INF552 Machine Learning for Data Informatics HW6

1. Contribution:

a. Shih-Yu Lai (30%): (Monday)

Email: shihyula@usc.edu

USC-ID: 7183984563

Write the ReadMe.

b. Shiuan-Chin Huang (35%): (Monday)

Email: shiuanch@usc.edu

USC-ID: 9815633745

Implement the non-linear SVM.

c. Danhui Wu (35%): (Tuesday)

Email: danhuiwu@usc.edu

USC-ID: 8038800437

Implement the linear SVM.

2. Steps of the algorithm and data structure

Linear_SVM

Step 1: Load the data from linsep.txt and split the data into two numpy array. One is store the 2D coordinates of a point and the other is store the classification label.

Step 2: We use cvxopt help us to calculate the matrix and follow the linear SVM formula to find weights, intercept and support vectors.

Step 3: Print out the equation, picture and support vectors.

Non-linear SVM

Step 1: Load the data from nonlinsep.txt and split the data into two numpy array. One is store the 2D coordinates of a point and the other is store the classification label.

Step 2: We use cvxopt help us to calculate the matrix and follow the non-linear SVM formula, and use polynomial kernel (*k(x,y)=(<x,y>+c)^2*) as our kernel function.

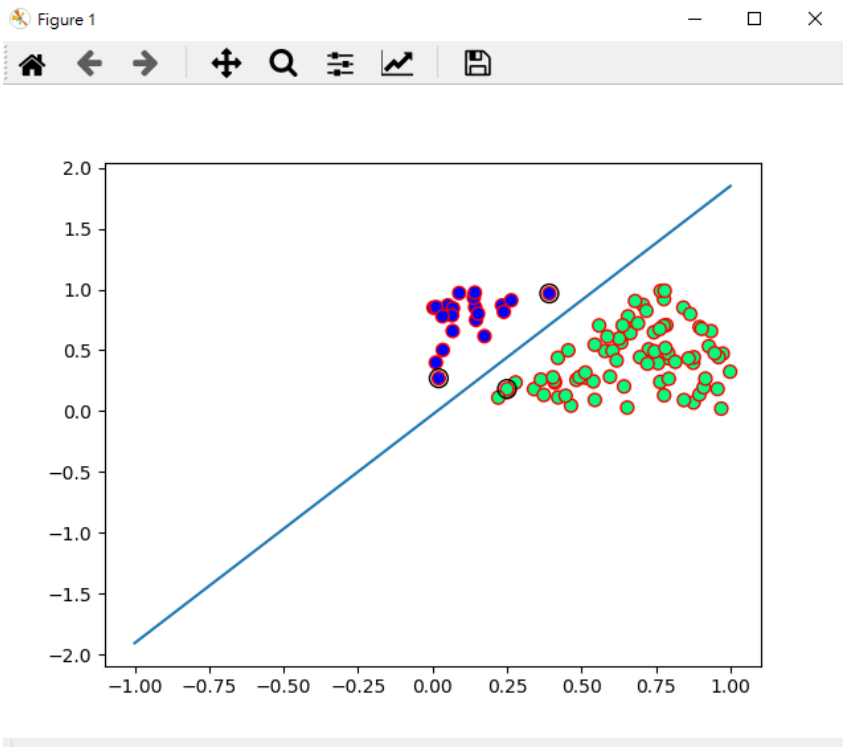
Step 3: Print out the kernel function, picture and support vectors.

3. Our Outcome

SVM_linear.py

It will output the weights, intercept, support vectors and the picture of the line.

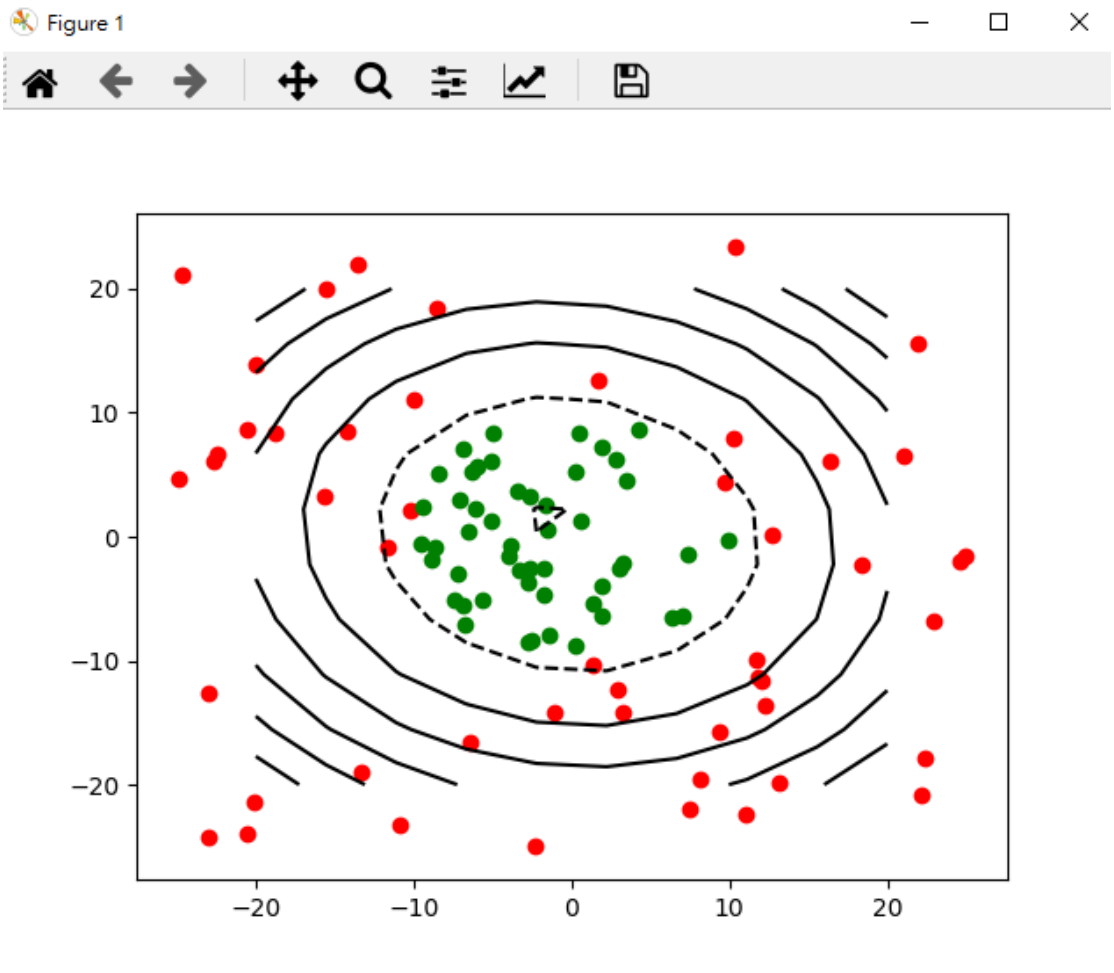
```
     pcost       dcost       gap    pres   dres
 0: -2.0636e+01 -4.3905e+01  3e+02  2e+01  2e+00
 1: -2.2372e+01 -3.7202e+01  9e+01  5e+00  5e-01
 2: -2.3112e+01 -3.8857e+01  5e+01  2e+00  2e-01
 3: -2.8318e+01 -3.3963e+01  1e+01  4e-01  4e-02
 4: -3.2264e+01 -3.3927e+01  2e+00  1e-02  1e-03
 5: -3.3568e+01 -3.3764e+01  2e-01  1e-03  1e-04
 6: -3.3737e+01 -3.3739e+01  2e-03  1e-05  1e-06
 7: -3.3739e+01 -3.3739e+01  2e-05  1e-07  1e-08
 8: -3.3739e+01 -3.3739e+01  2e-07  1e-09  1e-10
Optimal solution found.
weights: [[ 7.2500563 ]
 [-3.86188924]]
intercept: [-0.10698734]
support vectors: [[0.24979414 0.18230306]
 [0.3917889  0.96675591]
 [0.02066458 0.27003158]]
```



nonlin.py

It will output the intercept, the support vectors and the picture.

```
     pcost       dcost       gap    pres   dres
 0:  1.1435e-01 -1.5013e+02  2e+02  4e-17  4e-16
 1:  2.9541e-02 -1.6982e+00  2e+00  2e-16  1e-15
 2:  1.0180e-03 -1.7253e-02  2e-02  2e-16  7e-16
 3:  1.0214e-05 -1.7251e-04  2e-04  2e-16  4e-16
 4:  1.0214e-07 -1.7251e-06  2e-06  2e-16  4e-16
 5:  1.0214e-09 -1.7251e-08  2e-08  2e-16  5e-16
Optimal solution found.
15 Support Vectors
alphas:
[1.9609041556564035e-11, 2.414362320538459e-11, 2.02905002307524e-11, 1.5638839728667032e-11, 2.072439027040429e-11, 1.650286409345699e-11, 2.997908434220335e-11, 2.014934673176233e-11, 3.0498911299464e-11, 2.9126426103412135e-11, 1.1191162424698688e-10, 7.971573081508204e-11, 2.4318729209944475e-11, 2.4663881957197414e-11, 1.6304466304783477e-11]
intercept:
-4.140713156919734e-06
```



4. Challenge

The SVM algorighm is not too hard, but while we try to coding and implement it, how to use cvxopt takes us some time to figure out. Also, while we implement the non-linear SVM, understanding which kernel to choose and how to draw the circle graphic are also challenges for us.

5. Optimization

In the linear SVM, we use cvxopt to resolve the quadratic programming and use numpy array to calculate.

In the non-linear SVM, we use polynomial kernel. And after we tried many times, we use alpha > 1.5e-11 to prevent overfitting and got the better outcome.

6. Software Familiarization

sklearn.svm.LinearSVC:

This library saves people lots of time to use SVM algorithm. Similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

coef_ :

Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel.

intercept_ :

Constants in decision function.

classes_:

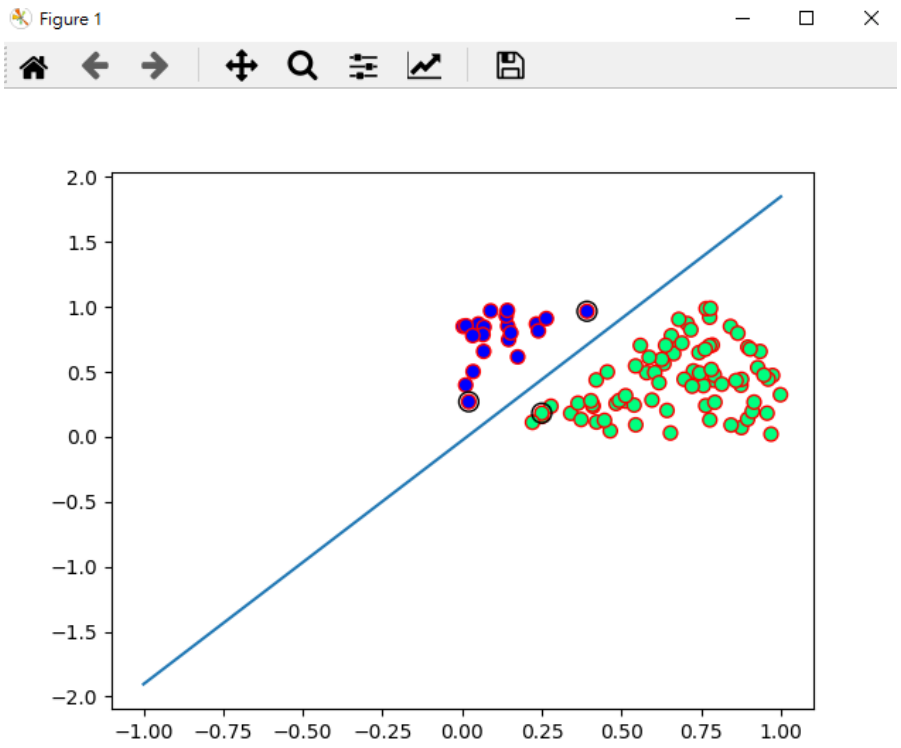
The unique classes labels.

n_iter_ :

Maximum number of iterations run across all classes.

Result:

```
w =  [[ 7.24837069 -3.86099178]]
b =  [-0.10703977]
Indices of support vectors =  [83 87 27]
Support vectors =  [[0.3917889  0.96675591]
 [0.02066458 0.27003158]
 [0.24979414 0.18230306]]
Number of support vectors for each class =  [2 1]
Coefficients of the support vector in the decision function =  [[ 1.29438449 32.43652457 33.73090907]]
```



7. Applications

1. Face Reconstruction:

   Can classify parts of the image as face or non-face and create the square to show which one is face.

2. Bioinformatics:

   Can identify the classification of genes, patients and so on. It includes protein classification and cancer classification.

3. Text and hypertext categorization:

   Use training data to classify documents into different categories. Ex: classification of news articles

4. Handwriting recognition:

   Recognize the handwritten characters.