24-787 Homework 6

Due date: 03/27/2021 @ 11:59 pm EST

Note: In case a problem requires programming, it should be programmed in `Python`. In Programming, you should use plain `Python` language, unless otherwise stated. For example, if the intention of a Problem is familiarity with numpy library, it will be clearly noted in that problem to use numpy. Please submit your homework through Gradescope.

Submissions: There are two steps to submitting your assignment on Gradescope:

1. **HW06_Writeup**: Submit a combined pdf file containing the answers to theoretical questions as well as the pdf form of the FILE.ipynb notebooks.

- To produce a pdf of your notebooks, you can first convert each of the .ipynb files to HTML.
- To do this, simply run: `ipython nbconvert -to html FILE.ipynb` for each of the notebooks, where FILE.ipynb is the notebook you want to convert. Then you can convert the HTML files to PDFs with your favorite web browser.
- If an assignment has theoretical and mathematical derivation, scan your handwritten solution and make a PDF file.
- Then concatenate them all together in your favorite PDF viewer/editor. The file name (FILE) for naming should be saved as HW-assignmentnumber-andrew-ID.pdf. For example for assignment 1, my FILE = HW-1-lkara.pdf
- Submit this final PDF on Gradescope, and **make sure to tag the questions correctly!**

2. **HW06_Code**: Submit a ZIP folder containing the FILE.ipynb notebooks for each of the programming questions. The ZIP folder containing your iPython notebook solutions should be named as HW-assignmentnumber-andrew-ID.zip

Q1: Support Vector Machine with `cvxopt` (40 pts)

In this problem, you'll practice to solve a classification problem using SVM. In class, we have seen how to formulate SVM as solving a constrained quadratic optimization problem. Now, you will implement SVM in the primal form. Conveniently, the `cvxopt` module in `Python` provides a solver for constrained quadratic optimization problem, which does essentially all of the work for you. This solver can solve arbitrary constrained quadratic optimization problems of the following form:

$$arg.min_z\{\frac{1}{2}z^TQz + p^Tz\}, \text{ s.t. } Gz \le h \tag{1}$$

(a) (15 pts) You are given a data file **clean_lin.txt**. The first two columns are coordinates of points, while the third column is label. Now let's implement the following quadratic optimization problem:

$$arg.min_{\mathrm{w},b}\{\frac{1}{2}||\mathrm{w}||^2\}, \text{ s.t. } y_i(\mathrm{w}^Tx^{(i)} + b) \ge 1, \ (i = 1, 2, 3, ...) \tag{2}$$

To do this, you have to find how to turn this constrained optimization problem into the standard form shown in (1). Things you should keep in mind: which variable(s) does z need to represent? What do you need to construct Q, p, G and h?
Hint: z should be 3×1. G should be n×3, where n is the number of training samples.
Train the linear SVM on the data using `cvxopt.solvers.qp(Q,p,G,h)`. Plot the decision boundary and margin boundaries.

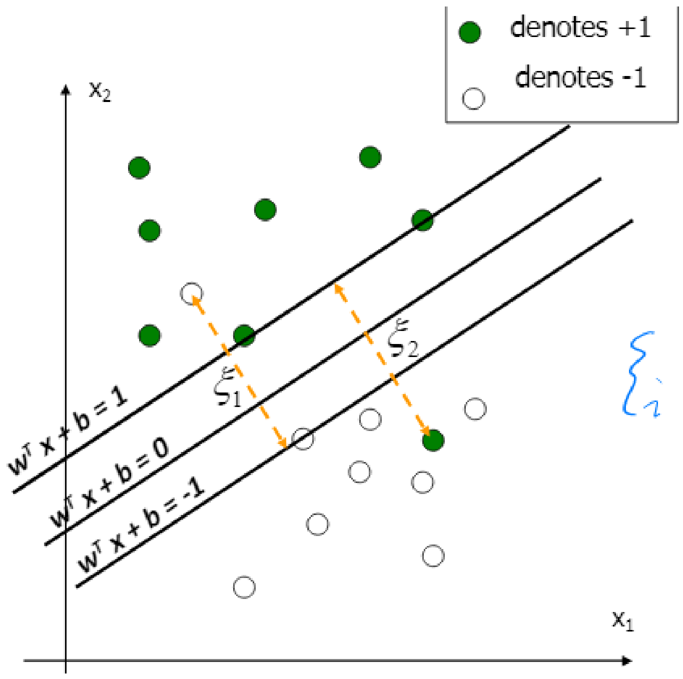

Figure 1: Hint: Use this snip from class slides to get expression of hyper plane and margins

Submit your code, computed final values of the Q, p, G and h vectors, and the plot of the decision boundary and margins. You should have a plot similar to Fig.2

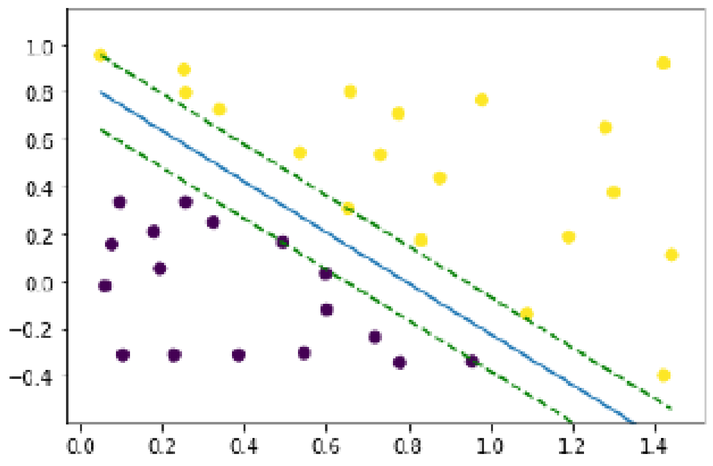

Figure 2: Expected decision boundary for the linearly separable dataset.

(b) (15 pts) Now let's go ahead to solve a linearly non-separable case using SVM. Load the training data `indirty_nonlin.txt`. As discussed in the lecture, introducing slack variables for each point to have a soft-margin SVM can be used for non-separable data. The soft-margin SVM optimization has following form:

$$arg.min_{\mathrm{w},b,\xi}\{\frac{1}{2}||\mathrm{w}||^2\} + C\sum_i \xi_i, \text{ s.t. } y_i(\mathrm{w}^T x^{(i)} + b) \geq 1 - \xi_i,\ \xi_i \geq 0\ (i = 1, 2, 3, ...) \tag{3}$$

At this point, use $C = 0.05$ in your code, but keep that as a variable because you will be asked to vary C in the subsequent questions. Again, you want to think about what the terms in the standard formulation represent now.
Hint: The problem is still quadratic in the design variables. So your solution, if found, will be the global minimum. $\mathbf{z}$ should be $(n+3) \times 1$. $\mathbf{G}$ should be $2n \times (n+3)$. If you construct your design vector as $[w_1; w_2; b; \xi_1, ..., \xi_n]$, you shall see that $\mathbf{G}$ can be constructed by putting together four sub matrices (upper left 3×3, lower right $n \times n$ etc.).
Finally, plot the decision boundary and margin boundaries. You should expect to have a plot similar to Fig. 3.
Submit your code, computed final values of the Q, p, G and h vectors, and the plot of the decision boundary and margins.

(c) (10 pts) Use your code in **(b)** to draw 4 plots, each corresponding to different values of C = [0.1; 1; 100; 1000000]. Discuss your observations of the decision margins.

Submit 4 plots using each C value and add a couple of lines of discussion.

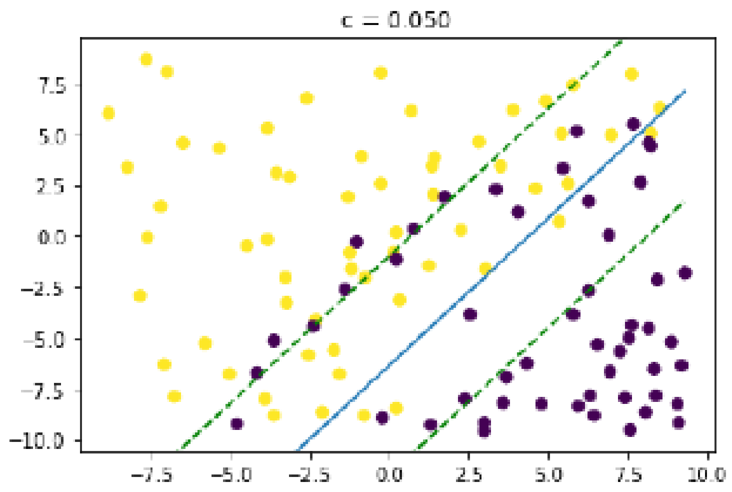

Figure 3: Decision boundary for the linearly non-separable dataset.

Q2: Support Vector Machine Experiments with `sklearn` (35 pts)

In this question, you will be conducting a series of SVM experiments with `sklearn` on multiple datasets with different settings. The focus is to explore the feasible parameter settings for different kinds of datasets. You will be provided with four datasets:

- Linear separable (`binary_linsep_yes_n240.txt`)
- Linear inseparable (`binary_linsep_no_n240.txt`)
- Moons (`binary_moons_linsep_no_n100.txt`)
- Circles (`concentriccircles_binary.txt`)

The SVM `sklearn` can be used by first import it at the beginning of your code with:

```
from sklearn.svm import SVC
```

You can find the document about the library here:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
(a) (25 pts) For each of the dataset, you are supposed to try the following parameters:

- **Kernels**: linear, poly, rbf
- **C**: which stands for the regularization parameters, the strength of the regularization is inversely proportional to C. Please try following values: {0.01,10,1000}

Thus, for each dataset, you will be having 9 models. For each model, you could specify the model parameters, fit the models and plot the model like the following sample code:

```
model = SVC(kernal='rbf',C=1)
model.fit(X,y) #X,y correpsonding to your loaded data
```

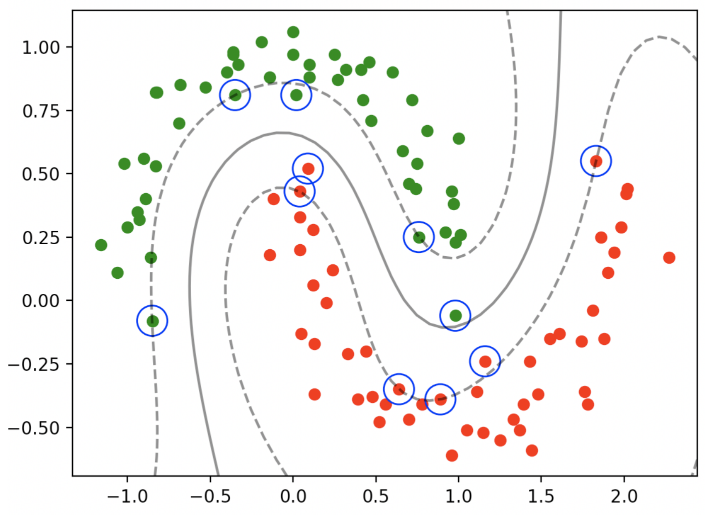

Figure 4: Sample plot of data and SVC classified boundary.

`plot_svc_decision_function(model)`
The plotting function `plot_svc_decision_function()` is supported by the backend code. In order to use it, you are supposed to put the file `plot_svc_decision_function.py` under the same path of your running code file and import it in your code by saying:
`from plot_svc_decision_function import plot_svc_decision_function`.
For each model, you would get an arch plot like Fig. 4. Since there are many repeating experiments, using subplots to include multiple figures into one plot is highly encouraged. Please make sure all the plots are shown on the submitted pdf file. Also, report the values of your model's support vector on the Moons dataset when C=1000 and the kernel is rbf, getting this with `model.support_vectors_`.

Submit your code, plots for all combinations of datasets and model parameters (altogether you would have 36 plots or subplots) and the support vector values of the above-mentioned case.

(b) (10 pts) Based on the above experiment results, summarize your observations on the parameter tuning of SVC on different parameters. For the Moons dataset, suppose you have two SVC models (a and b) with almost identical classification accuracy, while model a only have 5 support vectors and model b have 100 support vectors. Which model would you choose in terms of performance? Why?

Submit your observations and answers.

Q3: Support Vector Regression (25 pts)

This question uses **train.txt** and **test.txt**. You goal is to implement the Support Vector Regression on the given data using the built in libraries of `python-sklearn`. In this problem, you are advised to use the standardized features. All output plots must be in the original feature scales.
(a) (10 pts) Fit the data in *train.txt* for **C=1**, **epsilon=0.0001** and **kernel='rbf'** using the `sklearn.svm` module. Use the trained model to the test data in *train.txt*.

Report the RMSE value for your model.

(b) (10 pts) In order to study the influence of training parameters **C, epsilon**; report the RMSE values for all combination of C = 0.01, 10, 100 and epsilon = 0.0001, 0.01, 1.

Briefly explain the effect of each of the parameters on the model in you Jupyter notebook.

(c) (5 pts) Create a 100×100 uniformly spaced mesh grid ranging between the minimum and maximum of each of the features. Note that with this, you are generating new test (query) points.

Plot your model's response as a surface.

On the same plot, include the ground truth values as a point cloud directly obtained from the test file. The plot should be generated on the Jupyter Notebook. The plot must be such that the axes are composed of the original values of the two features, not the standardized values.
(Hint for plotting surface with `Python`, you might find the following documentation useful:
`https://matplotlib.org/3.1.1/gallery/mplot3d/surface3d.html`) Report the plots for the C and epsilon values mentioned in part(b).