SVM 분류 실습

선형 SVM 실습 #1 (분류)

- 실습 코드: 링크
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)

- 데이터 로더, 범주형 데이터 변환, 데이터 분할
  - 기존 실습과 동일

```
# Iris data 불러오기
import seaborn as sns # seaborn을 불러옴.
iris=sns.load_dataset('iris') # iris라는 변수명으로 Iris data를 download함.
X=iris.drop('species',axis=1) # 'species'열을 drop하고 특성변수 X를 정의함.
y_=iris['species'] # 'species'열을 label y를 정의함.

from sklearn.preprocessing import LabelEncoder # LabelEncoder() method를 불러옴
classle=LabelEncoder()
y=classle.fit_transform(iris['species'].values) # species 열의 문자형을 범주형 값으로 전환

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.3, random_state=123, stratify=y)

```

선형 SVM 실습 #1 (분류)

- 실습 코드: 링크
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)

- SVM API: Manual
  - SVC 클래스 호출
    - Kernel='linear', C=에러허용정도

```
#7.3 Python을 이용한 SVM
from sklearn.svm import SVC #SVM함수의 호출
svm=SVC(kernel='linear',C=1.0,random_state=1)
svm.fit(X_train,y_train) #SVM추정
y_train_pred=svm.predict(X_train) #train set의 y 예측치 구하기
y_test_pred=svm.predict(X_test)  #test set의 y예측치 구하기
```

선형 SVM 실습 #1 (분류)

- 실습 코드: 링크
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)

- Accuracy & Confusion Matrix
  - 기존 실습과 동일

```
[9]  1 from sklearn import metrics
     2 print(metrics.accuracy_score(y_train,y_train_pred))  # train set의 accuracy ratio
     3 print(metrics.accuracy_score(y_test,y_test_pred))   # test set의 accuracy ratio
```

```
0.9904761904761905
0.9777777777777777
```

```
1 metrics.confusion_matrix(y_test,y_test_pred)  #confusion_matrix
```

```
array([[15,  0,  0],
       [ 0, 14,  1],
       [ 0,  0, 15]])
```

선형 SVM 실습 #1 (분류)

- 실습 코드: 링크
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)

- Accuracy, Precision, Recall, F1 평가 지표
  - Metrics.classification_report 사용

```
1 from sklearn import metrics
2 print(metrics.classification_report(y_test,y_test_pred))   #classification_report
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.93      0.97        15
           2       0.94      1.00      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

~> 평가지표.

비선형 SVM 실습 #1 (분류)

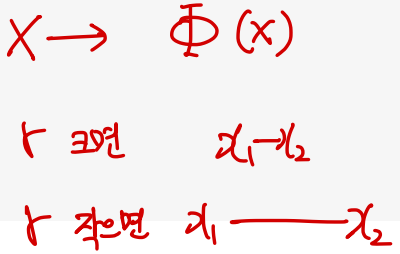
- 실습 코드: 링크
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)

- 데이터 로더, 범주형 데이터 변환, 데이터 분할
  - 기존 실습과 동일

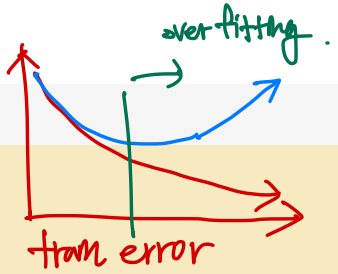
```
# Iris data 불러오기
import seaborn as sns # seaborn을 불러옴.
iris=sns.load_dataset('iris') # iris라는 변수명으로 Iris data를 download함.
X=iris.drop('species',axis=1) # 'species'열을 drop하고 특성변수 X를 정의함.
y_=iris['species'] # 'species'열을 label y를 정의함.

from sklearn.preprocessing import LabelEncoder # LabelEncoder() method를 불러옴
classle=LabelEncoder()
y=classle.fit_transform(iris['species'].values) # species 열의 문자형을 범주형 값으로 전환

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.3, random_state=123, stratify=y)

```

비선형 SVM 실습 #1 (분류)

- 실습 코드: 링크
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)



- SVM API: Manual
  - SVC 클래스 호출
    - Kernel='rbf', gamma=rbf의 설정 파라미터, C=에러허용정도
      - Gamma 작을 수록 두 데이터의 거리를 실제보다 멀게(overfit), 클수록 두 데이터의 거리를 실제보다 가깝게 변환함 (underfit)



[5]
```
1 ###############Kernel SVM 예제#############################
2 from sklearn.svm import SVC #SVM함수의 호출
3 ksvm=SVC(kernel='rbf',C=1.0,gamma=0.2,random_state=42)
4 ksvm.fit(X_train,y_train)
5 y_train_pred=ksvm.predict(X_train) # kernel SVM을 이용한 train set의 y 예측치 구하기
6 y_test_pred=ksvm.predict(X_test) # kernel SVM을 이용한 #test set의 y예측치 구하기
```

비선형 SVM 실습 #1 (분류)

- 실습 코드: 링크
- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2)

- Accuracy & Confusion Matrix
  - 기존 실습과 동일

```
[13]  1 from sklearn import metrics
      2 print(metrics.accuracy_score(y_train,y_train_pred))  # train set의 accuracy ratio
      3 print(metrics.accuracy_score(y_test,y_test_pred))   # test set의 accuracy ratio
```

```
0.9904761904761905
0.9555555555555556
```

```
1 metrics.confusion_matrix(y_test,y_test_pred)   #confusion_matrix
```

```
array([[15,  0,  0],
       [ 0, 13,  2],
       [ 0,  0, 15]])
```

비선형 SVM 응용 #1 (분류)

- 실습 코드: [링크]()
- 데이터셋: lwf 데이터 (face recognition DB)
- 데이터: 7명의 얼굴 사진, 총 1288장, 각 얼굴 사진 크기: 50x37 픽셀

↳ 지금은 딥러닝에 활용하기 매우 적은 양

- 데이터 로더

```python
##########얼굴인식(Face recogniton) 예제##########
from sklearn.datasets import fetch_lfw_people  #data set 불러오기
faces=fetch_lfw_people(min_faces_per_person=70,resize=0.4)
n_samples, h, w = faces.images.shape
X=faces.data
y=faces.target

print(faces.target_names)
print(faces.images.shape)
```

→ 라벨 (y=faces.target)

→ 7명의 이름 (faces.target_names)

□.data ⇒ X

□.target ⇒ Y

```
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
 'Gerhard Schroeder' 'Hugo Chavez' 'Tony Blair']
(1288, 50, 37)
```

사진개수 가로 세로 길이

비선형 SVM 응용 #1 (분류)

- 실습 코드: 링크
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔ (0,1,2,3,4,5,6)

- 데이터 분할, 데이터 압축
  - 분할: (학습:테스트) = (1:3) 비율로 Split
  - 압축: PCA Manual, PCA 클래스 호출
    - whiten➔ feature간 상관관계 최소화, svd_solver➔Randomized SVD 사용

```
[2]  1 # training set과 test set으로 데이터 나누기 --> 75:25의 비율
     2 from sklearn.model_selection import train_test_split
     3 X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

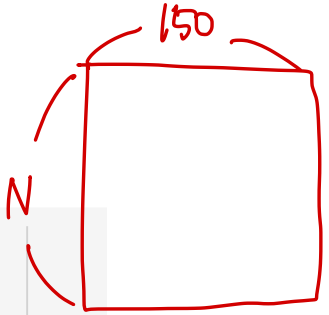
```
[4]  1 #주성분 분석을 이용해 150개의 특성변수에 대한 차원 축소
     2 #Randomized PCA사용  : 8장에서 자세하게 다룸.
     3 from sklearn.decomposition import PCA
     4 n_components=150
     5 pca=PCA(n_components=n_components, svd_solver='randomized',whiten=True).fit(X_train)
     6 eigenfaces=pca.components_.reshape(n_components,h,w)
     7 X_train_pca=pca.transform(X_train)  #주성분의 training data
     8 X_test_pca=pca.transform(X_test)    #주성분의 test data
```

PCA → 데이터 차원 ↓

50x37 = 1850
↳ 너무 큼
↳ PCA 사용

비선형 SVM 응용 #1 (분류)

- GridSearchCV API: Manual, SVM API: Manual
  - GridSearchCV: 파라미터를 변경하면서 가장 좋은 성능의 분류기를 찾는 방법
    - param_grid: 다양하게 실험할 파라미터
      - gamma=rbf의 설정 파라미터, C=에러허용정도
    - class_weight : unbalanced class data 를 위해 사용




```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
#커널 SVM을 이용 & 커널함수:방사형기저함수(kernel='rbf')를 사용
#방사기저함수의 최적화 gamma와 완화변수의 허용정도 c를 찾기위해 'GridSearchCV' 모듈 사용
param_grid={'C':[1e3,5e3,1e4,5e4,1e5],
            'gamma':[0.0001,0.0005,0.001,0.005,0.01,0.1]}
clf=GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid, cv=5)
clf.fit(X_train_pca,y_train)
##  프로그램 실행시 약 1분정도 시간 소요됨.
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight='balanced', coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [1000.0, 5000.0, 10000.0, 50000.0, 100000.0],
                         'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

비선형 SVM 응용 #1 (분류)

- GridSearchCV API: Manual
  - GridSearchCV: 파라미터를 변경하면서 최적의 파라미터 확인 법
    - best_params_ : GridSearchCV 로 추정된 파라미터만
    - best_estimator_ : SVC 모든 정보

```
1 print(clf.best_params_)            # 추정 parameter
2 print(clf.best_estimator_)
```

```
{'C': 1000.0, 'gamma': 0.005}
SVC(C=1000.0, break_ties=False, cache_size=200, class_weight='balanced',
    coef0=0.0, decision_function_shape='ovr', degree=3, gamma=0.005,
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

비선형 SVM 응용 #1 (분류)

- Accuracy, Precision, Recall, F1 평가 지표
  - Metrics.classification_report 사용

```
#학습된 결과를 시험데이터에 적용하여 정확도, recall, 그리고 f1을 제공하는 classification_report
y_fit=clf.predict(X_test_pca)

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix        #confusion matrix를 출력
print(classification_report(y_test,y_fit,target_names=faces.target_names))
```

```
                   precision    recall  f1-score   support

     Ariel Sharon       0.86      0.46      0.60        13
     Colin Powell       0.79      0.87      0.83        60
  Donald Rumsfeld       0.82      0.67      0.73        27
    George W Bush       0.84      0.98      0.91       146
Gerhard Schroeder       0.95      0.80      0.87        25
      Hugo Chavez       1.00      0.53      0.70        15
       Tony Blair       0.96      0.75      0.84        36

         accuracy                           0.85       322
        macro avg       0.89      0.72      0.78       322
     weighted avg       0.86      0.85      0.84       322
```