Programming Assignment 6 Report

PART 1: IMPLEMENTATION

Support Vector Machines on Linearly Separable Data:

Data Structures:

- Cvxopt: We utilized the matrix and solver from the cvxopt library in order to format the data into a quadratic programming problem and solve it.
- Pandas: We used pandas to read in the input data coordinates as well as their labels.
- Numpy: We converted the data frame structures into numpy arrays so that it would be easier to perform mathematical operations like calculating the dot product and formatting them into matrices for the cvxopt library.

Code-level optimizations:

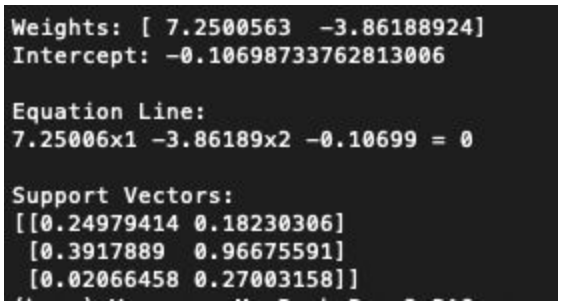
- We reshaped the numpy arrays for the alphas, labels, and coordinates so that we could just easily multiply them together and get their sum instead of using a for loop and iterating through each point when calculating the weights vector.
- We utilized the matrix class from the cvxopt library to easily represent the different components of the quadratic programming problem. The solver also made it easy to find the alphas in the solution.
- We used the numpy.where() function to easily get the indices of the alpha values that were greater than 0.00001.
- To calculate the intercept of the equation, we just took one of the alphas that was greater than 0.00001 and plugged in the weight vector found previously with the alpha's corresponding coordinate and label.
- We found the support vectors by just using the coordinates that had corresponding alphas greater than 0.00001.

Challenges:

- It was kind of confusing to represent the data as matrices for the cvxopt solver initially. We had to understand the dimensions of each input matrix.
- We had to make sure that the dimensions of the alphas, data coordinates, and labels were correct so that they could be easily multiplied together for their sum when calculating the weight vector.

Output:

- To run the program: python LinearSVM_WithoutLibrary.py



- Equation: $7.25006\,x_1 - 3.86189\,x_2 - 0.10699 = 0$
- Support Vectors:

[[0.24979414 0.18230306]
[0.3917889 0.96675591]
[0.02066458 0.27003158]]

Support Vector Machines on Non-Linearly Separable Data:

Data Structures:

- Cvxopt: We utilized the matrix and solver from the cvxopt library in order to format the data into a quadratic programming problem and solve it.
- Pandas: We used pandas to read in the input data coordinates as well as their labels.
- Numpy: We also stored the transformed coordinates for the kernel in a 2D numpy array. We converted the data frame structures into numpy arrays so that it would be easier to perform mathematical operations like calculating the dot product and formatting them into matrices for the cvxopt library.

Code-level optimizations:

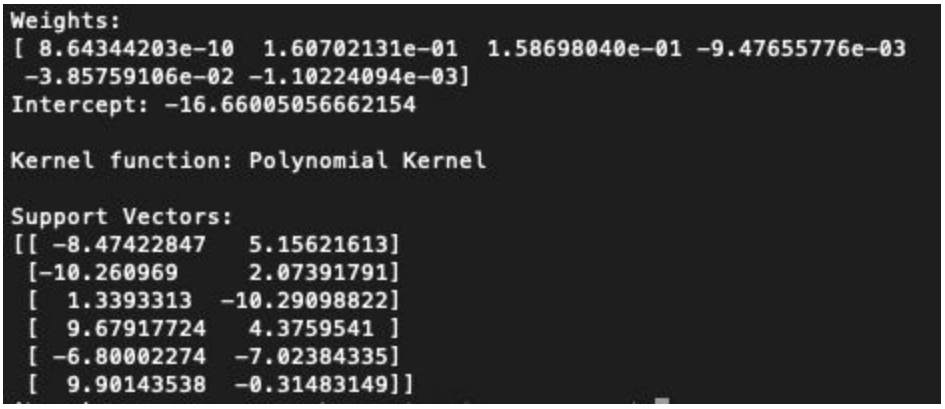
- We utilized the polynomial kernel for transforming the nonlinear data. We converted each data point into a vector for the kernel function with the form $[1, x^2, y^2, \sqrt{2}x, \sqrt{2}y, \sqrt{2}xy]$. And in order to get the Q matrix for the quadratic problem, we multiplied the outer product of the label vectors with the dot product of each data vector with one another.
- To calculate the weight vector, we just used the alphas that were greater than 0.00001. We took the sum of these alphas multiplied by their corresponding labels and coordinates.
- We found the support vectors by just using the coordinates that had corresponding alphas greater than 0.00001.

Challenges faced:

- It was kind of difficult understanding how to represent the kernel function by transforming the data coordinates. It made more sense when we had to take the dot product of these vectors in order to get the Q matrix.
- The intercept was a bit more difficult to calculate for the nonlinear data as well since we had to subtract the sum of the product of the transformed data for the kernel with the alphas and labels from a specific label.

Output:

- To run the program: python NonlinearSVM_WithoutLibrary.py



- Kernel function: Polynomial Kernel
- Support Vectors:
[[-8.47422847 5.15621613]

[-10.260969 2.07391791]
[1.3393313 -10.29098822]
[9.67917724 4.3759541]
[-6.80002274 -7.02384335]
[9.90143538 -0.31483149]]

PART 2: SOFTWARE FAMILIARIZATION

Support Vector Machines on Linearly Separable Data:

Libraries

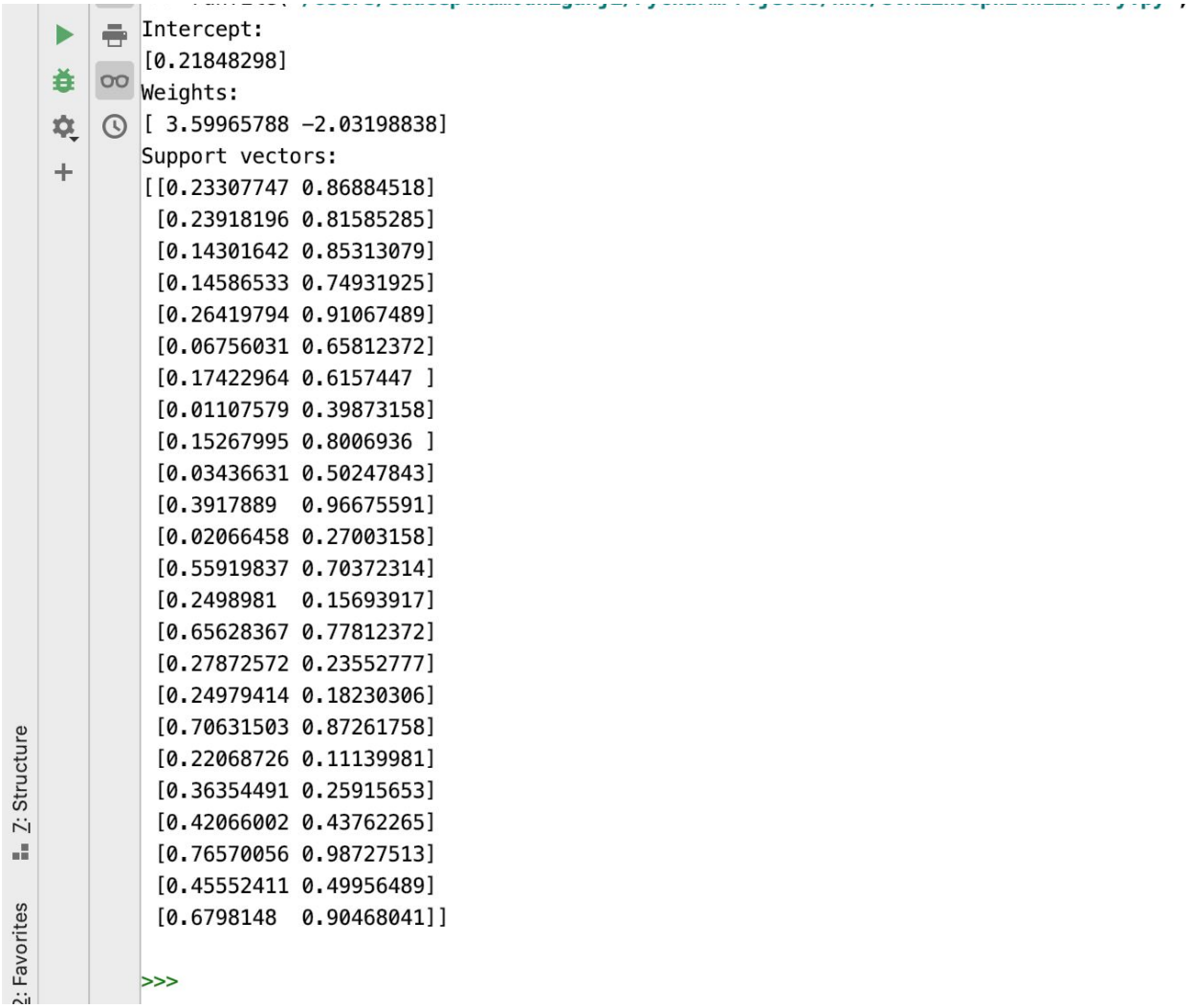
- Sklearn: sklearn provides many machine learning algorithms. We have used it to import svm package which provides a Support Vector Machine Classification algorithm through its SVC class. We have used the kernel to be of linear type as we were classifying linearly separable data.
- Pandas: We have used pandas to read the input data which is a csv file into a dataframe.
- Numpy: We have further converted the above dataframe into an array data structure by using numpy. We separated the input data into a two arrays, one having the set of 2D points as an array of an array and the other having the classification labels as a single array.
- Matplotlib.pyplot: We have used matplotlib.pyplot to plot the support vectors and the margin line separating the points.

Implementation Comparison

- In the implementation without libraries, the input points and labels were read into a dataframe as 3 different columns and then separated into labels and data. This data was later used to find the maximum margin line with three nearest support vectors. We used quadratic program solvers from cvxopt for this purpose in our program.
- In the implementation with libraries, the input data was read using numpy and then separated and then fitted to a SVC class instance with a linear kernel. We have then used this kernel to directly find the intercept and weights of one of the margin lines along with the support vectors. These points were then plotted using matplotlib.

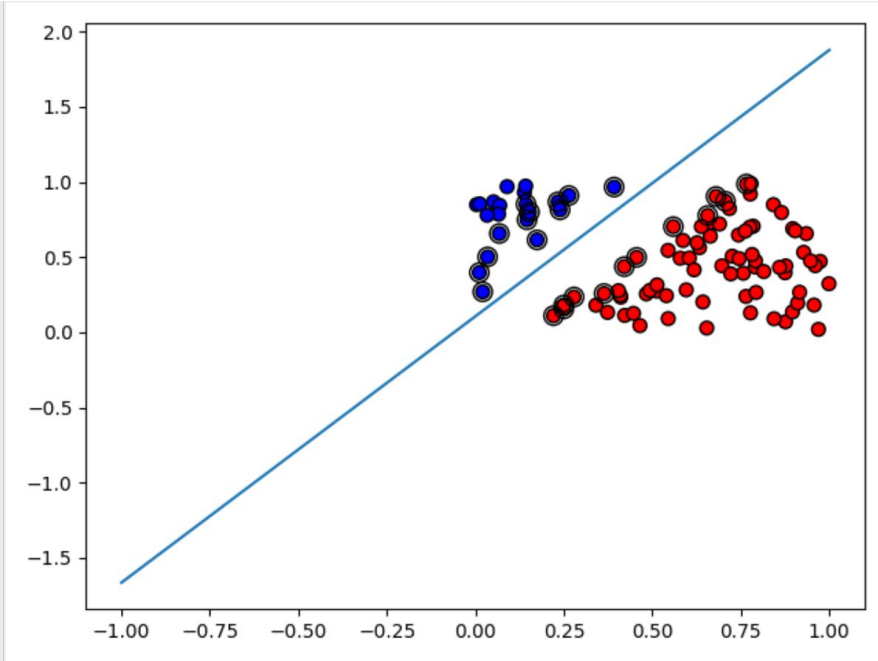
Result Comparison

- The line equation with support vectors obtained by using the library function is as follows:



The results obtained are different from the result obtained via the implementation without libraries. This is because we calculated the maximum margin line in the without libraries implementation whereas we calculated only one of the margin lines and all its support vectors in the with libraries implementation.

- The plot obtained is as follows:



Possible Improvements

- We can use different kernels based on the type of data we have and what kind of information we plan to obtain from the data can improve classification accuracy and performance.
- We can check for alphas that are greater than 0.0001 when calculating our weight vector instead of just multiplying them all and getting their sum. Since most alphas will be 0, their values would not affect the end sum, so we could make our calculation of the weight vector in the linearly separable data more efficient by only performing the computations with alphas greater than 0.00001.
- Varying the type and form of the parameters, like for parameters A&B, using A as a floating point integer, using A/B instead of using them separately might give a better performance in some other datasets.

<u>Support Vector Machines on Non-Linearly Separable Data:</u>

Libraries

- Sklearn: We have used sklearn to import svm package which provides a Support Vector Machine Classification algorithm through its SVC class. We have used the kernel to be of poly type with a degree 2 as we were classifying non-linearly separable data. This degree is used only when we are dealing with a poly kernel and the remaining kernels don't need it.
- Pandas: We have used pandas to the read the input data which is a csv file into a dataframe.
- Numpy: We have further converted the above dataframe into an array data structure by using numpy. We separated the input data into a two arrays, one having the set of 2D points as an array of an array and the other having the classification labels as a single array.

Implementation Comparison

- In the implementation without libraries, the input points and labels were read into a dataframe as 3 different columns and then separated into labels and data. This data was later used to find the maximum margin line by using matrix solvers from cvxopt.
- In the implementation with libraries, the input data was read using numpy and then separated and then fitted to a SVC class instance with a poly kernel with degree 2. We have then used this

kernel to directly find the intercept and weights of one of the margin lines along with the support vectors.

- We used a polynomial kernel with a degree of 2 in both of our implementations of the nonlinear data points.

Result Comparison

- The line equation with support vectors obtained by using the library function is as follows:

```
Intercept:
[-1.70566938]
Weights:
[-1.         -1.         -1.         -0.82999867 -1.         -1.
 -1.         -1.         -1.         -1.         -1.         -1.
 -1.         -1.          1.          1.          1.          1.
  1.          1.          1.          1.          1.          1.
  1.          0.82999867  1.          1.        ]
Support vectors:
[[ -8.47422847   5.15621613]
 [ -2.7471552   -8.47244873]
 [ -6.90647562   7.14833849]
 [ -7.47227246  -5.09869845]
 [ -9.53754332  -0.51895777]
 [ -9.46760885   2.36139525]
 [ -8.92844214  -1.80373857]
 [  6.3666283   -6.49712918]
 [  0.20162846  -8.81260121]
 [  6.99249663  -6.41143087]
 [ -6.80002274  -7.02384335]
 [  9.90143538  -0.31483149]
 [ -4.98349411   8.31816584]
 [  4.27289989   8.67079427]
 [ 10.24592717   7.95373492]
 [-15.64719728   3.32039056]
 [-11.64621294  -0.87217731]
 [ 12.74780931   0.19913032]
 [-10.02833317  11.09354511]
 [  3.28969027 -14.15854536]
 [  2.91722251 -12.27214032]
 [ -1.08933763 -14.10562483]
 [-10.260969     2.07391791]
 [  1.66404809  12.68562818]
 [  1.3393313  -10.29098822]
 [ 16.42108453   6.07221393]
 [ 11.75880948  -9.85890377]
 [  9.67917724   4.3759541 ]]
```

The results obtained are different from the result obtained via the implementation without libraries. This is because we calculated the maximum margin line in the without libraries implementation whereas we calculated only one of the margin lines and all its support vectors in the with libraries implementation.

Possible Improvements

- For other forms of input data other than our homework data, we could standardize the classification by scaling every dimension into +1,-1 to improve performance.
- By taking care of missing data and outliers, we might get a better SVM model with greater accuracy.

PART 3: APPLICATIONS

- Diabetes Detection: We can use support vector machine(SVM) techniques to detect whether a person is in a diabetes and pre-diabetes stage. We can use SVM models to select sets of variables to yield the best classification of individuals. The classification schemes could have different categories like diagnosed diabetes, undiagnosed diabetes, pre-diabetes and no-diabetes. SVMs can further be used for detecting persons with common diseases by using common variables.
- Cancer Genomics Studies: SVMs can be used as a powerful classification tool for cancer genomic classification or subtyping. With the high-throughput technologies giving rise to a large amount of genomic and epigenomic data , the classification feature of SVMs can be used for the discovery of new biomarkers, new drug targets and a better understanding of cancer driver genes. This is because od SVMs ability to recognise subtle patterns in in complex datasets.
- Face Recognition: Face recognition wich is a K class problem, where K is the number of known individuals can be converted to binary classification problem. This can be done by reformulating the problem into a problem which models dissimilarities between two facial images. By modifying the interpretation of the decision surface generated by SVM, we can generate a similarity metric between faces that is learned from examples of differences between faces.

Group Members and Contributions

- Our group members for this programming assignment were Sudeeptha Mouni Ganji (ID: 2942771049) and Vanessa Tan (ID: 4233243951).
- Vanessa and Sudeeptha worked together to research and understand the algorithms.
- After implementation, Sudeeptha and Vanessa worked together to write the report.

Programming:

Algorithm Implementation : Vanessa Tan, Sudeeptha Mouni Ganji

Library Implementation: Sudeeptha Mouni Ganji, Vanessa Tan

Report Writing:

Part 1:Implementation Without Libraries	Vanessa Tan Sudeeptha Mouni Ganji
Part 2: Software Familiarization	Vanessa Tan Sudeeptha Mouni Ganji
Part 3: Applications	Sudeeptha Mouni Ganji Vanessa Tan