**Bài 4: Chuẩn hóa dữ liệu**

**Link tham khảo thêm: https://vimentor.com/en/lesson/tien-xu-ly-du-lieu-trong-linh-vuc-hoc-may-phan-3**

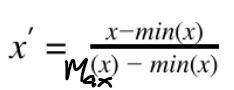
Bước tiền xử lý dữ liệu như một bước bắt buộc trong rất nhiều bài toán.

Điều khó khăn là mỗi thuật toán lại có những giả định về dữ liệu khác nhau nên nó cần những bước chuẩn hóa dữ liệu khác nhau.

Nếu chúng ta làm tốt bước tiền xử lý, thuật toán Máy học sẽ được cải thiện hiệu quả rõ rệt.

**4.1 Điều chỉnh tỉ lệ (Rescale Data)**

Dữ liệu gồm nhiều đặc tính (cột), và mỗi đặc tính thì lại có các đơn vị và độ lớn nhỏ khác nhau. Điều này tác động tới tính hiệu quả của nhiều thuật toán, ví dụ thời gian thực hiện, quá trình hội tụ, hay thậm chí ảnh hưởng cả tới độ chính xác của thuật toán. Chính vì vậy, người ta thường tiến hành điều chỉnh dữ liệu để các đặc tính cùng có chung một tỉ lệ (data scaling) . Và thường để các đặc tính có giá trị trong khoảng [0, 1]. Kết quả sẽ giúp cho nhiều thuật toán quan trọng trong Máy học sử dụng kĩ thuật Gradient Descent (sẽ đề cập sau) hội tụ nhanh. Việc điều chỉnh tỉ lệ thường dùng công thức sau đây (giả sử chúng ta đang làm trên một cột dữ liệu số cụ thể, gọi là *F*):

                                                               (4.1)

Trong đó : x’ là giá trị sau khi điều chỉnh; *x*: là giá trị ban đầu trong cột *F*,

 min( max): là giá trị nhỏ nhất (lớn nhất) của cột *F.*

Công thức (4.1) trên được cài đặt trong lớp MinMaxScaler [4] của gói preprocessing trong scikit-learn [3], một thư viện mã nguồn mở dùng Python được dùng rất phổ biến trong cộng đồng Máy học. Chương trình (4.1) sau sẽ cho kết quả giống nhau giữa thư viện và công thức (4.1).

Chương trình sau sẽ minh họa cho việc điều chỉnh tỉ lệ dữ liệu:

**01. from** pandas **import** read\_csv

**02. import** os

**03. from** sklearn **import** preprocessing

**04.** duongDan = os.getcwd() + **'\data\iris.data.csv'**

**05.** tenCot = [**'sepal length'**, **'sepal width'**, **'petal length'**, **'petal width'**, **'class'**]

**06.** duLieu = read\_csv(duongDan, names=tenCot)

**07.** maTran = duLieu.values

**08.** X = maTran[:,0:4]**09.** y = maTran[:,4]

**10.** dieuChinh = preprocessing.MinMaxScaler(feature\_range= (0,1))

**11.** X\_dieuChinh = dieuChinh.fit\_transform(X)

**12. print** (X[:3])         # In ra 3 dòng đầu của X

**13. print** (X\_dieuChinh[:3])        # In ra 3 dòng đầu của X sau khi điều chỉnh

**14.** max = duLieu[**'sepal length'**].max() # lấy giá trị lớn nhất của cột

**15.** min = duLieu[**'sepal length'**].min() # lấy giá trị nhỏ nhất của cột

**16.** x11 = duLieu[**'sepal length'**][0]    # lấy giá trị cột đầu tiên, hàng đầu tiên

**17.** x11\_dieuChinh = 1.0 \* (x11 - min) / (max - min)

**18. print** (x11\_dieuChinh)

**19.** X\_banDau = dieuChinh.inverse\_transform(X\_dieuChinh)

**20. print** (X\_banDau[:3])           # In ra 3 dòng đầu của X sau khi chuyển ngược lại

Chương trình 4.1: Điều chỉnh tỉ lệ của toàn bộ tập dữ liệu về [0,1].

Một vài chú thích thêm cho Chương trình 4.1:

-          Từ dòng 01-06: là kết nối tới tệp dữ liệu hoa Ailen

-          Dòng 07: chuyển toàn bộ dữ liệu từ DataFrame sang mảng dữ liệu nhiều chiều (ndarray)

-          Dòng 08: tách các đặc tính riêng ra, thường đặt là X (lấy các cột 0-3)

-          Dòng 09: tách nhãn riêng ra, thường đặt là y (lấy cột 4)

-          Dòng 10: thiết lập chế độ điều chỉnh dữ liệu với kĩ thuật MinMaxScaler

-          Dòng 11: Biến đổi X, dữ liệu sau khi biến đổi được lưu vào X\_dieuchinh

-          Dòng 12 và 13 để in ra dữ liệu trước và sau khi điều chỉnh

-          Từ dòng 14-18 để kiểm định cho công thức (4.1)

-          Dòng 19: chuyển lại điều chỉnh

-          Dòng 20: khi in ra để kiểm định X = X\_banDau

Thực thi Chương trình 4.1 sẽ cho kết quả

*[[5.1 3.5 1.4 0.2]*

*[4.9 3.0 1.4 0.2]*

*[4.7 3.2 1.3 0.2]]*

*[[ 0.22222222  0.625       0.06779661  0.04166667]*

*[ 0.16666667  0.41666667  0.06779661  0.04166667]*

*[ 0.11111111  0.5         0.05084746  0.04166667]]*

*0.222222222222*

*[[ 5.1  3.5  1.4  0.2]*

*[ 4.9  3.   1.4  0.2]*

*[ 4.7  3.2  1.3  0.2]]*

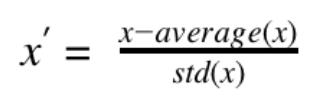
**4.2 Chuẩn hóa dữ liệu (Standardize Data)**

Nhiều thuật toán trong Máy học giả định rằng dữ liệu đầu vào có phân phối Gauss.  Chính vì vậy, khi chuẩn hóa dữ liệu về dạng chuẩn phân phối Gauss:

+ với giá trị trung bình bằng 0

+ và phương sai bằng 1 (người học có thể xem ở [2]). Nhờ việc chuẩn hóa, các thuật toán như *linear regression, logistic regression* được cải thiện.

Công thức để chuẩn hóa dữ liệu:

                                                                  (4.2)

Trong đó  average là giá trị trung binh, và  std là độ lệch chuẩn.

Thư viện scikit-learn cũng cài đặt công thức (4.2) trong lớp StandardScalar [5]:

|  |
| --- |
| **01. from** pandas **import** read\_csv  **02. import** os**03. from** sklearn **import** preprocessing  **04. from** numpy **import** set\_printoptions  **05.** duongDan = os.getcwd() + **'\data\iris.data.csv'**  **06.** tenCot = [**'sepal length'**, **'sepal width'**, **'petal length'**, **'petal width'**, **'class'**]  **07.** duLieu = read\_csv(duongDan, names=tenCot)  **08.** maTran = duLieu.values  **09.** X = maTran[:,0:4]  **10.** y = maTran[:,4]  **11.** dieuChinh = preprocessing.StandardScaler().fit (X)# lớp StandardScaler  12. X\_dieuChinh = dieuChinh.transform(X)  **13.** set\_printoptions(precision=3)  **14. print** (X\_dieuChinh[:5])  **15. print** (X\_dieuChinh.mean(axis = 0))      # tính giá trị trung bình mỗi cột  **16. print** (X\_dieuChinh.std(axis = 0))       # tính giá trị phương sai mỗi cột |

Chương trình 4.2: Điều chỉnh tỉ lệ của toàn bộ tập dữ liệu theo phân phối Gauss.

Một vài chú thích thêm cho chương trình 4.2:

-          Dòng 04: khai bóa thư viện numpy để định dạng số khi in ra

-          Dòng 05-10: giống như Chương trình 4.1

-          Dòng 11: dùng lớp StandardScaler để về sau có thể áp dụng cho tập dữ liệu kiểm tra hay dự đoán giống như đã áp dụng cho tập huấn luyện

-          Dòng 12: Chuyển dữ liệu sang dạng phân phối chuẩn Gauss

-          Dòng 13: Định dạng số khi in ra có 3 số sau phần thập phân

-          Dòng 14: Dữ liệu sau khi điểu chỉnh

-          Dòng 15: Tính trung bình cộng của mỗi cột tương ứng sau khi điều chỉnh dữ liệu. Tham số axis = 0 khi phép tính tác động nên cột, nếu axis = 1 khi phép tính tác động nên hàng.

-          Dòng 16: Tính phương sai của mỗi cột tương ứng sau khi điều chỉnh dữ liệu

Chú ý rằng Dòng 11 và 12, có thể dùng bằng một câu lệnh sau:

|  |
| --- |
| X\_dieuChinh = preprocessing.StandardScaler().fit\_transform(X) |

Tuy nhiên chúng ta nên giữ nguyên vì chúng ta cần đối tượng StandardScaler (ở dòng 11) để áp dụng cho dữ liệu dự đoán (hay kiểm tra) về sau.

Để kiểm định xem tổng của mỗi cột sau khi điều chỉnh có phân phối chuẩn Gauss, và kết quả của Chương trình 4.2 như sau:

*[[-0.901  1.032 -1.341 -1.313]*

*[-1.143 -0.125 -1.341 -1.313]*

*[-1.385  0.338 -1.398 -1.313]*

*[-1.507  0.106 -1.284 -1.313]*

*[-1.022  1.263 -1.341 -1.313]]*

*[ -4.737e-16  -6.632e-16   3.316e-16  -2.842e-16]*

*[ 1.  1.  1.  1.]*

Như chúng ta thấy, trung bình cộng của mỗi cột xấp xỉ 0, và phương sai là 1. Thư viện dùng trong scikit-learn đúng là cài đặt công thức (4.2)!

## 4.3 Bình thường hóa dữ liệu (Normalize Data)

Bình thường hóa dữ liệu là sự điều chỉnh tỉ lệ dữ liệu sao cho:

+ mỗi thể hiện (trên hàng) đều cho độ dài là 1.

Kĩ thuật này rất cần thiết cho dữ liệu thưa (gồm nhiều số 0) trên mỗi cột đặc tính. Điều này đặc biệt ảnh hưởng tới các thuật toán lấy trọng số của các giá trị nhập vào như *neuron networks*, hay các thuật toán dùng độ đo khoảng cách (như k-Nearest Neighbors). Thư viện scikit-learn cũng cài đặt tác vụ này trong lớp Normalizer [6]:

|  |
| --- |
| **01. from** pandas **import** read\_csv  **02. import** os  **03. from** sklearn **import** preprocessing  **04. from** numpy **import** set\_printoptions  **05.** duongDan = os.getcwd() + **'\data\iris.data.csv'**  **06.** tenCot = [**'sepal length'**, **'sepal width'**, **'petal length'**, **'petal width'**, **'class'**]  **07.** duLieu = read\_csv(duongDan, names=tenCot)  **08.** maTran = duLieu.values  **09.** X = maTran[:,0:4]  **10.** y = maTran[:,4]  **11.** dieuChinh = preprocessing.Normalizer().fit (X)# lớp Normalizer  12. X\_dieuChinh = dieuChinh.transform(X)  **13.** set\_printoptions(precision=3)  **14. print** (X\_dieuChinh[:5]) |

Chương trình 4.3: Điều chỉnh tỉ lệ của toàn bộ tập dữ liệu về dạng bình thường (normalization).

Chương trình 4.3 không khác gì nhiều Chương trình 4.2, ngoài việc sử dụng lớp Normalizer và bỏ 2 dòng cuối.

Chương trình 4.3 sẽ cho kết quả như sau:

*[[ 0.804  0.552  0.221  0.032]*

*[ 0.828  0.507  0.237  0.034]*

*[ 0.805  0.548  0.223  0.034]*

*[ 0.8    0.539  0.261  0.035]*

*[ 0.791  0.569  0.221  0.032]]*

Nếu chúng ta cộng tổng bình phương của các giá trị trên một hàng, kết quả sẽ là 1. Đây chính là điều chúng ta mong muốn.

## 4.4 Số hóa dữ liệu (Digitalization)

Rất nhiều các thuật toán Máy học chỉ chấp nhận dữ liệu nhập vào dưới dạng số.

Tuy nhiên, có nhiều trường hợp dữ liệu không được dưới dạng số, mà được ghi dưới

dạng liệt kê rời rạc, ví dụ như: ["male", "female"], ["from Europe", "from US", "from Asia"],

["uses Firefox", "uses Chrome", "uses Safari", "uses Internet Explorer"].

Hay như trong bài toán của chúng ta, hoa Ailen, cột kết quả cuối cùng (class) có 3 dạng:

[“Iris - setosa”, “Iris - versicolor”, “Iris - virginica”].

Để cho thuận tiện trong quá trình xử lý, những kiểu dữ liệu này cần phải chuyển sang kiểu

số nguyên, ví dụ ["male", "female"] có thể chuyển sang [0, 1], và

[“Iris - setosa”, “Iris - versicolor”, “Iris - virginica”] nên chuyển sang [0, 1, 2].

Thư viện scikit-learn có lớp LabelEncoder thực thi tác vụ này [7].

|  |
| --- |
| **01. from** pandas **import** read\_csv  **02. import** os  **03. from** sklearn **import** preprocessing  **04. from** numpy **import** set\_printoptions  **05.** duongDan = os.getcwd() + **'\data\iris.data.csv'**  **06.** tenCot = [**'sepal length'**, **'sepal width'**, **'petal length'**, **'petal width'**, **'class'**]  **07.** duLieu = read\_csv(duongDan, names=tenCot)  **08.** maTran = duLieu.values  **09.** X = maTran[:,0:4]**10.** y = maTran[:,4]  11. chuyenDoi = preprocessing.LabelEncoder()  12. chuyenDoi.fit(y)  13. y\_chuyenDoi = chuyenDoi.transform(y)  **14. print** y\_chuyenDoi |

Chương trình 4.4: Chuyển dữ liệu trong tập đặc trưng sang dạng số.

Một số chú thích cho Chương trình 4.4:

-          Dòng 11: tạo ra lớp LabelEncoder để chuyển đổi nhãn.

-          Dòng 12: nhận diễn các nhãn trong *y*.

-          Dòng 13: chuyển nhãn cũ sang nhãn mới trong đoạn [0, n-1], trong đó n là số các nhãn khác nhau trong *y*. Ở ví dụ của chúng ta n = 3.

Chương trình 4.4 sẽ cho kết quả như sau:

*[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0*

*0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1*

*1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2*

*2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2*

*2 2]*

Kết quả cho 50 số 0, 50 số 1 và 50 số 2 liên tiếp, lần lượt ứng với ba nhãn “Iris - setosa”,

“Iris - versicolor”,  và “Iris - virginica”. Như vậy là dữ liệu nhãn đã được chuyển sang

số như mong đợi. Vậy là mọi thứ đã sẵn sàng! :-)

## 4.5 Kết luận:

         Tiền xử lý là bước chuẩn bị dữ liệu thường được yêu cầu trước khi thực hiện các thuật toán trong Máy học, nhằm giúp thuật toán hiệu quả hơn. Trong nhiều bài toán, chúng ta phải sử dụng một vài cách xử lý dữ liệu khác nhau, nhằm chọn ra được cách phù hợp nhất.

* Một cấu trúc chương trình Máy học tới lúc này của chúng ta là:

1.      Kết nối dữ liệu (có thể online, hoặc tệp trên máy tính)

2.      Tách dữ liệu ra thành: dữ liệu nhập – đặc tính (features) và dữ liệu dự đoán (label)

3.      Áp dụng các kĩ thuật tiền xử lý dữ liệu cho dữ liệu nhập

* Mỗi một kĩ thuật điều chỉnh dữ liệu đều có nhược điểm riêng:

o   Nếu dữ liệu chúng ta có điểm kì dị (outliers), quá trình điều chỉnh và bình thường hóa (normalization) dữ liệu của chúng ta sẽ cho ra hầu hết các giá trị nằm trong đoạn dữ liệu rất nhỏ. Lưu ý rằng, rất nhiều tập dữ liệu có các điểm kì dị.

o    Trong khi nếu sử dụng chuẩn hóa dữ liệu (Standardization), dữ liệu mới của chúng ta sẽ không bị chặn trong một đoạn xác định, giống như ở Normalization.

o   Trong thực tế, kĩ thuật Standardization thường được khuyên dùng.

**bài tập**

Thực hiện các kỹ thuật chuẩn hóa dữ liệu đã học với bộ dữ liệu wine.csv.

Với mỗi kỹ thuật hiện thị kết quả (dòng/cột tùy chọn) để có thể kiểm chứng đặc tính của kỹ thuật chuẩn hóa đó.