Support Vector Machine

Ko, Youngjoong

Dept. of Computer Science & Engineering,
SKK University

Index

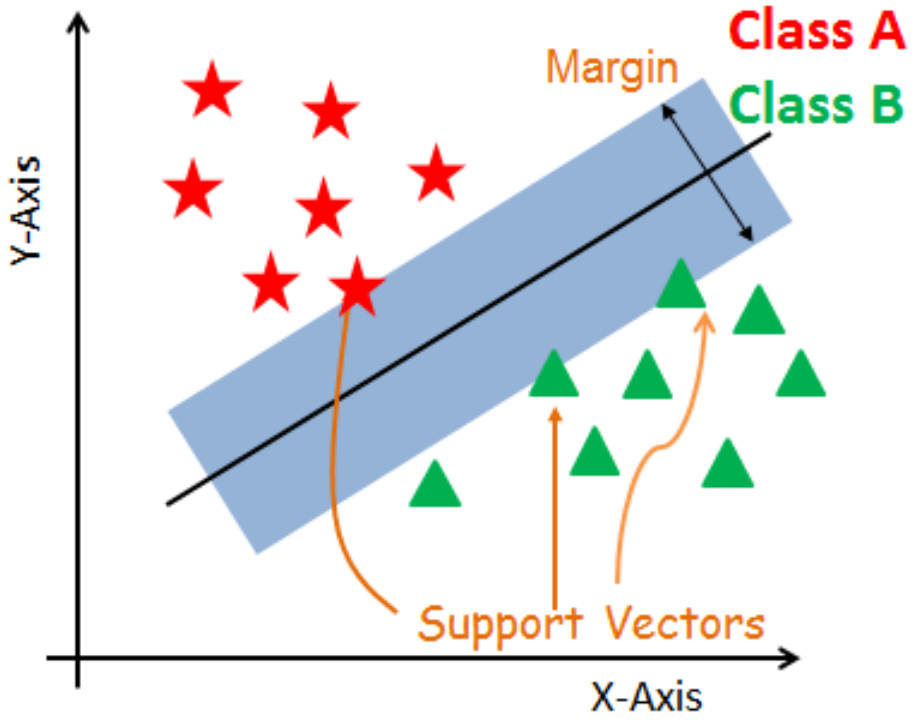
- **Introduction**
- **Support Vector Machine**
- **Assignment**

Introduction

- **Introduction**
  - The assignment is to implement the ‘Support Vector Machine (SVM)’ model for text classification.
  - We will provide main.py, util.py, train.json and test.json. You have to implement the Support Vector Machine using scikit-learn library in main.py.

Support Vector Machine

- **Support Vector Machine (Joachims, 1998)**
  - The SVM finds the surface that separate examples labeled as A from examples labeled B in the training dataset.



Support Vector Machine

- **There are two Python Classes in the 'sklearn.svm'**
  - svm.LinearSVC (Linear Support Vector Classification)
    - LinearSVC is a class which is capable of performing multi-class classification on a dataset.
    - You have to use LinearSVC class for this assignment.
  - svm.LinearSVR(Linear Support Vector Regression)
    - The methods of Support Vector Classification can be extended to LinearSVR class for regression problems. This method is called Support Vector Regression.

Support Vector Machine

- **svm.LinearSVC Parameters**
  - C : float, optional (default =1.0)
    - A parameter for regularization. The strength of the regularization is inversely proportional to C.
    - The value of the C must be strictly positive.
  - max_iter : int (default=1000)
    - The maximum number of iterations to be run.

Support Vector Machine

❖ Define and Learn the SVM model using Train data

- Define the 'SVM' model.

```
from sklearn.svm import LinearSVC
classifier = LinearSVC(C=1.0, max_iter=1000)
classifier.fit(train_inputs, train_labels)
```

- Input 'Train data input list' and 'Train labels list' to SVM model.

```
from sklearn.svm import LinearSVC
classifier = LinearSVC(C=1.0, max_iter=1000)
classifier.fit(train_inputs, train_labels)
```

Train input

Train label

Support Vector Machine

❖ **Evaluate the model using Test data**

```
prediction = classifier.predict(test_inputs)
```

Test input

Given Data

- **train.json & test.json**
  - Data Format : a list of dictionaries
  - Data Example :

```
[
    {
        “paragraph” : “Since Game of Throne first aired, …”
        “label”: “tv”,
        “id”: “31”      *this is an article id
    },
    …
]
```

  - The number of Labels : 5 (finance, lifestyle, tv, sports, entertainment)

Assignment

- **Various Possible Inputs**
  - TF
  - Normalized TF-IDF
  - Binary vector
    - 1 if a word in document else 0
    - For example,

      doc_a = I love dog doc_b = I like cat

      Vocab = { cat, dog, I, like, love }

      vec_a = [0, 1, 1, 0, 1] vec_b = [1, 0, 1, 1, 0]

Assignment

- **Assignment**
  - Implement the ‘SVM’ model for text classification in main.py.
    - Use TF, normalized TF-IDF and binary vectors as input for the model respectively.
  - Implementing the SVM model consists of 3 steps
    - Preprocessing Train/Test data and calculating vectors (TF, normalized TF-IDF and binary)
    - Training the model with Train data
    - Evaluating the model with Test data
  - You have to get at least 70% of accuracy in the test dataset for all input vectors (TF, normalized TF-IDF and binary).

Assignment

- **Assignment**
  - In this assignment, you have to calculate precision, recall, and f1-score for each labels by yourself.
  - You also need to calculate micro/macro average precision/recall/f1-score, and model accuracy.
  - Additional libraries not allowed.
  - See page 10 for specific output format.

Assignment

❖ Submission File

1) StudentName _StudentID_main.py (python version 3.x)
   - e.g., 홍길동_2020123123_main.py.
     e.g., MichaelJackson_2020123123_main.py

2) StudentName _StudentID.txt
   - e.g., 홍길동_2020123123.txt
     e.g., MichaelJackson_2020123123.txt

Assignment

- **Outlook of the Text File**

```
Input Type : Binary
               precision    recall  f1-score    # docs

entertainment      75.00     60.00     66.67       20
      finance      75.00     75.00     75.00       20
    lifestyle      73.91     85.00     79.07       20
       sports      81.82     90.00     85.71       20
           tv      89.47     85.00     87.18       20

    micro avg      79.00     79.00     79.00      100
    macro avg      79.04     79.00     78.73      100
     accuracy                           79.00      100

Input Type : TF
               precision    recall  f1-score    # docs

entertainment      73.68     70.00     71.79       20
      finance      70.00     70.00     70.00       20
    lifestyle      75.00     75.00     75.00       20
       sports      78.26     90.00     83.72       20
           tv      88.89     80.00     84.21       20

    micro avg      77.00     77.00     77.00      100
    macro avg      77.17     77.00     76.95      100
     accuracy                           77.00      100

Input Type : TF-IDF
               precision    recall  f1-score    # docs

entertainment      70.59     60.00     64.86       20
      finance      84.21     80.00     82.05       20
    lifestyle      73.91     85.00     79.07       20
       sports      85.71     90.00     87.80       20
           tv      80.00     80.00     80.00       20

    micro avg      79.00     79.00     79.00      100
    macro avg      78.89     79.00     78.76      100
     accuracy                           79.00      100
```

Assignment

- **Cautions**
  - Use ‘Python3’ and ‘Google Colab’.
  - Do not import any library except already imported libraries.
  - Copy will be scored 0.

Thank you for your attention!

고 영 중 (Ko, Youngjoong)

http://nlp.skku.edu/