CSE4020

Lab – 4

Routu Venkat Sai Sampath
19BCE0238

Aim: Perform Linear SVM on the given data and produce the line equation.

Procedure:

1. Get the (x,y) value pairs from the data set classified as negative and positive samples.
2. Now, work on calculating the SVM line. The algorithm to calculate linear SVM is
    a. First take the nearest positive and negative values. Let the number of such values be N.
    b. Now there will be N support vectors, so model each vector by adding a bias into its coordinates.
    c. Now we will need to solve the following linear problem. Here $a_{n,n}$(on the left) refers to the dot product of two support vectors. The x values are the alpha values that are needed to calculate the weight of the SVM. And finally, the right side a values are either 1 or -1 based on if the sample is positive or negative.

    $$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,n+1} \\ a_{2,n+1} \\ a_{3,n+1} \\ \cdots \\ a_{n,n+1} \end{bmatrix}$$

    d. Now we will apply the following formula to calculate the values for weights.

    $$\tilde{w} = \sum_i \alpha_i \tilde{s}_i$$

    e. Once the weights are calculated fit the values from the weight's matrix into ax + by + c = 0 and the linear SVM will be generated.
3. We got the hyperplane

Aim: Perform Nonlinear SVM on the given data and find the equation of the polynomial.

Procedure

1. Get the (x,y) value pairs from the data set classified as negative and positive samples.
2. Now in nonlinear svm, we need to scale data based on their values. The rule for scaling is given as

$$\Phi_1\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 4 - x_2 + |x_1 - x_2| \\ 4 - x_1 + |x_1 - x_2| \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} > 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases}$$

3. Once the data is scaled based on the above equation, we can perform linear SVM on it.
   a. First take the nearest positive and negative values. Let the number of such values be N.
   b. Now there will be N support vectors, so model each vector by adding a bias into its coordinates.
   c. Now we will need to solve the following linear problem. Here $a_{n,n}$(on the left) refers to the dot product of two support vectors. The x values are the alpha values that are needed to calculate the weight of the SVM. And finally, the right side a values are either 1 or -1 based on if the sample is positive or negative.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,n+1} \\ a_{2,n+1} \\ a_{3,n+1} \\ \cdots \\ a_{n,n+1} \end{bmatrix}$$

   d. Now we will apply the following formula to calculate the values for weights.

$$\tilde{w} = \sum_i \alpha_i \tilde{s}_i$$

   e. Once the weights are calculated fit the values from the weight's matrix into ax + by + c = 0 and the linear SVM will be generated.

Data used

linear_19BC... Open with Microsoft Excel

x1	x2	class
3	1	positive
3	-1	positive
6	1	positive
6	-1	positive
1	0	negative
0	1	negative
0	-1	negative
-1	0	negative

non_linear_1... Open with Microsoft Excel

x1	x2	class
2	2	positive
2	-2	positive
-2	2	positive
-2	-2	positive
1	1	negative
1	-1	negative
-1	1	negative
-1	-1	negative

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.svm import SVC
import math
import matplotlib.pyplot as plt
```

In [2]:

```
linear_df = pd.read_csv("linear_19BCE0238.csv")
non_linear_df = pd.read_csv("non_linear_19BCE0238.csv")
```

In [3]:

```
X_linear = linear_df.iloc[:,:2]
y_linear = linear_df.iloc[:,-1]

X_non_linear = non_linear_df.iloc[:,:2]
y_non_linear = non_linear_df.iloc[:,-1]
```

In [4]:

```
def compute_linear_SVM(X, y):
    classifier = SVC(kernel='linear', C = 1.0)
    classifier.fit(X, y)

    x_component = round(classifier.coef_[0][0],2)
    y_component = round(classifier.coef_[0][1],2)
    bias = round(classifier.intercept_[0], 2)

    print("{0}*X + {1}*Y + {2}= 0".format(x_component, y_component, bias))

    return (classifier,x_component,y_component,bias)
```

In [5]:

```
print("Line Equation for linear SVM")
linear_classifier = compute_linear_SVM(X_linear,y_linear)
```

```
Line Equation for linear SVM
1.0*X + -0.0*Y + -2.0= 0
```

In [6]:

```
def scale(tup):
    x1, x2 = tup
    if(math.sqrt((x1**2)+(x2**2)) > 2):
        return ((4-x2 + abs(x1-x2), 4-x1 + abs(x1-x2)))
    else:
        return ((x1,x2))

def compute_non_linear_SVM(X, y):
    data = X.copy()
    for _ in range(X.shape[0]):
        data.iloc[_,:] = scale(tuple(data.iloc[_,:]))
    values = compute_linear_SVM(data, y)
    return values
```

In [7]:

```
print("Line Equation for non linear SVM")
non_linear_classifier = compute_non_linear_SVM(X_non_linear, y_non_linear)
```

```
Line Equation for non linear SVM
1.0*X + 1.0*Y + -3.0= 0
```