CSE 326/426 (Spring 2019) Project 2

Due on 11:55pm, Apr 3, 2019

Goal: Implement SVM with the SMO algorithm for classification.

Instruction:

- First read the notes on SVM from Stanford Machine Learning, "The Simplified SMO Algorithm", which is self-contained if you're already familiar with soft-SVM in the dual form. Otherwise please go to PRML and the paper "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines" for more details. The pseudo codes in the Stanford note are not entirely correct (it has convergence issues), so read the Section 3.3 for the correct pseudo codes of the SMO algorithm in the technical report "Sequential Minimization Optimization for SVM".
- Then work on SVM-related questions in HW5 to understand the equations in the above notes/papers to help you implement/debug SMO.
- Now start coding. Decompress the file project_2.tgz and find the following file structure

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

- You will implement soft-SVM in the dual form with the SMO (Sequential Minimization Optimization) method. Functions to be implemented are linear and Gaussian kernel functions, the hinge loss function, dual and primal objective functions, decision function of SVM using the dual variables, training of SVM, and making binary prediction. More detailed instructions are given in the comments of the functions.
- Eight (8) files are provided: in files problem$*$.py ($* = 1, \ldots, 3$), there are functions for you to fill up 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.

  ```
  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 6% of the final grade (30% for all projects: 6% for projects 1-3, and 12% for the team project).

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

```
<your_LIN>_P1.tgz
```

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