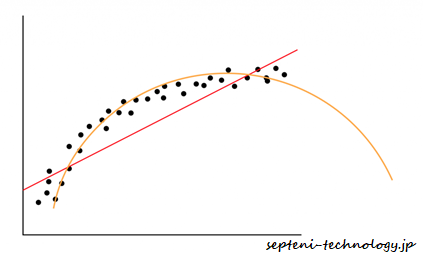


Hình . Vấn đề với Linear Regression

Trong trường hợp này dữ liệu không tập trung quanh đường thẳng h(x) mà có hơi hướng theo dạng đường cong hơn. Chính vì thế, khi sử dụng hàm h(x) này, kết quả dự đoán sẽ không tốt. Khi mô hình không đáp ứng được xu hướng biến thiên của dự liệu, ta sẽ gặp phải vấn đề như trên, chúng được gọi là “underfitting”. Quá trình ta cần làm thực chất là điều chỉnh hàm số phù hợp với sự biến thiên của dữ liệu, nên khi hàm số không biến thiên theo xu hướng của dữ liệu thì mô hình sẽ không khớp với dữ liệu. Khi gặp vấn đề này chúng ta cần hàm số phức tạp hơn để hàm tiên đoán h(x) đạt hiệu quả tốt hơn.

## Ứng dụng

Để giải quyết bài toán trên, thay vì dùng đường thẳng để mô tả sự biến thiên của dữ liệu, ta sẽ tăng bậc của phương trình để giải quyết hiện tượng “underfitting”. Đồ thị sẽ có dạng bậc cao hơn và phức tạp hơn, để mô tả chính xác hơn sự biến thiên của dữ liệu và “cost function” ta thu được sẽ nhỏ hơn. Từ đó, ta thu được một mô hình mô tả dữ liệu chính xác hơn. Ta gọi đó là thuật toán Polynomial Regression.



Hình . So sánh Linear Regression với Polynomial Regression

## Mô tả thuật toán

### 3.1 Mô tả dữ liệu

Từ thực tế, ta có thể biểu diễn mô hình dữ liệu dưới dạng như sau:

Bảng . Bảng mô tả dữ liệu cho Polynomial Regression với dữ liệu đơn chiều

|  |  |
| --- | --- |
| **Dữ liệu đầu vào** | **Kết quả đạt được** |
| x1 | y1 |
| x2 | y2 |
| x3 | y3 |
| x4 | y4 |
| … | … |

Bảng . Bảng mô tả dữ liệu cho Polynomial Regression với dữ liệu đa chiều

|  |  |  |
| --- | --- | --- |
| **Dữ liệu đầu vào** | | **Kết quả đạt được** |
| x11 | x21 | y1 |
| x12 | x22 | y2 |
| x13 | x23 | y3 |
| x14 | x24 | y4 |
| … | … | … |

### 3.2 Phương trình biểu diễn mô hình của thuật toán

Từ bảng mô tả dữ liệu trên, để xây dựng mô hình mô tả sự biến thiên của dữ liệu loại tăng dần sử dụng Polynomial Regression, ta có phương trình dưới dạng:

* + - Dữ liệu một chiều:

hθ(x) = θ0 + θ1x1 + θ2x12

* + - Dữ liệu nhiều chiều:

hθ(x) = θ0 + θ1x1 + θ2 x2 + θ3 x12 x2 + θ4 x1x22+ …

Bước tiếp theo, biến đối phương trình biểu diễn mô hình Polynomial Regression về phương trình có dạng của thuật toán Linear Regression:

* + - Đặt các thành phần biến trong phương trình thành biến mới.
    - Viết lại phương trình tương đương mới có dạng của phương trình đường thẳng.
    - Phương trình tổng quát sau khi biến đổi có dạng:

hθ(x) = θ0 + θ1x1 + θ2x2 + θ3x3 + θ4x4 + θ5x5 + θ6x6 + … + θnxn

n: số lượng phần tử trong phương trình ban đầu

### 3.3 Cách tính và đánh giá hiệu năng của mô hình

Với dữ liệu như trên, ta có thể mô tả dữ liệu theo phương trình biểu diễn như sau:

|  |  |
| --- | --- |
|  | với m là số dòng dữ liệu trong dataset, n là số phần tử trong phương trình biểu diễn mô hình. |
| X y |  |

Để tìm những tham số của phương trình, ta có nhiều cách:

Cách 1: Normal Equation

Với ma trận trên, ta có phương trình:

X. = y XT.X. = XT. y

(XT.X)-1. XT. X. = (XT.X)-1. XT. y

= (XT.X)-1. XT. y

Cách 2: Gradient descent

|  |  |
| --- | --- |
| Lặp lại tới khi hội tụ  {    } | j = : m là số lượng trong công thức  *:* learning rate |

Một số thuật toán khác như: Conjugate gradient, BFGS, L\_BFGS…

Để đánh giá hiệu năng của thuật toán, ta sử dụng cost function:

J() =

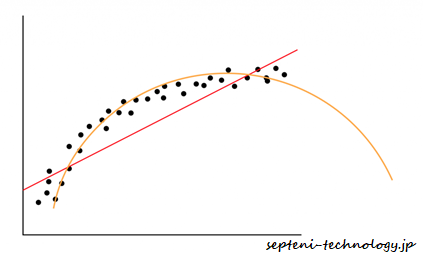
n: số lượng dòng dữ liệu trong dataset

: kết quả do thuật toán tính ra dựa trên dữ liệu đầu vào tại dòng thứ i

: kết quả thực tế trong dữ liệu đầu vào tại dòng thứ i

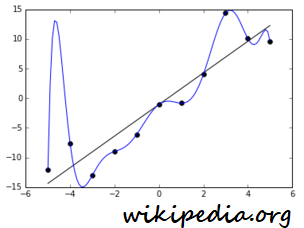
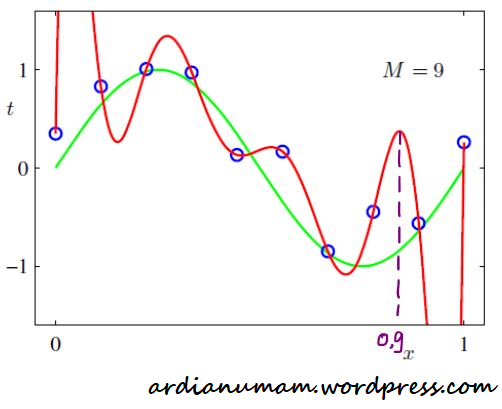
## So sánh với các thuật toán khác

Như vậy, ta có thể thấy thuật Polynomial Regression là biểu đồ mô tả sự biến thiên dữ liệu tốt hơn so với Linear Regression trong trường hợp này. Từ đó cost function với thuật toán này sẽ thấp hơn và sự chính xác cao hơn.



Hình . Kết quả so sánh Polynomial Regression và Linear Regression

Tuy vậy, không phải khi nào Polynomial Regression cũng tốt hơn Linear Regression. Với bài toán có dữ liệu biến thiên phức tạp, đòi hỏi phương trình có bậc cao hơn để mô tả. Nhưng bên cạnh đó, một số dữ liệu biến thiên tuyến tính, khi đó Linear Regression đủ để mô tả khối dữ liệu đó, nếu ta dùng Polynomial Regression, bậc của phương trình cao hơn, ta sẽ gặp hiện tượng overfitting. Hoặc ta áp dụng bậc cho phương trình cao hơn so với bậc cần tìm, ta cũng gây hiện tượng overfitting Hiện tượng này sẽ tìm ra cost function nhỏ, nhưng khi áp dụng vào dữ liệu kiểm tra, sai số sẽ rất lớn.

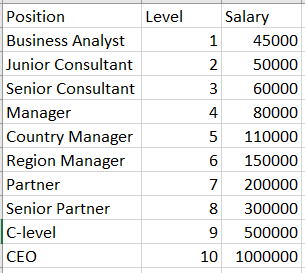
 

Hình . Overfitting với Linear Regression  Hình . Overfitting với Polynomial Regression

## Lập trình thuật toán

### 5.1 Bộ dữ liệu cho thuật toán

* Chủ đề: Dự đoán lương của nhân viên dựa theo cấp bậc vị trí.
* Địa chỉ tải về: <https://www.kaggle.com/testpython/polynomial-position-salary-data>.
* Số dòng dữ liệu cho thuật toán: 10.



Hình . Mô tả dữ liệu mẫu cho thuật toán Polynomial Regression

### 5.2 Mục tiêu thuật toán

Dự đoán lương của nhân viên dựa theo cấp bậc vị trí của họ trong công ty sử dụng Polynomial Regression và so sánh mức độ hiểu quả với thuật toán Lineal Regresison.

### 5.3 Thư viện sử dụng

* Scikit-learn (Phụ lục - Mục 1).
* Pandas (Phụ lục - Mục 2).
* Numpy (Phụ lục - Mục 3).
* Matplotlib (Phụ lục - Mục 4).
* OS (Phụ lục - Mục 5).

### 5.3 Quá trình thực hiện

Bước 1: Khai báo nơi lưu trữ dataset.

import os

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

ROOT\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))

CONFIG\_PATH = os.path.join(ROOT\_DIR, 'datasets')

data = pd.read\_csv(CONFIG\_PATH + '/Position\_Salaries.csv')

Bước 2: Khai báo thứ tự dữ liệu đầu vào X là cột 1 và 2, dữ liệu đầu ra cần đạt được là cột cuối cùng (cột 3).

X = data.iloc[:,1:2].values

y = data.iloc[:,-1].values

Bước 3: Căn giữa dữ liệu về giá trị trung bình bằng 0 và biến đổi dữ liệu đầu vào về dạng Polynomial, khai báo hai kiểu model có dạng Linear Regression và truyền, tinh chỉnh bộ dữ liệu đầu vào cho hai model.

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 2)

X\_poly = poly\_reg.fit\_transform(X)

from sklearn.linear\_model import LinearRegression

linear\_reg\_model = LinearRegression()

linear\_reg\_model.fit(X, y)

poly\_reg\_model = LinearRegression()

poly\_reg\_model.fit(X\_poly, y)

Bước 4: Sử dụng thư viện Matplotlib vẽ hai sơ đồ học được bằng hai thuật toán Linear và Polynimial cho người dùng, đồng thời chấm các điểm cho kết quả thực tế từ dữ liệu đầu vào lên sơ đồ.

import matplotlib.pyplot as plt

plt.scatter(X, y, color='red', label='Actual observation points')

plt.plot(X, linear\_reg\_model.predict(X), label='Linear regressor fit curve')

plt.plot(X, poly\_reg\_model.predict(poly\_reg.fit\_transform(X)), label='Polynmial regressor fit line')

plt.title('Truth or bluff (Linear Regression)')

plt.xlabel('Position Level')

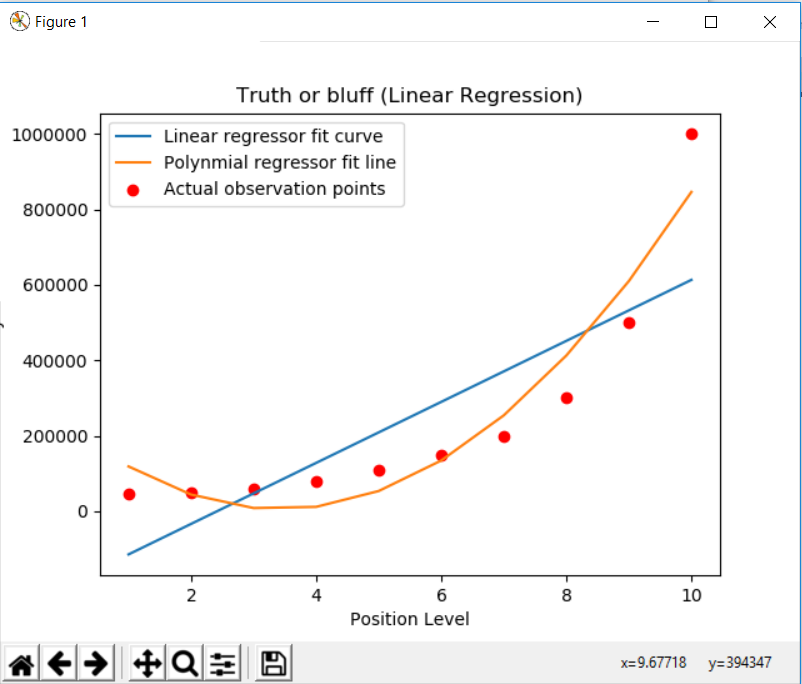
plt.ylabel('Salary')

plt.legend()

plt.show()

### 5.4 Kết quả thu được

Kết quả thu được được mô tả như hình dưới đây:



Hình 10. Kết quả thu được sau khi chạy thuật toán Polynomial Regression

Như ta đã thấy, nếu sử dụng thuật toán Linear Regression, sơ đồ học ra sẽ cho cost function rất lớn và không thể mô tả đúng được sự biến thiên của dữ liệu. Nếu dùng thuật toán Polynomial Regression, sơ đồ học ra sẽ biến thiên theo đường parabol và mô tả chính xác hơn sự biến thiên của dữ liệu.

# CHƯƠNG 4: LOGISTIC REGRESSION

## Đặt vấn đề

Như ở những thuật toán trên, ta có thể thấy dữ liệu đầu ra ta muốn đạt được là liên tục. Vậy với những kiểu giá trị rời rạc thì phải làm như thế nào?

Với những kiểu dữ liệu mà giá trị dự đoán đầu ra như sau:

* Dự đoán một email có phải là email spam hay không. Kết quả đầu ra là có hoặc không.
* Phân loại email thuộc nhóm e mail nào. Kết quả đầu ra là nhóm bạn, nhóm công việc, nhóm học tập, nhóm quảng cáo.
* Phân nhóm ảnh. Kết quả đầu ra có thể là nhóm động vật, cây cối hoặc nhóm con người.

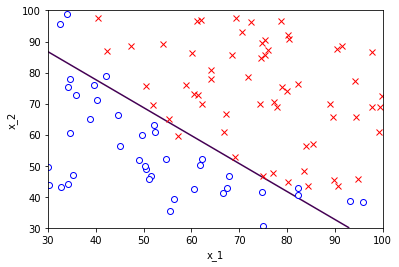
Ở những ví dụ trên, ta thấy kết quả mong muốn của ta không phải là một giá trị liên tục mà đó là một nhóm giá trị rời rạc. Vậy để giải quyết bài toán này, ta sẽ sử dụng thuật toán Logistic Regression.

## Ứng dụng

Mô hình hồi quy Logistic sẽ dùng hàm biến đổi logarit cho biến đầu ra để biến mô hình từ quan hệ phi tuyến tính sang tuyến tính. Nói một cách khác, nó biểu diễn quan hệ hồi quy tuyến tính dưới dạng hàm logarit, nên đôi khi nó cũng được gọi là Logit Regression [2].

Mô hình Logistic có một giả định rằng biến phụ thuộc (dự đoán) có giá trị rời rạc. Nếu biến dự đoán chỉ lấy hai giá trị rời rạc, đó là mô hình Binary Logistic Regression. Nếu biến dự đoán lấy nhiều hơn hai giá trị, đó là mô hình Multinomial Logistic Regression.

Ví dụ, xem một bức ảnh có phải là ảnh chân dung hay không. Thì ở đây ta coi đầu ra y = 1 nếu bước ảnh có mặt con người và y = 0 nếu bức ảnh không có mặt người nào. Đầu vào x ở đây sẽ là các pixel một bức ảnh đầu vào.

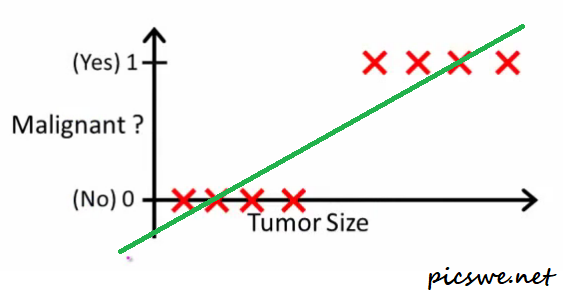


Hình 11. Ví dụ về Logistic Regression

## Mô tả thuật toán

Đầu tiên, ta dùng Linear Regression để dự đoán y khi biết x. Ở đây ta bỏ qua điều kiện giá trị của y là rời rạc (0 hoặc 1). Ở Linear Regression ta có hàm dự đoán:

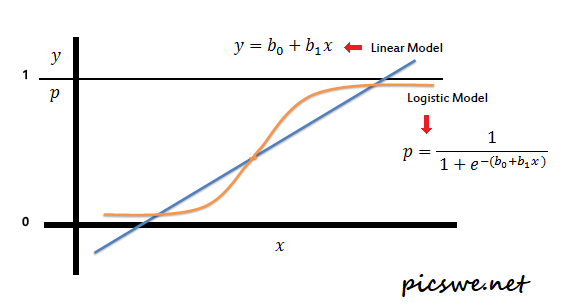
Sau khi thực hiện chạy dữ liệu dưới thuật toán Linear Regression, ta được mô hình có dạng như sau:



Hình 12. Mô hình Linear Regression cho dữ liệu rời rạc

Vì giá trị đầu ra là rời rạc, nên ta giới hạn y trong đoạn [0, 1]. Do vậy, ta sử dụng kết hợp sigmoid function (logistic function) , trong đó:

Tóm lại, ta có hàm dự đoán với đầu vào *x* như sau:



Hình 13. Áp dụng Sigmoid function cho thuật toán Logistic Regression

Nếu ban đầu, ta chỉnh sửa dữ liệu với giá trị trung bình là 0, ta sẽ được mô hình có trục Oy nằm giữa, từ đó dựa trên hàm sigmoid, ta có thể dễ dàng đoán dữ liệu đầu vào sẽ thuộc nhóm nào, cụ thể nếu kết quả tính được nhỏ hơn 0.5, dữ liệu đó sẽ thuộc nhóm 0, và nếu lớn hơn 0.5, dữ liệu đó sẽ thuộc nhóm 1.



Hình 14. Đường thẳng sigmoid khi được tinh chỉnh dữ liệu

Để tính cost function, ta sử dụng hàm sau:

Để tìm giúp hàm có giá trị nhỏ nhất, ta cần tìm điểm cực tiểu của phương trình , ta có thể sử dụng Gradient descent:

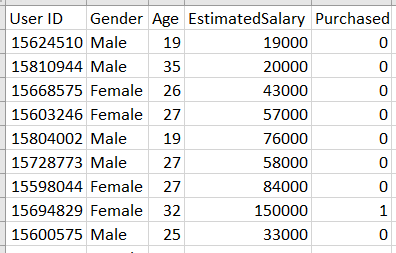
|  |  |
| --- | --- |
| Lặp lại tới khi hội tụ  {    } | j = : m là số lượng trong công thức  *:* learning rate |

Một số phương pháp khác như: Newton-Raphson, Conjugate gradient, BFGS, L\_BFGS…

## Lập trình thuật toán

### 4.1 Bộ dữ liệu cho thuật toán

* Chủ đề: Lịch sử thanh toán cho quảng cáo của người dùng Facebook dựa trên độ tuổi và mức lương ước tính.
* Địa chỉ tải về: https://www.kaggle.com/rakeshrau/social-network-ads.
* Số dòng dữ liệu cho thuật toán: 400.



Hình . Mô tả dữ liệu mẫu cho thuật toán Logistic Regression

### 4.2 Mục tiêu thuật toán

Dự đoán quyết định thanh toán cho dịch vụ quảng cáo Facebook dựa trên độ tuổi và mức lương ước tính từ người dùng.

### 4.3 Thư viện sử dụng

* Scikit-learn (Phụ lục - Mục 1).
* Pandas (Phụ lục - Mục 2).
* Numpy (Phụ lục - Mục 3).
* Matplotlib (Phụ lục - Mục 4).
* OS (Phụ lục - Mục 5).

### 5.3 Quá trình thực hiện

Bước 1: Khai báo nơi lưu trữ dataset và bỏ cột User ID và cột Gender

ROOT\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))

CONFIG\_PATH = os.path.join(ROOT\_DIR, 'datasets')

print(os.listdir(CONFIG\_PATH))

data = pd.read\_csv(CONFIG\_PATH + '/Social\_Network\_Ads.csv')

#print(data)

data.drop(columns=['User ID','Gender',],axis=1,inplace=True)

data.head()

Bước 2: Khai báo thứ tự dữ liệu đầu vào X là tất cả các cột trừ cột cuối (cột 1 và 2), dữ liệu đầu ra cần đạt được là cột cuối cùng (cột 3).

y = data.iloc[:,-1].values

X = data.iloc[:,:-1].values

Bước 3: Sử dụng thư việc scikit-learn lấy 25% dữ liệu làm dữ liệu kiểm tra và chỉnh sửa lại dữ liệu để học về giá trị trung bình là 0, sau đó chuyển đổi cả 2 gói dữ liệu.

# Phân tách dữ liệu

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# Phân tách dữ liệu

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

Bước 4: Chỉnh sửa lại gói dữ liệu đầu vào và tính hiệu năng của thuật toán.

from sklearn.linear\_model import LogisticRegression

classifierLR = LogisticRegression()

classifierLR.fit(X\_train, y\_train)

from sklearn.metrics import accuracy\_score

y\_predLR=classifierLR.predict(X\_test)

print(accuracy\_score(y\_test, y\_predLR))

Bước 4: Sử dụng thư viện Matplotlib vẽ sơ đồ học được bằng thuật toán Logistic Regression cho người dùng xem, đồng thời chấm các điểm cho kết quả thực tế từ dữ liệu đầu vào lên sơ đồ.

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifierLR.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Logistic Regression (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

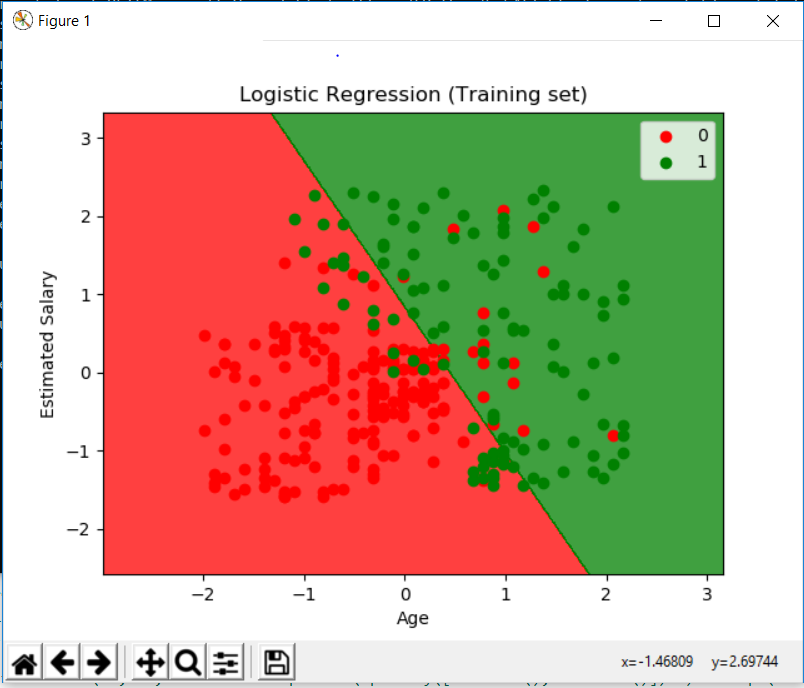
plt.legend()

plt.show()

### 5.4 Kết quả thu được

Thuật toán chạy với hiệu suất đạt 89%.

Kết quả thu được được mô tả như hình dưới đây:



Hình 16. Kết quả thu được sau khi chạy thuật toán Logistic Regression

Như ta đã thấy, thuật toán đã học với độ chính xác 89%, và đã chia dataset thành hai khu vực.

# CHƯƠNG 5: NEURAL NETWORKS

## Đôi nét lịch sử

Vào năm 1957, Frank Rosenblatt đã thiết kế mạng nơron (neural network) đầu tiên cho máy tính, trong đó mô phỏng quá trình suy nghĩ của bộ não con người.

Vào năm 2011, 2011 - Google Brain đã được phát triển, và mạng deep nơron (deep neural network) của nó có thể học để phát hiện và phân loại nhiều đối tượng theo cách mà một con mèo thực hiện. [3]

## Ứng dụng

Mạng nơ ron là một tập các thuật toán, được mô hình hóa theo não bộ con người, được tạo ra để nhận diện các khuôn mẫu, hình mẫu của tập dữ liệu. Nó cố gắng hiểu ý nghĩa của từng dòng dữ liệu thông qua các thuật toán tạo nên sự nhận thức cho máy móc bằng những loại dữ liệu đã được dán nhãn cho đến những dữ liệu thô chưa được xác định. Kết quả mà mạng nơ ron có thể nhận diện ra chính là những con số, bao gồm các vectors và dĩ nhiên những con số đó đều được ánh xạ đến dữ liệu hiện thực của con người như là hình ảnh, âm thanh, đoạn văn, thời gian.

Mạng nơ ron giúp ta gom nhóm và phân loại. Chúng ta có thể tưởng tượng rằng chúng là những cố máy có linh hồn và nhận thức được kết quả mà chúng ta mong muốn sau khi đã dạy chúng học thông qua hàng loạt các dữ liệu mẫu.

## Mô tả thuật toán

Hình 17. Mạng nơ ron cơ bản

Các thành ph//ần trong mạng Nơ ron:

Các lớp:

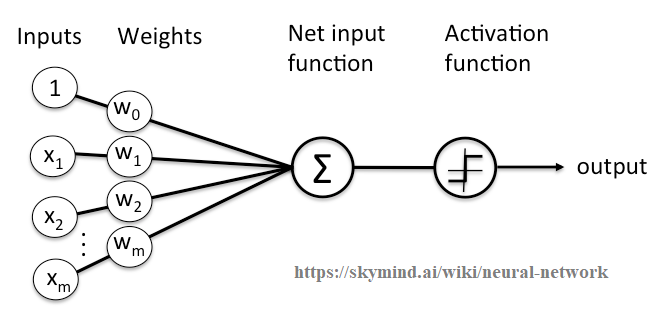
Input layer: Lớp đầu tiên trong mạng nơ ron, thông thường số unit của lớp chính là số lượng feature chúng ta đưa vào.

Hidden layer: Lớp nằm giữa Input layer và Output Layer, có thể có nhiều Hidden layer với số unit khác nhau tùy độ phức tạp của bài toán và dữ liệu cần xử lí. Nơi đây tiếp nhận input từ các Node (ví dụ input layer) và tiến hành xử lí tính toán.

Output layer: Lớp cuối cùng trong mạng nơ ron, nơi đây tiếp nhận input từ các output đã được xử lí ở tầng hidden layer và tiến hành tính toán lần cuối để cho ra kết quả. Số lượng unit ở tầng này phụ thuộc vào số lượng label mà chúng ta muốn.

Units: Một node hình tròn trong một layer được gọi là một unit. Unit ở các input layer, hidden layers, và output layer được lần lượt gọi là input unit, hidden unit, và output unit. Đầu vào của các hidden layer được ký hiệu bởi **z**, đầu ra của mỗi unit thường được ký hiệu là **a** (thể hiện activation, tức giá trị của mỗi unit sau khi ta áp dụng activation function lên **z**). Đầu ra của unit thứ **i** trong layer thứ l được kí hiệu **ai(l)**

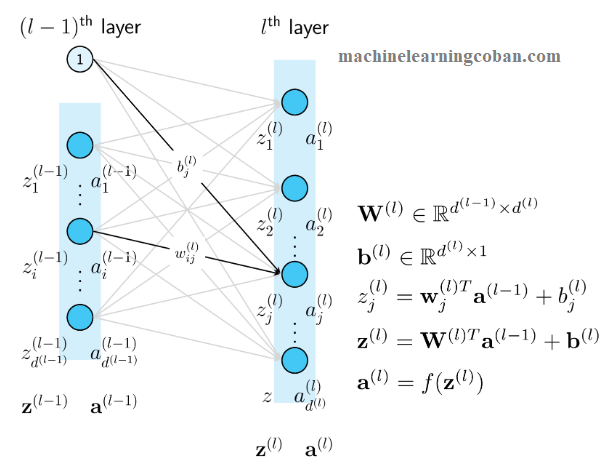
Giả sử thêm rằng số unit trong layer thứ l (không tính bias) là d(l). Vector biểu diễn output layer thứ l được kí hiệu là a(l) thuộc Rd(l)

Activation functions

Hình 18. Các ký hiệu sử trong mô tả Neural Network

Mỗi output của một unit (trừ các input units) được tính dựa vào công thức:





Hình 19. Sơ đồ neural network

Trong đó f(.) là một (nonlinear) activation function. Ở dạng vector, biểu thức bên trên được viết là:



Khi activation function f(.) được áp dụng cho một ma trận (hoặc vector), ta hiểu rằng nó được áp dụng cho từng thành phần của ma trận đó. Sau đó các thành phần này được sắp xếp lại đúng theo thứ tự để được một ma trận có kích thước bằng với ma trận input. Trong tiếngAnh, việc áp dụng lên từng phần tử như thế này được gọi là element-wise.

**Backpropagation**

Lan truyền ngược (backpropagation) là giải thuật cốt lõi giúp cho các mô hình học sâu có thể dễ dàng thực thi tính toán được. Với các mạng NN hiện đại, nhờ giải thuật này mà thuật toán tối ưu với đạo hàm (gradient descent) có thể nhanh hơn hàng triệu lần so với cách thực hiện truyền thống.

**Forward propagation**

Như bạn thấy thì tất cả các nốt mạng (nơ-ron) được kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào tới tầng ra. Tức là mỗi nốt ở một tầng nào đó sẽ nhận đầu vào là tất cả các nốt ở tầng trước đó mà không suy luận ngược lại.

## Cài đặt thuật toán

Trong phần trình bày cài đặt thuật toán Neural Networks của nhóm em, nhóm em sẽ sử dụng thư viện Keras để hỗ trợ quá trình cài đặt

Chúng ta sẽ đi qua các bước sau để tiển hành cài đặt thuật toán:

* Bước 1: Load dữ liệu
* Bước 2: Thiết lập Model
* Bước 3: Compile Model
* Bước 4: Train Model
* Bước 5: Đánh giá Model
* Bước 6: Dự đoán dữ liệu

Load dữ liệu

Trong phần project của nhóm, nhóm sẽ sử dụng dataset chuẩn đoán bệnh tim được lấy từ Kaggle

<https://www.kaggle.com/ronitf/heart-disease-uci>

Phân tích về tập dữ liệu :

Gồm 13 cột đại diện cho từng đặc điểm của người bệnh:

> 1. Age: tuổi

> 2. Sex: giới tính

> 3. chest pain: vết thương ngực

> 4. resting blood pressure : huyết áp

> 5. serum cholestoral in mg/dl : Nồng độ cholestoral trong máu

> 6. fasting blood sugar > 120 mg/dl: lượng đường huyết

> 7. resting electrocardiographic results (values 0,1,2): kết quả điện tâm đồ

> 8. maximum heart rate achieved: Nhịp tim lớn nhất

> 9. exercise induced angina :Chứng đau thắt ngực

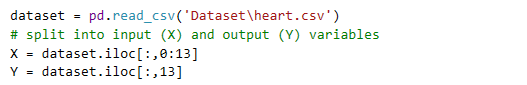
> 10. oldpeak = ST depression induced by exercise relative to rest

> 11. the slope of the peak exercise ST segment

> 12. number of major vessels (0-3) colored by flourosopy

> 13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

Cột cuối là Target có giá trị là 0 (không có bệnh) và 1 (có mầm móng bệnh ung thư)

Tiếp theo là load dữ liệu

Ở đây sử dụng thư viện Pandas để load dữ liệu

Các dữ liệu của feature được lấy từ 0 đến cột thứ 13 và riêng label được lấy ở cột cuối.

Thiết lập Model

Keras có 2 cách sử dụng để thiết lập Model đó chính là sử dụng Sequential Model và API Model. Với Project của chúng em đã sử dụng Sequential Model

Model trong Keras được thiết lập bởi những layers nối tiếp nhau. Vì thế chúng em tạo một Model tuần tự và thêm vào từng lớp cho đến khi chúng ta cảm thấy hài lòng.

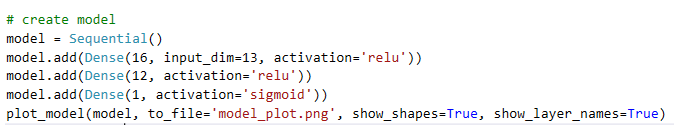
Đầu tiên phải chắc chắn rằng lớp input layers khớp với số lượng feature ta đưa vào.

Trong trường hợp này, chúng em khởi tạo những network weight là những số nhỏ ngẫn nhiên từ 0 đến 0.05 bởi vì đây là mặc định của Keras. Một cách khác là có thể khởi tạo từ Gaussian.

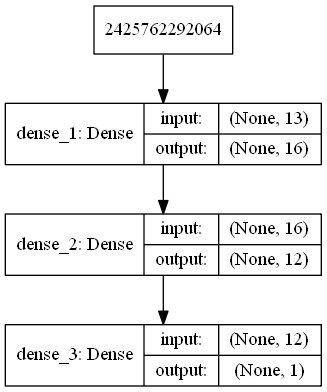
Chúng em sử dụng activation ‘relu’ cho 2 layers đầu tiên và hàm sigmoid cho layer output. Bởi vì để cải thiện tốt hơn về mặt hiệu suất

Sigmoid ở output layer sẽ chắc sẵn rằng ouput của chúng ta sẽ nằm trong khoảng từ 0 – 1 và dễ dàng để ánh xạ tới kết quả trong thực tế hơn đơn gian là vì đồ thị của hàm sigmoid chỉ biến thiên từ 0 – 1

Dưới đây là đoạn code xây dựng Model:



Và được thể hiện dưới dạng hình ảnh sau:



Hình 20. Sơ đồ hiện thực Model

Compile Model

Quá trình compile sẽ lựa chọn cách tốt nhất để thể hiện mạng lưới neuron để trainning dữ liệu và thực hiện predcit dữ liệu.

Trong quá trình compile chúng ta cần xác định vài thuộc tính sử dụng cho quá trình training.

Ở đây, chúng em sử dụng loss function có tên “binary\_crossentropy” và sử dụng giải thuật gradient descent có tên “adam”. Cuối cùng, do mục đính chúng ta là phân loại dữ liệu nên việc thu thập và báo cáo lại độ chính xác trong mỗi lần traning tập dữ liệu là điều cần thiết vì thế chúng em sử dụng metrics= ”accuracy”

Train Model

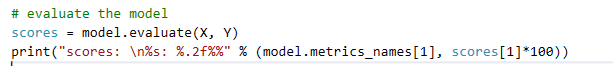


Bắt đầu chạy dữ liệu trên model chúng ta đã thiết lập, ở đây có một số qui định cho quá trình training như sau:

Epochs=300: số lần chạy hết qua dữ liệu là 300 lần

Batch-size: giới hạn dữ liệu chạy 1 lần là 20

**Đánh giá Model**

Sau khi Model đã được train chúng ta tiến hành đánh giá hiệu suất của mạng nơ ron chúng ta vừa mới tạo ra trên chính tâp dữ liệu chúng ta sử dụng để trainning

Kết quả in ra score càng gần với 1 chứng tỏ model chúng ta đã train là tốt

Dự đoán dữ liệu

Để dự đoán dữ liệu chỉ cần sử dụng hàm Predict



Tuy nhiên khi dự đoán dữ liệu sẽ có kết quả là một dãy số trong khoảng [0,1]

Vì đây là bài toán Binary Classification nên đầu ra của dự đoán chỉ có thể là 1 hoặc 0 vì thế ở đây chúng ta sử dụng hàm Roud để làm tròn dữ liệu.

# BẢNG PHÂN CÔNG CÔNG VIỆC

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | Công việc |
| 1 | Đào Xuân Thủy | - Tìm hiểu và viết chương trình ví dụ và báo cáo về Polynomial Regression.  - Tìm hiểu và viết chương trình ví dụ và báo cáo về Logistic Regression. |
| 2 | Lâm Phước Bảo | - Tìm hiểu và viết chương trình ví dụ và báo cáo về Multivariate Linear Regression.  - Tìm hiểu và viết chương trình ví dụ và báo cáo về Newral Networks. |

# PHỤ LỤC

## Scikit-learn

Scikit-learn (trước đây là scikits.learn ) là một thư viện học máy miễn phí cho ngôn ngữ lập trình Python. Nó có các thuật toán phân loại, hồi quy và phân cụm khác nhau bao gồm hỗ trợ vectơ, random forests, gradient boosting, k-means và DBSCAN, và được thiết kế để tương tác với các thư viện khoa học và số học Python NumPy và SciPy. [1]

## Pandas

Trong lập trình máy tính, pandas là một thư viện phần mềm được viết cho ngôn ngữ lập trình Python để thao tác và phân tích dữ liệu. Cụ thể, nó cung cấp các cấu trúc dữ liệu và các thao tác để thao tác trên mảng các số và chuỗi thời gian. Đây là phần mềm miễn phí được phát hành theo giấy phép BSD. Tên được lấy từ thuật ngữ "panel data", thuật ngữ kinh tế lượng cho các tập dữ liệu bao gồm việc quan sát một cá thể trong nhiều khoảng thời gian. [7]

## Numpy

NumPy là một thư viện cho ngôn ngữ lập trình Python, hỗ trợ tính toán cho các mảng và ma trận lớn, đa chiều, cùng với một tập hợp lớn các hàm toán học cấp cao để tính toán trên các mảng này. Tổ tiên của NumPy là Numeric, ban đầu được tạo ra bởi Jim Hugunin với sự đóng góp từ một số nhà phát triển khác. [6]

## Matplotlib

Matplotlib là một thư viện vector cho ngôn ngữ lập trình Python và là phần mở rộng toán học NumPy. Nó cung cấp API hướng đối tượng để nhúng các ô vào các ứng dụng bằng cách sử dụng các bộ công cụ GUI đa năng như Tkinter, wxPython, Qt hoặc GTK+. Ngoài ra còn có một giao diện "pylab" dựa trên một máy trạng thái (như OpenGL), được thiết kế gần giống với MATLAB. [4]

## OS

Trong python cung cấp các chức năng để tương tác với hệ điều hành. Os là một module cung cấp các chức năng dùng để giao tiếp với hệ điều hành hệ điều hành. Các mô-đun \* os \* và \* os.path \* bao gồm các chức năng để tương tác với các tệp, cây thư mục trong hệ thống.

1. **Keras**

Keras là một library được phát triển vào năm 2015 bởi François Chollet, là một kỹ sư nghiên cứu deep learning tại google. Nó là một open source cho neural network được viết bởi ngôn ngữ python. keras là một API bậc cao có thể sử dụng chung với các thư viện deep learning nổi tiếng như tensorflow (được phát triển bởi gg), CNTK (được phát triển bởi microsoft), Theano (người phát triển chính Yoshua Bengio). Keras có một số ưu điểm như: [5]

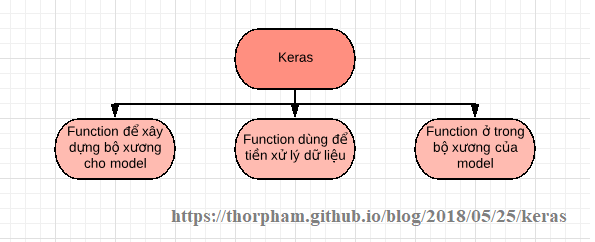
Dễ sử dụng, xây dựng model nhanh.

Có thể run trên cả cpu và gpu

Hỗ trợ xây dựng CNN, RNN và có thể kết hợp cả 2.

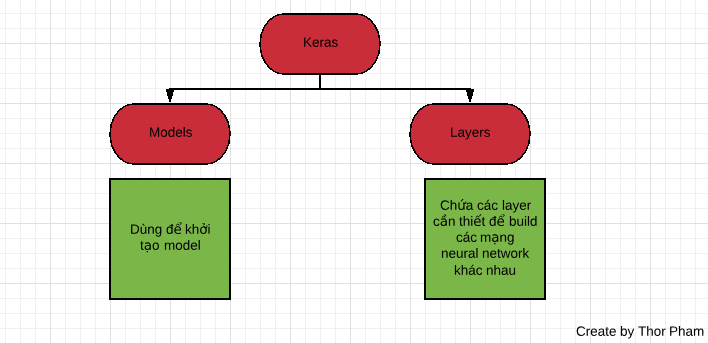
**Cấu trúc:**

Cấu trúc của keras chúng ta có thể chia ra thành 3 phần chính:



Hình 21. Thành phần của keras

Module dùng để xây dựng bộ xương cho model:



Hình 22. Module xây dựng Keras

# TÀI LIỆU THAM KHẢO

[1] David Cournapeau, *scikit-learn,* Scientific software in Python, https://en.wikip edia.org/wiki/Scikit-learn, 05/05/2019.

[2] Hau Nguyen, *Binary logistic regression (hồi quy logistic)*, Học máy cho người Việt, <http://ml4vn.blogspot.com/2017/08/bai-7-binary-logistic-regression-hoi.html>, 07/08/2017.

[3] Hồ Sỹ Hùng, *Tóm lược lịch sử phát triển của ngành Machine Learning,* https://techm aster.vn/posts/33923/lich-su-phat-trien-machine-learning, 09/06/2016.

[4] John D. Hunter, *Matplotlib*, Scientific software in Python, [https://en.wikipedia.org/ wiki/Matplotlib](https://en.wikipedia.org/%20wiki/Matplotlib), 26/02/2019.

[5] Thong Pham, *Tìm hiểu về thư viện keras trong deep learning,* Deep Learing Tutotials, <https://thorpham.github.io/blog/2018/05/25/keras/>, 25/05/2018.

[6] Travis Oliphant, *NumPy*, Scientific software in Python, [https://en.wikipedia.org/ wiki/NumPy](https://en.wikipedia.org/%20wiki/NumPy), 14/04/2019.

[7] Wes McKinney, *pandas (software)*, Scientific software in Python, https://en.wikip edia.org/wiki/Pandas\_(software), 28/04/2019.