5. SVM

March 28, 2022

[1]:
```
# Import Libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

[2]:
```
# Load dataset
data = load_breast_cancer(as_frame=True)

X=data.data
y=data.target
print(X.shape)
#X.head(5)
```

```
(569, 30)
```

[3]:
```
#Split data into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

[4]:
```
from sklearn.svm import SVC
clf = SVC(kernel='rbf')
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
cm = np.array(confusion_matrix(y_test,y_pred))
cr = classification_report(y_test,y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.93      0.72      0.81        39
           1       0.87      0.97      0.92        75

    accuracy                           0.89       114
   macro avg       0.90      0.85      0.86       114
weighted avg       0.89      0.89      0.88       114
```

[5]:
```
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
clf = make_pipeline(StandardScaler(), SVC(kernel='rbf'))
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
cm = np.array(confusion_matrix(y_test,y_pred))
cr = classification_report(y_test,y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        39
           1       0.99      0.97      0.98        75

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

[6]:
```
n_samples, n_features = X_train.shape

w = np.zeros(n_features)
b = 0

X_train = np.array(X_train)
y_train = np.where(y_train <= 0, -1, 1)   # replace 0 with -1 in y
y_test = np.where(y_test <= 0, -1, 1)   # replace 0 with -1 in y
```

[7]:
```
C = 1.0

for t in range(1, 2000):

    # set learning rate
    lr = 1/t

    subgrad_w = 0
    subgrad_b = 0

    # sum over all subgradients of hinge loss for a given samples x,y
    for x_i, y_i in zip(X_train,y_train):
        f_xi = np.dot(w.T,x_i) + b

        decision_value = y_i * f_xi

        if decision_value < 1:
```

```
            subgrad_w += - y_i*x_i
            subgrad_b += -1 * y_i
        else:
            subgrad_w += 0
            subgrad_b += 0

    # multiply by C after summation of all subgradients
    subgrad_w = C * subgrad_w
    subgrad_b = C * subgrad_b

    # update weights
    w = w - lr * (w + subgrad_w)

    # update bias
    b = b - lr * subgrad_b

w,b
```

[7]:
```
(array([ 7.46053987e+01,  5.64042771e+01,  4.14809900e+02,  8.70940470e+01,
         5.86110655e-01, -8.26001336e-01, -2.18585458e+00, -8.95062552e-01,
         1.07747129e+00,  5.05372781e-01,  2.85252826e-01,  2.10156898e+00,
        -3.17065708e+00, -1.73843033e+02, -3.32293647e-04, -3.19419419e-01,
        -4.80943407e-01, -9.11417369e-02, -1.36056913e-02, -1.86474563e-02,
         8.09719640e+01,  6.50957829e+01,  4.25724447e+02, -1.71935318e+02,
         6.92658439e-01, -2.82349504e+00, -4.85699592e+00, -1.19453153e+00,
         1.07151891e+00,  3.49161856e-01]),
 555.9127508026048)
```

[8]:
```
y_pred = np.sign(np.dot(X_test,w)+b)
print(y_pred)
```

```
[ 1. -1. -1.  1.  1.  1. -1.  1. -1.  1.  1.  1.  1. -1.  1. -1.  1.  1.
  1. -1.  1. -1.  1.  1.  1.  1. -1.  1. -1.  1. -1.  1.  1.  1.  1.  1.
 -1.  1.  1. -1.  1.  1.  1.  1.  1.  1.  1.  1. -1.  1.  1.  1. -1.  1.
  1. -1.  1.  1.  1.  1.  1. -1. -1.  1.  1. -1.  1.  1.  1.  1.  1.  1.
 -1.  1.  1.  1.  1. -1.  1.  1.  1. -1. -1.  1. -1.  1.  1.  1.  1.  1.
  1.  1.  1. -1.  1.  1.  1. -1.  1. -1.  1. -1.  1.  1.  1.  1.  1. -1.
 -1.  1.  1.  1. -1.  1.]
```

[9]:
```
#acc = (y_pred == y_test).count()
#print("Accuracy: ",acc/X.shape[0])
from sklearn.metrics import classification_report, confusion_matrix
cm = np.array(confusion_matrix(y_test,y_pred))
cr = classification_report(y_test,y_pred)
print(cr)
```

```
              precision    recall  f1-score   support
```

```
              -1       0.94      0.74      0.83        39
               1       0.88      0.97      0.92        75

        accuracy                           0.89       114
       macro avg       0.91      0.86      0.88       114
    weighted avg       0.90      0.89      0.89       114
```

[ ]: