CSE 326/426 (Fall 2022) Project 2

Due on 11:55 pm, Oct 24, 2022

Goal: Implement soft-SVM with the SMO algorithm for classification.

Instruction:

- Read the lecture note on SVM. You will need to use the SMO algorithm for soft-SVM. Pseudo-codes of SMO are provided as a skeleton of the SMO algorithm in the lecture note. Make sure you understand the math of SMO so that the details of the SMO can be put into codes.
- Work on the SVM-related questions in the HW to help you understand the math before implementation.
- Now start coding. Decompress the file project_2.zip and find the following files:

```
.
|-- data
‘-- src
    |-- problem1.py
    |-- problem2.py
    |-- problem3.py
    |-- problem4.py
    |-- svm_visualization.ipynb
    |-- test1.py
    |-- test2.py
    ‘-- test3.py
```

- Functions to be implemented are
  - problem1.py: linear and Gaussian kernel functions, the hinge loss function,
  - problem2.py: dual and primal objective functions, decision function of SVM using the dual variables,
  - problem3.py: training of SVM and test prediction accuracy. You will use the functions defined in problem 1 and 2 to simplify your codes.

  More detailed instructions are given in the comments of the functions.
- In files problem$*$.py ($* = 1, \ldots, 3$), there are functions for you to fill out with your codes.
- Files test$*$.py ($* = 1, \ldots, 3$) will unit-test the correctness of your implementations in the corresponding problem$*$.py files. For example, after you implement problem1.py file, run

  ```
  nosetests -v test1
  ```

  to test the correctness of your implementation of problem1.py. Note that passing the tests does not mean your implementations are entirely correct: the test can catch only a limited number of mistakes.

  If you use Anaconda (highly recommended), there should NOT be any packages you need to install. Otherwise, you will need to install nosestest for such unit test.

- We provide simple visualization of SVM models by plotting training data, the decision boundary, and the margins in the notebook:

  `src/svm_visualization.ipynb`

  You don't need to plot the training/test loss during training, as they are output to the terminal when we run problem4.py.

- There is no need to add/modify codes in svm_visualization.ipynb or problem4.py.

Grading: This project has 100 points in total. The number of points for each functions are printed when you run nosetests. 20 points go to the training quality, such as speed, accuracy, and completeness, when running problem4.py. The project counts towards the 40% of the final grade.

Submitting: There is no hand-written report required, and your submission should include the ONLY file

`<your_LIN>_P2.zip`

which, when decompressed, contains the same file trees shown above but with functions in problem∗.py implemented. Submit the zip file to Coursesite.

The example output figures within the plotting notebook should look similar to this:

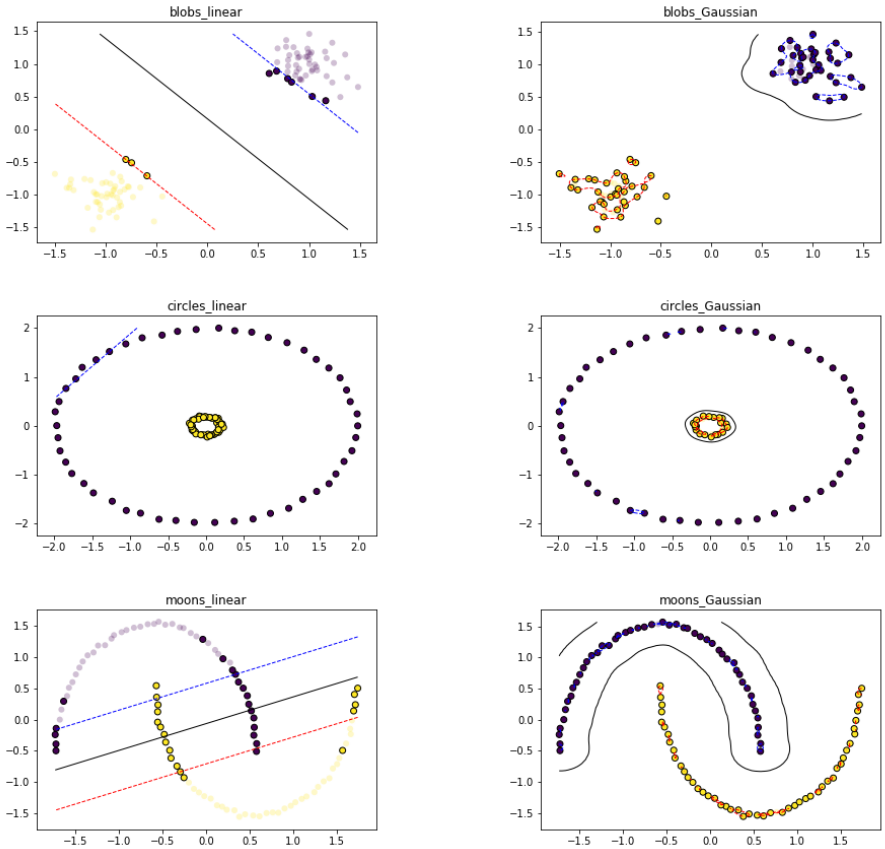

Figure 1:

Figure 2: Example plots of the SVM decision boundaries on three datasets with linear and Gaussian kernels.