Tiền xử lý dữ liệu trong lĩnh vực học máy (Phần 3)

Chuẩn hóa dữ liệu

Tương tự như việc tinh chỉnh dữ liệu, việc chuẩn hóa dữ liệu cũng là một bước vô cùng quan trọng trong việc giải quyết một vấn đề học máy. Lý do không chỉ bởi vì các thông tin thu được từ dữ liệu ra sao mà còn vì các phương pháp phân tích dữ liệu khác nhau tập trung vào các khía cạnh khác nhau của dữ liệu. Ví dụ, một phương pháp phân cụm tập trung vào việc phân tích sự tương tự của các điểm dữ liệu, trong khi phương pháp phân tích các thành phần chính PCA lại tập trung vào việc chỉ ra độ rộng của các thành phần chính. Nếu ta chuẩn hóa và thay đổi các thuộc tính của dữ liệu thì có thể nó làm tăng tính hiệu quả của phương pháp phân cụm nhưng lại che mờ đi kết quả của phương pháp phân tích PCA.

Trung tâm hóa dữ liệu(Centering data)

Trung tâm hóa dữ liệu là phương pháp đưa các điểm dữ liệu trong tập dữ liệu về xoay quanh giá trị 0 thay vì xoay quanh giá trị trung bình của tập dữ liệu. Việc đưa dữ liệu về trung tâm không làm thay đổi ma trận hiệp phương sai mà chỉ làm thay đổi các giá trị đặc trưng số. Trong một số trường hợp, việc đưa dữ liệu về trung tâm có thể làm cho các tính toán hội tụ nhanh hơn. Về mặt công thức ta có:

$$x' = x - average(x)$$

trong đó *x'* là giá trị sau khi trung tâm hóa, *x* là giá trị ban đầu và *average(x)* là giá trị trung bình của đặc trưng.

```
data = lw.values

# Centering data #

# x = x - mean

data[:,0:8] = data[:,0:8] - np.mean(data[:,0:8], axis=0)

print('Data after centering')

print(data)
```

Co giãn dữ liệu (Scaling data)

Co giãn dữ liệu là một phương pháp chuẩn hóa phạm vi của các đặc trưng dữ liệu và được thực hiện trong suốt quá trình tiền xử lý dữ liệu.

Vì phạm vi của các dữ liệu thô là rất rộng, trong khi đối với một số thuật toán học máy, các hàm mục tiêu của chúng sẽ không hoạt động đúng khi dữ liệu không được chuẩn hóa. Ví dụ là một mô hình phân lớp tính toán khoảng cách Euclidean giữa hai điểm dữ liệu thể hiện cho kích thước con người gồm chiều cao tính theo cm và cân nặng tính theo kg. Rõ ràng chúng ta không thể áp dụng trực tiếp lý thuyết Euclidian để tính khoảng cách giữa hai người vì hai số liệu về chiều cao và cân nặng là hoàn toàn khác nhau về bản chất. Trong trường hợp này, việc co giãn các đặc trưng về cùng một thước đo cụ thể là rất hữu ích.

Để co giãn dữ liệu dạng số, người ta thường áp dụng một trong các cách sau đây:

Chuẩn hóa min-max (rescaling)

Chuẩn hóa min-max là phương pháp đơn giản nhất trong việc co giãn phạm vi của đặc trưng bằng việc co giãn chúng về phạm vi [0,1] hoặc [-1,1]. Công thức chung được cho như sau:

$$x' = \frac{x - min(x)}{(x) - min(x)}$$

với *x* là giá trị ban đầu, *x'* là giá trị sau khi chuẩn hóa, *min(x)* là giá trị nhỏ nhất của đặc trưng và *max(x)* là giá trị lớn nhất của đặc trưng.

```
print('Min max scaling')

from sklearn import preprocessing as pp

mms = pp.MinMaxScaler()

data_mms = mms.fit_transform(data)

print(data_mms)
```

Co giãn trung bình (mean normalization)

Tương tự như phương pháp rescaling, phương pháp co giãn xoay quanh trung bình có giá trị nằm trong khoảng [-0.5, 0.5] và được cho bởi công thức:

$$x' = \frac{x - average(x)}{(x) - min(x)}$$

```
print('Mean normalisation')

mmin = data[:,0:8].min(axis=0)

mmax = data[:,0:8].max(axis=0)

maverage = data[:,0:8].mean(axis=0)

newdata = data.copy()

newdata[:,0:8] = (newdata[:,0:8] - maverage)

newdata[:,0:8] = np.divide(newdata[:,0:8], np.array(mmax - mmin))

print(newdata)
```

Chính quy hóa (standardisation)

Trong lĩnh vực học máy, chúng ta có thể sẽ phải xử lý một lượng lớn các kiểu dữ liệu khác nhau, ví dụ như dữ liệu dạng tín hiệu âm thanh, các điểm ảnh trong một bức ảnh,… và những dữ liệu này có thể là các dữ liệu nhiều chiều. Việc chính quy hóa dữ liệu giúp cho giá trị của mỗi đặc trưng có trung bình bằng 0 và phương sai bằng 1. Phương pháp này được sử dụng rộng rãi trong việc chuẩn hóa dữ liệu của nhiều thuật toán học máy (SVM, logistic regression và ANNs).

Để tính toán chính quy hóa dữ liệu, ta phải xác định trung bình và độ lệch chuẩn cho phân phối của mỗi đặc trưng. Tiếp theo ta lấy mỗi giá trị của đặc trưng trừ đi giá trị trung bình rồi chia cho độ lệch chuẩn của đặc trưng đó như công thức dưới đây:

$$x' = \frac{x - average(x)}{std(x)}$$

Trong đó *x* là véc-tơ đặc trưng ban đầu, *average(x)* là trung bình của véc-tơ đặc trưng đó và *std(x)* là độ lệch chuẩn của nó.

```
# Standardization #

print('Standardization')

std = pp.StandardScaler()

data_std = std.fit_transform(data)

print(data_std)
```

Vec-tơ đơn vị

Một lựa chọn khác để co giãn các thành phần của các véc-tơ đặc trưng là biến đổi sao cho véc-tơ đặc trưng sau khi biến đổi có độ dài bằng 1. Thông thường người ta sẽ lấy giá trị của mỗi đặc trưng chia cho độ dài Euclidean của véc-tơ đặc trưng:

$$x' = \frac{x}{\|x\|}$$

```
# Unit vector #

print('Unit vector')

# l2-normalize the samples

data_uv = pp.normalize(data, norm='l2')

print(data_uv)
```

Mã hóa đặc trưng dạng nhóm

Như đã nói ở phần trước, bộ dữ liệu của bạn có thể chứa các đặc trưng dạng nhóm. Các đặc trưng này thường được lưu trữ dưới dạng chữ để biểu thị cho các đặc tính khác nhau của dữ liệu. Một vài ví dụ của đặc trưng dạng nhóm như đặc trưng về màu sắc bao gồm “Đỏ”, “Vàng”, “Xanh”, đặc trưng về kích cỡ như “Nhỏ”, “Vừa”, “Lớn”, hay đặc trưng về vị trí địa lý như “Hà Nội”, “Ninh Bình”, “Hòa Bình”. Bất kể là loại nào thì chúng ta đều phải đối mặt với một vấn đề là làm thế nào để sử dụng các đặc trưng này trong việc phân tích dữ liệu. Rất nhiều thuật toán học máy có thể hỗ trợ các đặc trưng dạng nhóm nhưng cũng có rất nhiều thuật toán không thể chạy với loại đặc trưng này. Vì thế, các nhà phân tích dữ liệu phải đối mặt với thách thức về việc làm thế nào để chuyển các dữ liệu dạng nhóm thành các dữ liệu dạng số cho các tiến trình tiếp theo.

Mã hóa đặc trưng dạng nhóm đề cập đến vấn đề biến đổi một đặc trưng dạng nhóm thành một hoặc nhiều đặc trưng dạng số. Bạn có thể sử dụng bất kỳ phương pháp toán học nào hoặc phương pháp logic nào mà bạn muốn để chuyển đổi các đặc trưng dạng nhóm vì không có một giới hạn nào cho việc này cả. Các phép biến đổi cần phụ thuộc vào hướng phân tích của bạn. Tuy nhiên, trong phạm vi bài viết này, chúng ta cùng nhau tìm hiểu một số phương pháp mã hóa đặc trưng dạng nhóm phổ biến hiện nay như: *mã hóa số (numeric encoding*), *mã hóa one-hot (one-hot encoding)* và *mã hóa nhị phân (binary encoding).*

Mã hóa số - Numeric Encoding

Mã hóa số là việc ta gán mỗi giá trị của đăng trưng dạng nhóm thành một số bất kỳ và khác nhau từng đôi một. Ví dụ như trong thuộc tính màu sắc, ta có màu “Đỏ” là 1, “Vàng” là 2 và “Xanh” là 3. Phương pháp này cũng còn được gọi là mã hóa nhãn (label encoding) hoặc mã hóa số nguyên (integer encoding).

Bên cạnh đó, các giá trị số nguyên có bản chất thứ tự và các thuật toán học máy có khả năng hiểu được và khai thác bản chất này. Do đó cách mã hóa này thường phù hợp với các đặc trưng có tính chất cấp độ, thứ tự. Đặc trưng kích cỡ bao gồm “Nhỏ”, “Vừa”, “To” là một ví dụ rõ ràng cho dạng này.

Mã hóa One-Hot

Với các đặc trưng dạng nhóm không có thứ tự, việc sử dụng mã hóa số sẽ khiến bản chất dữ liệu bị thay đổi. Việc này cho phép mô hình giả định rằng các giá trị của đặc trưng dạng nhóm có bản chất thứ tự dẫn đến việc kết quả dự đoán của mô hình không chính xác. Trong trường hợp này, mã hóa one-hot có thể được áp dụng một cách hiệu quả hơn. Phương pháp này bỏ đi đặc trưng dạng nhóm và biến đổi mỗi giá trị của đặc trưng đó thành một biến nhị phân.

Trong ví dụ về đặc trưng màu sắc, đặc trưng này có 3 giá trị rời rạc và do vậy chúng ta sẽ biến đổi đặc trưng này thành 3 đặc trưng nhị phân đồng thời đặc trưng màu sắc bị xóa bỏ. Một cách tổng quát hóa, mã hóa one-hot sẽ cần *n* đặc trưng mới để lưu trữ giá trị cho một đặc trưng nhóm có *n* giá trị rời rạc.

Màu sắc		**Đỏ**	**Vàng**	**Xanh**
Đỏ		1	0	0
Vàng		0	1	0
Xanh		0	0	1

Mã hóa nhị phân

Mục tiêu của mã hóa nhị phân là sử dụng mã nhị phân để băm các giá trị của đặc trưng dạng nhóm thành các giá trị nhị phân.

			Binary encoded			
Categorical features		=	x1	x2	x4	x8
Cam	=>	1	1	0	0	0
Táo	=>	2	0	1	0	0
Xoài	=>	3	1	1	0	0
Mít	=>	4	0	0	1	0
Chanh	=>	5	1	0	1	0
Bưởi	=>	6	0	1	1	0
Nhãn	=>	7	1	1	1	0
Ổi	=>	8	0	0	0	1
Khế	=>	9	1	0	0	1

Sử dụng luật mũ của mã hóa nhị phân, chúng ta có thể tính được rằng với một đặc trưng dạng nhóm có số các giá trị rời rạc là *n*, thì ta sẽ cần ít nhất **log**(*n*+1)/**log**(2) đặc trưng nhị phân để lưu trữ. Điều đó có nghĩa là nếu một đặc trưng dạng nhóm có 4294967295 giá trị rời rạc khác nhau thì ta chỉ cần 32 đặc trưng nhị phân để lưu trữ mà thôi. Một con số rất ấn tượng so với 4294967295 đặc trưng của mã hóa one-hot.

Tổng kết

Khi đối mặt với một vấn đề mà bạn muốn giải quyết nó bằng Học Máy, bạn cần tìm ra những đặc trưng trong dữ liệu của bạn là gì, chúng gồm những loại nào và làm sao để tìm ra những đặc trưng hữu ích cho quá trình huấn luyện. Khi thực hiện những công việc này nghĩa là bạn đang làm bước thu thập dữ liệu trong quá trình phân tích dữ liệu.

Tiếp theo, bạn đã biết cách để thực hiện các phép toán cơ bản để làm sạch dữ liệu của bạn, ví dụ như dữ liệu *nan (not a number)*, xóa đi dữ liệu dư thừa và điền vào dữ liệu bị khuyết. Các đầu việc này nằm trong bước tinh chỉnh dữ liệu của quá trình phân tích dữ liệu.

Cuối cùng, bạn đã biết cách mã hóa các đặc trưng dữ liệu một cách đứng đắn trong một khía cạnh phù hợp với học máy. Đôi khi có thể các bạn sẽ muốn tiền xử lý dữ liệu trong suốt quá trình thu thập, điều này thực tế rơi vào bước biến đổi dữ liệu (transforming data). Nhưng nó không thành vấn đề, vì không có một luồng xử lý cứng nhắc nào cho việc chuẩn bị dữ liệu trước khi đưa vào huấn luyện.

References:

1. https://statisticalhorizons.com/wp-content/uploads/MissingDataByML.pdf
2. https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4
3. https://machinelearningmastery.com/handle-missing-data-python
4. https://arxiv.org/pdf/1710.01011.pdf
5. https://www.bu.edu/sph/files/2014/05/Marina-tech-report.pdf
6. *Time-Series Lecture at University of San Francisco by Nathaniel Stevens*