SVM

August 6, 2023

1 Bài toán

Phân loại văn bản sử dụng SVM

Mục tiêu:

- Xây dựng được mô hình Naive Bayes sử dụng thư viện sklearn.
- Ứng dụng, hiểu cách áp dụng mô hình nb vào giải quyết bài toán thực tế (vd: phân loại 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 …

Dữ liệu: - 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: Phân loại - Input: n vector mã hóa của các văn bản - ma trận $X = [x_1, x_2, ...x_n]$ - Output: nhãn y là 1 trong 10 nhãn trên

2 Các bước tiến hành

2.1 Chuẩn bị các thư viện cần thiết

[ ]:
```
!pip3 install pyvi
```

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import learning_curve

from sklearn.datasets import load_files
from pyvi import ViTokenizer  #####

from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
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.metrics import accuracy_score

%matplotlib inline
```

2.2 Load dữ liệu từ thư mục đã có sẵn

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

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

print(data_train.filenames)
print()

print("Tổng số file: {}".format(len(data_train.filenames)))
```

2.3 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

[ ]:
```
# 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(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
```

[ ]:
```
from sklearn.feature_extraction.text import CountVectorizer

corpus = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?",
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
# vectorizer.get_feature_names_out()

for k, v in vectorizer.vocabulary_.items():
    i += 1
    print(i, ": ", (k, v))
    # if i > 10:
    #     break


print(X.toarray())

vectorizer2 = CountVectorizer(analyzer="word", ngram_range=(2, 2))
X2 = vectorizer2.fit_transform(corpus)
# vectorizer2.get_feature_names_out()
```

2.3.1 Bài 1: sử dụng TfidfVectorizer

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

[ ]:
```
### Code Here
```

```
###
```

2.4 Chia dữ liệu làm 2 phần training và testing

- Training chiếm 80 % dữ liệu
- Testing chiếm 20 % dữ liệu

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

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)

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

2.5 Training svm model

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

[ ]:
```
print("- Training ...")
print("- Train size = {}".format(X_train.shape))
model = svm.SVC(kernel="linear", C=1.0)
model.fit(X_train, y_train)
print("- model - train complete")
```

2.6 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

[ ]:
```
from sklearn.metrics import accuracy_score

print("- Testing ...")
y_pred = model.predict(X_test)
print("- Acc = {}".format(accuracy_score(y_test, y_pred)))
```

2.6.1 Bài 2: Thực hiện lại các bước trên với kernel 'rbf'

[ ]:
```
### Code Here

###
```

2.6.2 Bài 3: Dự đoán nhãn của văn bản

[ ]:
```
new_doc = "Công phượng ghi bàn cho đội tuyển Việt nam"
### Code Here

###
```

2.7 4. Bài tập bổ sung:

2.7.1 4.1 Thử nghiệm các tham số

- Các tham số với giá trị khác nhau có thể ảnh hưởng để kết quả học
- Cần thử nghiệm kỹ lượng 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

[ ]:
```
list_C = [0.001, 0.01, 0.1, 1, 5.0, 10.0, 100]
list_acc = []
title = "Learning Curves SVM, Linear kernel, change C"

# duyệt qua mảng các giá trị của tham số C
for i, C in enumerate(list_C):
    # Với từng giá trị C nhận được,
    # thực hiện build model và training cross-validate
    # vẽ kết quả tìm được lên đồ thị đường.
    model = svm.SVC(kernel="linear", C=C)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    list_acc.append(accuracy_score(y_test, y_pred))
```

[ ]:
```
import seaborn as sns

fig = sns.lineplot(x=list(range(0, 7)), y=list_acc)
fig.set_xticks(range(0, 7))
fig.set_xticklabels([0.001, 0.01, 0.1, 1, 5.0, 10.0, 100])
```

[ ]:
```
# hàm sinh id màu
def get_cmap(n):
    return "C" + str(n)


# Hàm thực hiện training model, crossvalidate và vẽ lên đồ thị sử dụng mat␣
↪libplot
```

```
def plot_learning_curve(estimator, title, label_curve, X, y, ylim=None,␣
↪cv=None, n_jobs=1, train_sizes=np.linspace(0.1, 1.0, 5), new_plot=False,␣
↪idx_color=0):
    # Khởi tạo bức ảnh mới với thư viện plot lib
    if new_plot:
        # plt.figure()
        plt.title(title)
        plt.xlabel("Training examples")
        plt.ylabel("Accuracy")
        plt.grid()

    # chú thích nếu có
    if ylim is not None:
        plt.ylim(*ylim)

    # thực hiện training model, ghi nhận các giá trị trong quá trình training
    # cv = số fold cross validate, số phần bộ dữ liệu được chia để thực hiện␣
↪training testing.
    # train_sizes = mảng tỉ lệ, các tỉ lệ được hệ thống chọn làm điểm dừng để␣
↪thực hiện 1 testing
    # train_sizes = [0.3, 0.5] => hệ thống lấy 30 % dữ liệu để train và thực␣
↪hiện test, tương tự 50 % ..
    # scoring = hàm mục tiêu để đánh giá chất lượng mô hình và vẽ lên đồ thị
    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y,␣
↪cv=cv, n_jobs=n_jobs, train_sizes=train_sizes, scoring="accuracy")

    # Lấy trung bình cộng các giá trị output của các fold
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    # random 1 màu để vẽ
    color = get_cmap(idx_color)

    # thực hiện vẽ các giá trị số lên đồ thị với màu vừa được random
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,␣
↪test_scores_mean + test_scores_std, alpha=0.1, color=color)
    plt.plot(train_sizes, test_scores_mean, "o-", color=color,␣
↪label=label_curve)

    plt.legend(loc="best")
    return plt
```

[ ]:
```
list_C = [0.001, 0.01, 0.1, 1, 5.0, 10.0, 100]
# model title
```

```
title = "Learning Curves SVM, Linear kernel, change C"

# duyệt qua mảng các giá trị của tham số C
for i, C in enumerate(list_C):
    # Với từng giá trị C nhận được,
    # thực hiện build model và training cross-validate
    # vẽ kết quả tìm được lên đồ thị đường.
    text_clf = Pipeline(
        [
            ("clf", svm.SVC(kernel="linear", C=C)),  # mô hình svm với tham số C
        ]
    )

    plt = plot_learning_curve(text_clf, title, "C = %f" % (C),␣
 ↪data_preprocessed, data_train.target, (0.0, 1.01), cv=10, n_jobs=-1,␣
 ↪idx_color=i, new_plot=i == 0)

# lưu hình ảnh ra file
# plt.savefig('images/changeC.png', bbox_inches='tight')
plt.show()
```

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

[ ]:
```
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"]}

model = svm.SVC()
# Create the GridSearchCV object
best_model = GridSearchCV(model, params_grid, cv=4, n_jobs=-1,␣
 ↪scoring="accuracy")

# Fit the data with the best possible parameters
best_model.fit(X_train, y_train)

# Print the best estimator with it's parameters
print(best_model.best_params_)
print(best_model.best_estimator_)

# Test best_model
print("Testing")
y_pred = best_model.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

2.7.2 4.2 Phân loại số viết tay

[ ]:
```
# 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)
```

2.7.3 Bài 6

[ ]:
```
##### 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
######################
```