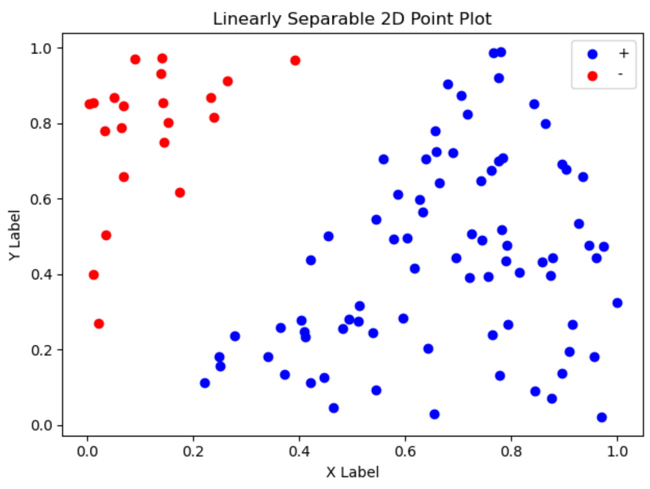
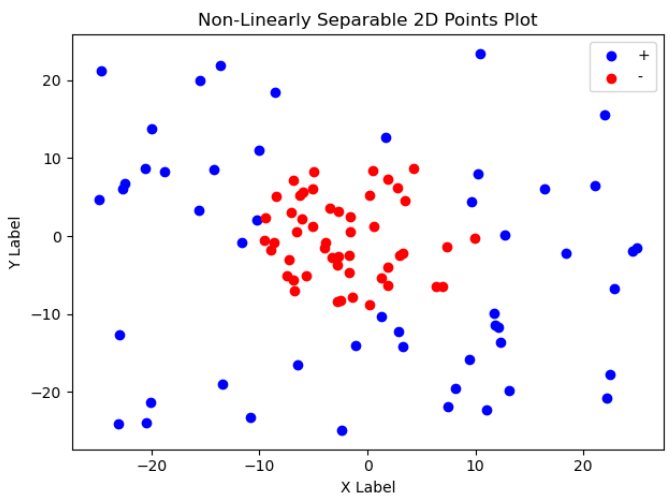
Name: Kushal Sharma
Email: kushals@usc.edu
Course: INF 552 Summer 2020
Assignment: HomeWork 6 (Support Vector machine)

Support Vector Machine:

We are given the linearly separable as well as non linearly separable 2D data points along with their labels, positive or negative. We need to find the fattest margin line or contour which can classify these points based on positive or negative labels. Data points in 2D space are plotted as below, red for negative label and blue for positive label points.





Linearly Separable case:
For linearly separable points, the linear kernel function or simple form of SVM is used.

Equation of the line is in format of

$$\mathbf{W^T.X + b = 0}$$

The objective function is as follows:

$$\min_{w,b} \|w\|^2$$

such that

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, \text{for all } i$$

The objective function in Lagrangian dual form is given as below. Using **cvxopt** qudratic programming solver, optimal solution is found in 8 iterations.

$$\max_{\lambda \geq 0} -\frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j y_i y_j x^{(i)^T} x^{(j)} + \sum_i \lambda_i$$

such that

$$\sum_i \lambda_i y_i = 0$$

The quadratic programming solver works as:

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

The linear kernel function is given as:

np.dot(x1, x2)

The value of lambda is coming as low as 10^{-9} or 10^{-10} for some instances but not coming up as exactly zero. So we took the threshold value below which the value of lambda is considered 0. We considered threshold value to be 10^{-5} (or, 1e-5).

We found 3 support vectors for which the value of lambda > 0 and their corresponding indices as mentioned below.

Support Lagrangian multiplier:
[33.73875192 1.29468506 32.4440672]

Support index: [27, 83, 87]

Support Vectors:

```
 [[ 0.24979414  0.18230306  1.        ]
 [ 0.3917889   0.96675591 -1.        ]
 [ 0.02066458  0.27003158 -1.        ]]
```

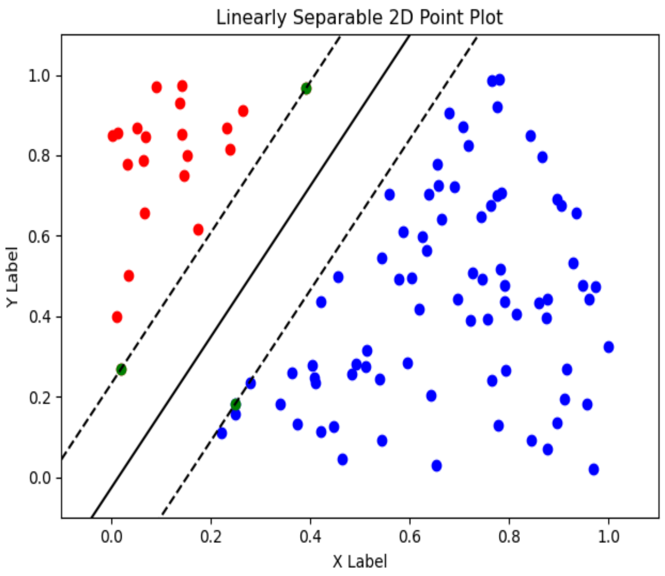
Weights and biases are given as:
weights: [7.25005616 -3.86188932]

bias: -0.10698729032461896

Thus the equation of the line is given as below:

7.25x -3.86y -0.11 = 0

3 support vectors are denoted in green color on the plot.



There is no miss-classified point.

Non-Linearly Separable Case:

The non-linear pattern in original space is transformed in higher dimension space to make it linearly separable. The higher dimension coordinated are defined by some function of coordinates in original space. Thus the objective function to linearly classify points in higher dimension space becomes-

$$w^T.\Phi(x) + b = 0$$

Objective function becomes:

$$\begin{aligned} &\min\ \mathcal{P}(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^2 \\ &\text{subject to } \forall i \quad y_i(\mathbf{w}^\top \Phi(\mathbf{x}_i) + b) \geq 1 \end{aligned}$$

The objective function in Lagrangian dual form is given as below. Using **cvxopt** quadratic programming solver, optimal solution is found in 15 iterations.

$$\max\ \mathcal{D}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i\alpha_i\, y_j\alpha_j\, \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$$

$$\text{subject to} \quad \begin{cases} \forall i \quad \alpha_i \geq 0, \\ \sum_i y_i\alpha_i = 0. \end{cases}$$

$$\hat{y}(\mathbf{x}) = \mathbf{w}^{*\top}\mathbf{x} + b^* = \sum_{i=1}^{n} y_i\alpha_i^*\, \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}) + b^*$$

The quadratic programming solver works as:

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

The polynomial kernel function used for non-linear case is given as K =

(np.dot(x1, x2)) ** 2

The value of alpha is coming as low as 10^{-9} or 10^{-10} for some instances but not coming as exactly zero. So we take the threshold value, below which the value of alpha is considered 0. We considered threshold value to be 10^{-5} (or, 1e-5).

We found 4 support vectors for which the value of alpha > 0 and their corresponding indices as mentioned below.

Support Lagrangian multiplier:
[0.02738944 0.00661983 0.01234583 0.02166343]

Support index: [36, 51, 59, 95]

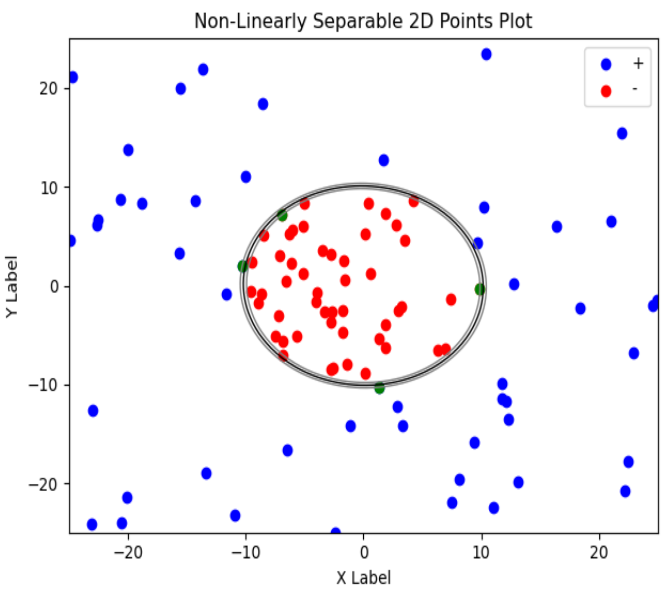
Support Vectors:

```
 [[-10.260969     2.07391791   1.        ]
 [  1.3393313  -10.29098822   1.        ]
 [ -6.90647562   7.14833849  -1.        ]
 [  9.90143538  -0.31483149  -1.        ]]
```

bias: -18.93152733913999

The equation of line is given as :
0.18x2 +0.19y**2 +0.01xy -18.93 = 0**

4 support vectors are denoted in green color on the plot.



There is no miss-classified point.

For the soft margin SVM, where we allow some miss-classification to be present in in-sample, the objective function in singular as well as dual form is given as below:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \mathcal{P}(\mathbf{w},b,\boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to} \ \begin{cases} \forall i \quad y_i(\mathbf{w}^\top\Phi(\mathbf{x}_i)+b) \geq 1-\xi_i \\ \forall i \quad \xi_i \geq 0 \end{cases}$$

C controls the compromise between large margins and small margin violations.

$$\max \ \mathcal{D}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i\alpha_i\, y_j\alpha_j\, K(\mathbf{x}_i,\mathbf{x}_j)$$
$$\text{subject to} \ \begin{cases} \forall i \quad 0 \leq \alpha_i \leq C \\ \sum_i y_i\alpha_i = 0 \end{cases}$$

Data Structure:

- Numpy array is used to do mathematical operations like dot product, outer product.
- Numpy meshgrid and linespace are used to take multiple points within a range to plot the contour of the curve.
- Cvxopt matrix is used to provide input to quadratic programming solver in required format.

Code Level optimization:

- Numpy functions are used to calculate the Q matrix which NxN matrix of ($y_iy_jx_ix_j$) each term instead of doing it in iterations. Results were faster to achieve.

Challenges:

- Plotting the contour for polynomial kernel function was bit challenging.
- Reporting the equation of line or curve involved some thinking process as I had to first break the whole equation form, then calculate the coefficients of individual variable term independently and then combine them together to print or write into file the whole equation.

<u>Part 2 – Software Familiarization:</u>

Scikit-learn library provides methods for classification based on support vector machine using linear, polynomial or RBF (radial based function) kernel functions.

Linear separable case:

We can have two option of carrying out the classification using sklearn library finctions.

```
svc = SVC(kernel='linear', C=1000)
&
svc = LinearSVC(C=1000)
```

We need to make sure to keep value of C high enough to allow no or lesser no. of miss-classification in in-sample.
The results obtained matches with results obtained from our program.

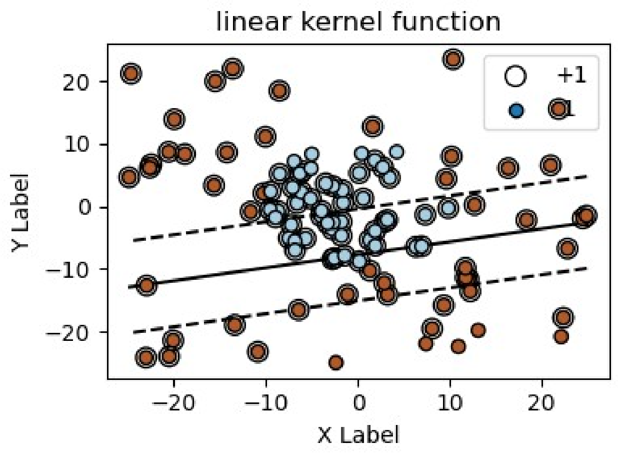
Sklearn Results:

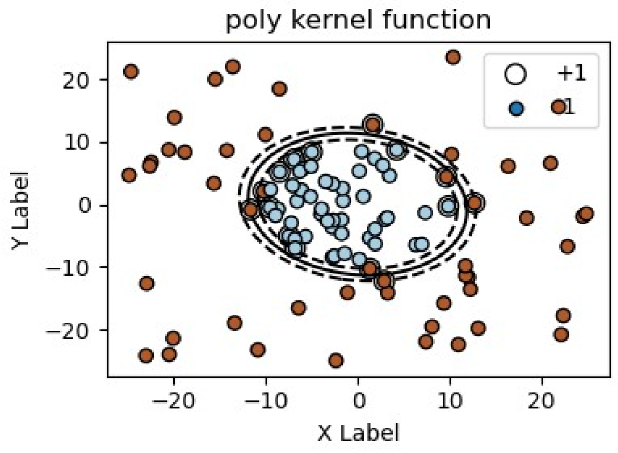
```
sklearn-weights: [ 7.24837069 -3.86099178]

sklearn-bias: -0.10703977170718931

sklearn-support-indices: [83 87 27]

sklearn-support-vectors:
 [[0.3917889  0.96675591]
 [0.02066458 0.27003158]
 [0.24979414 0.18230306]]

sklear-confusion-matrix:
 [[21  0]
 [ 0 79]]

sklearn-classification-report:
              precision    recall  f1-score   support

        -1.0       1.00      1.00      1.00        21
         1.0       1.00      1.00      1.00        79

    accuracy                           1.00       100
   macro avg       1.00      1.00      1.00       100
weighted avg       1.00      1.00      1.00       100
```

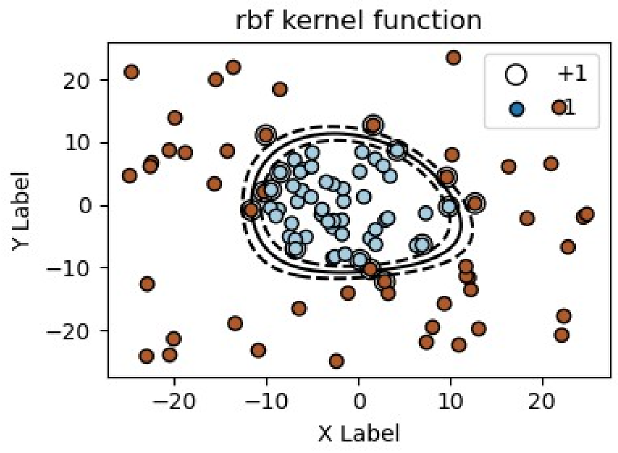
Non-Linear separable case:

For non-linear case, we tried all types of kernel functions in Support Vector Classifier (SVC) – linear, poly and rbf. With high value of C (regularization parameter which tells the allowance of soft margin), we can achieve the high accuracy in poly and rbf. Degree of the polynomial kernel function also play a role.

C= 10, degree = 2:
linear kernel score: 0.69
poly kernel score: 0.96
rbf kernel score: 0.98







Part 3 - Applications

SVM as classifier shares the similar field of application in real world as other classifiers. However, due to high accuracy result and ability to transform into higher dimension space (upto infinite dimension space with rbf), it is well suited for applications which have high penalty for risk of failure. Following are some of the classification application of SVM.

1. Face Detection: SVM classify parts of image as a face and non-face and create a square boundary around the face.

2. Text and hypertext categorization: SVM classify documents into different categories on the basis of score generated and then compare with threshold value.

3. Image Classification: SVMs provide better search accuracy in comparison to traditional query-based searching techniques.

4. Bioinformatics: SVM is used for protein classification, cancer classification, identify classification of genes, patients on the basis of genes and other biological problems.

5. Protein fold and remote homology detection: SVM is used for protein remote homology detection which depends on how the protein sequence modeled. The method used to compute the kernel function between them.

6. Handwritten recognition: SVM to recognize hand-written characters that use for data entry and validating signatures on documents.

7. Generalized predictive controls: SVM based GPC are used to control chaotic dynamics with useful parameter. The system follows chaotic dynamics with respect to the local stabilization of the target.