

Bài toán phân loại sử dụng SVM

Mục tiêu:

- Xây dựng được mô hình svm sử dụng thư viện sklearn.
- Ứng dụng, hiểu cách áp dụng mô hình svm vào giải quyết bài toán thực tế (vd: phân loại tin tức dạng văn bản) .
- Sử dụng độ đo Accuracy để làm độ đo đánh giá chất lượng mô hình.

Vấn đề:

- Có một tập các văn bản dạng text không có nhãn, làm sao để biết văn bản này là thuộc về thể loại nào, pháp luật, đời sống, văn học, thể thao ...
- Cần xây dựng mô hình học máy phân loại các thể loại của văn bản dựa trên nội dung.

Dữ liệu:

- Có tập các văn bản và nhãn tương ứng của từng văn bản trong một khoảng thời gian
- Tập các nhãn - 10 nhãn văn bản: > Giải trí, Khoa học - Công nghệ, Kinh tế, Pháp luật, Sức khỏe, Thể thao, Thời sự, Tin khác, Độc giả, Đời sống - Xã hội
- Ví dụ văn bản nhãn **thể thao**: > "Dân_trí Real Madrid đã dẫn trước trong cả trận đấu , nhưng họ vẫn phải chấp_nhận bị Dortmund cầm hòa 2-2 ở Bernabeu . Real Madrid chấp_nhận đứng thứ_hai ở bảng F Champions League ..."

Bài toán:

- Input: tập các từ trong văn bản 1 mẫu dữ liệu $X = [x_1, x_2, \dots, x_n]$
- Output: nhãn y là 1 trong 10 nhãn trên

```
[ ]: # Cài đặt thư viện xử lý ngôn ngữ cho tiếng Việt!
```

```
!pip install pyvi
```

```
[3]: import matplotlib.pyplot as plt
import numpy as np

from sklearn.datasets import load_files
from pyvi import ViTokenizer

from sklearn import svm
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import learning_curve

%matplotlib inline
```

1. Sử dụng `sklearn.datasets.load_files` để load dữ liệu từ thư mục đã down từ trước

Cấu trúc thư mục như sau

- data/news_1135/
 - Kinh tế:
 - * bài báo 1.txt
 - * bài báo 2.txt
 - Pháp luật
 - * bài báo 3.txt
 - * bài báo 4.txt

```
[4]: data_train = load_files(container_path="data/news_1135/", encoding="utf-8")

print(data_train filenames)
print()

print("Tong so file: {}".format( len(data_train.filenames)))
print("Danh sách nhãn và id tương ứng: ", [(idx, name) for idx, name in
↪ enumerate(data_train.target_names)] )
```

```
['data/news_1135/Tin khác/0218e1df21ce358b9c6485176a48f1fcaeedef67.txt'
'data/news_1135/Khoa học - Công
nghệ/bf9889f5f2fffd6c92fa877d35ef0ef5f34f0666d.txt'
'data/news_1135/Tin khác/d74aab054ffe9f8661df13bc52b438b48a63fe48.txt'
...
'data/news_1135/Thời sự/a06c1ec4c146d3b4eb5070a1967e10e5e21bdc5b.txt'
'data/news_1135/Sức khỏe/4187c4a1d528fd9ea4630d2709229df0b0d09c3d.txt'
'data/news_1135/Thể thao/7adaf0c561796f2411340150f18417543ad4403c.txt']
```

Tong so file: 1135

Tiền xử lý dữ liệu đưa dữ liệu từ dạng text về dạng ma trận

- Thử nghiệm để kiểm tra hoạt động chuyển hoá dữ liệu về dạng ma trận

```
[5]: # load dữ liệu các stopwords
with open("data/vietnamese-stopwords.txt", encoding="utf-8") as f:
    stopwords = f.readlines()
stopwords = [x.strip().replace(" ", "_") for x in stopwords]
print("Danh sách 10 từ dừng đầu tiên (từ không mang ý nghĩa phân loại): ",
↪ stopwords[:10])

# Transforming data
# Chuyển hoá dữ liệu text về dạng vector tfidf
# - loại bỏ từ dừng
```

```

# - sinh từ điển
module_count_vector = CountVectorizer(stop_words=stopwords)
model_rf_preprocess = Pipeline([('vect', module_count_vector),
                                ('tfidf', TfidfTransformer()),
                                ])
# Hàm thực hiện chuyển đổi dữ liệu text thành dữ liệu số dạng ma trận
# Input: Dữ liệu 2 chiều dạng numpy.array, mảng nhãn id dạng numpy.array
data_preprocessed = model_rf_preprocess.fit_transform(data_train.data,
↳data_train.target)
print("10 từ đầu tiên trong từ điển:")
i = 0
for k,v in module_count_vector.vocabulary_.items():
    i+=1
    print(i, ": ", (k, v))
    if i > 10:
        break

```

Danh sách 10 từ dừng đầu tiên (từ không mang ý nghĩa phân loại): ['a_lô', 'a_ha', 'ai', 'ai_ai', 'ai_nấy', 'ai_đó', 'alô', 'amen', 'anh', 'anh_ấy']

10 từ đầu tiên trong từ điển:

```

1 : ('dân_trí', 6928)
2 : ('sở', 17869)
3 : ('gd', 7729)
4 : ('đt', 23214)
5 : ('tỉnh', 20851)
6 : ('gia_lai', 7816)
7 : ('văn_bản', 21779)
8 : ('2258', 858)
9 : ('sgdđt', 17039)
10 : ('vp', 21572)
11 : ('chấn_chỉnh', 4971)

```

Bài 1: Sử dụng trực tiếp TfidfVectorizer()

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```

[ ]: # Chú ý: có thể biến đổi dữ liệu về dạng Tf-Idf trực tiếp sử dụng
↳TfidfVectorizer()
# Bài tập:
# - thực hiện điều đó
# - hiển thị 10 từ trong văn bản đầu tiên có giá trị tfidf cao nhất
# code

```

Chia dữ liệu làm 2 phần training và testing

- Training chiếm 80 % dữ liệu

- Testing chiếm 20 % dữ liệu

```
[6]: from sklearn.model_selection import ShuffleSplit

# chia dữ liệu thành 2 phần sử dụng hàm train_test_split.
test_size = 0.2
# cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
X_train, X_test, y_train, y_test = train_test_split( data_preprocessed,
    ↪data_train.target, test_size=test_size)

# hiển thị một số thông tin về dữ liệu
print("Dữ liệu training: ", X_train.shape, y_train.shape)
print("Dữ liệu testing: ", X_test.shape, y_test.shape)
print("Danh sách nhãn và id tương ứng: ", [(idx, name) for idx, name in
    ↪enumerate(data_train.target_names)] )
```

```
Dữ liệu training: (908, 24389) (908,)
Dữ liệu testing: (227, 24389) (227,)
Danh sách nhãn và id tương ứng: [(0, 'Giải trí'), (1, 'Khoa học - Công
nghệ'), (2, 'Kinh tế'), (3, 'Pháp luật'), (4, 'Sức khỏe'), (5, 'Thể
thao'), (6, 'Thời sự'), (7, 'Tin khác'), (8, 'Đời sống - Xã hội'),
(9, 'Độc giả')]
```

2. Training svm model

Sử dụng thư viện sklearn để xây dựng mô hình - Thử trước với hàm nhân phân tách là Linear, tham số C=1.0 - svm.SVC(kernel='linear', C=1.0)

```
[7]: print("- Training ...")
print("- Train size = {}".format(X_train.shape))
model = svm.SVC(kernel='linear', C=1)
model.fit(X_train, y_train)
print("- model - train complete")
```

```
- Training ...
- Train size = (908, 24389)
- model - train complete
```

3. Testing svm model

Thực hiện dự đoán nhãn cho từng văn bản trong tập test

Độ đo đánh giá: > accuracy = tổng số văn bản dự đoán đúng / tổng số văn bản có trong tập test

```
[8]: from sklearn.metrics import accuracy_score
print("- Testing ...")
y_test_pred = model.predict(X_test)
```

```
y_train_pred = model.predict(X_train)
print("- Test Acc = {}".format(accuracy_score(y_test, y_test_pred)))
print("- Train Acc = {}".format(accuracy_score(y_train, y_train_pred)))
```

```
- Testing ...
- Test Acc = 0.8370044052863436
- Train Acc = 0.9933920704845814
```

Bài 2: thực hiện lại các bước trên với kernel = 'rbf'

```
[ ]: # code
```

3.1. Thực hiện sử dụng model đã được train để infer 1 văn bản mới

- Dữ liệu mới đến ở dạng dữ liệu thô => cần tiền xử lý dữ liệu về dạng dữ_liệu_ma_trận
- infer sử dụng hàm model.predict(dữ_liệu_ma_trận)

```
[17]: # Tiền xử lý dữ liệu sử dụng module model_rf_preprocess.
new_doc = "Công phượng ghi bàn cho đội tuyển Việt nam"
# Trước hết, cần thực hiện tách từ sử dụng pyvi
tokenized_new_doc = ViTokenizer.tokenize(new_doc)
# Cần đưa văn bản ở dạng mảng/vector
tokenized_new_doc = [tokenized_new_doc]
print(tokenized_new_doc)
# Rồi sử dụng module model_rf_preprocess
input_data_preprocessed = model_rf_preprocess.transform(tokenized_new_doc)
print(input_data_preprocessed)
```

```
['Công phượng ghi_bàn cho đội_tuyển Việt_nam']
(0, 24149)    0.4617859604824952
(0, 21498)    0.23577234678310735
(0, 15553)    0.6394232142292748
(0, 7777)     0.4617859604824952
(0, 5847)     0.33023750089838017
```

Bài 3: dự đoán nhãn của văn bản trên

```
[ ]: # code
```

4. Bài tập bổ sung:

4.1 Thử nghiệm các tham số

- Các tham số với giá trị khác nhau có thể ảnh hưởng đến kết quả học

- Cần thử nghiệm kỹ lưỡng để đưa ra kết quả khách quan: tham số C, gamma, kernel.
 - Chọn mô hình với bộ tham số cho kết quả tốt nhất
- Gợi ý:
 - <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
 - Sử dụng grid search

Bài 4: Vẽ Learning curve khảo sát Acc của SVM-linear với tham số C thay đổi

```
[ ]: # code
# tham khảo miền của C
list_C = [0.001, 0.01, 0.1, 1, 5.0, 10.0, 100]
```

Bài 5: Sử dụng GridSearchCV để tìm bộ tham số tốt nhất

```
[46]: # code
# Có thể tham khảo giá trị các tham số như sau
params_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
               'gamma': [0.0001, 0.001, 0.01, 0.1],
               'kernel': ['linear', 'rbf', 'poly'] }
```

4.2 Phân loại số viết tay

```
[81]: # Standard scientific Python imports
import matplotlib.pyplot as plt

# Import datasets, classifiers and performance metrics
from sklearn import datasets, svm, metrics

# The digits dataset
digits = datasets.load_digits()

# The data that we are interested in is made of 8x8 images of digits, let's
# have a look at the first 4 images, stored in the `images` attribute of the
# dataset. If we were working from image files, we could load them using
# matplotlib.pyplot.imread. Note that each image must have the same size. For
# →these
# images, we know which digit they represent: it is given in the 'target' of
# the dataset.
images_and_labels = list(zip(digits.images, digits.target))
for index, (image, label) in enumerate(images_and_labels[:4]):
    plt.subplot(2, 4, index + 1)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    plt.title('Training: %i' % label)

# To apply a classifier on this data, we need to flatten the image, to
```

```
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
target = digits.target
X_train, X_test, y_train, y_test = train_test_split( data, target,
    ↳test_size=test_size)

print("Dữ liệu training = ", X_train.shape, y_train.shape)
print("Dữ liệu testing = ", X_test.shape, y_test.shape)
```

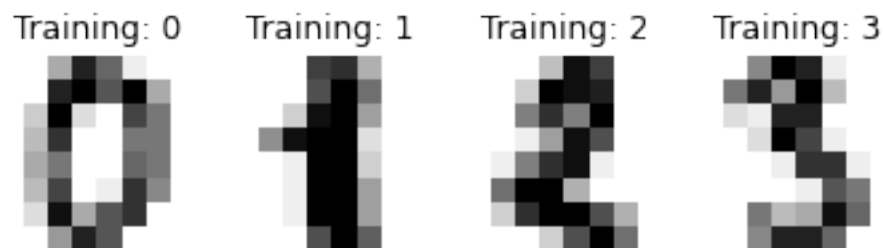
Dữ liệu training = (1437, 64) (1437,)

Dữ liệu testing = (360, 64) (360,)

Training...

Testing

0.9916666666666667



Bài 6: phân loại với dữ liệu trên

```
[1]: ##### exercise #####
# Yêu cầu: Ứng dụng mô hình svm vào bài toán phân loại ảnh
# Gợi ý: dữ liệu đã được chia train, test, Áp dụng phần 2. và 3. để training và
    ↳testing model. Chú ý nên có thêm phần tuning model
#####

# model = None

#####
```