CS 686-02 - Lab 2 Spring, 2018

Lab 2 - SVM Implementation

The goal of this assignment is to change an implementation of the SMO algorithm for Support Vectors to a class in our object-oriented framework. You will test this implementation by running it on some synthetic, linearly-separable data.

Background

Peter Harrington, author of Machine Learning in Action, has created an SMO implementation and made it available via github, here: https://github.com/pbharrin/machinelearninginaction3x/blob/master/Ch06/svmMLiA.py. This is a fantastic resource for us, but it can be criticised in at least two ways:

1. It does not fit into a larger framework, such as our classifier.py and the logistic_regression.py implementations
2. Harrington's implementation is conflated with specific data — i.e. it cannot be applied to any data

Your task is to correct these problems with Harrington's code and test it against some synthetic, linearly separable data

Requirements

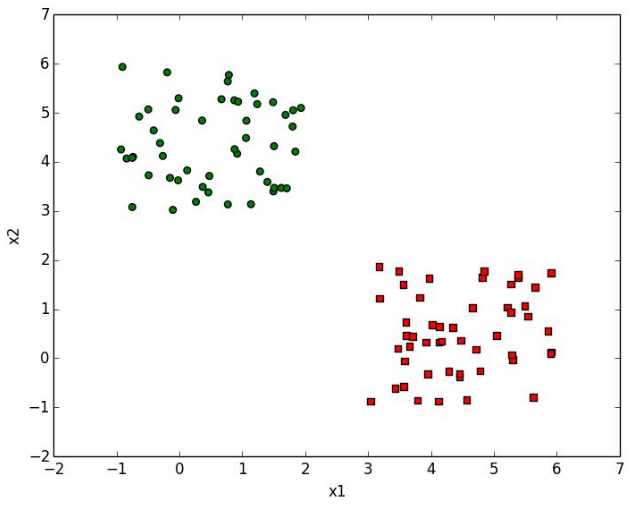

Figure 1: Plot of data in linearly_separable.csv

Perform the following steps:

1. Get data linearly_separable.py from https://github.com/dbrizan/cs686-2018-01/blob/master/linearly_separable.csv
2. Plot this data to ensure it is linearly separable. Your plot should look like Figure 1.
3. If you don't already have it, get classifier.py from the link above.
4. Create your own class svm_basic.py, which has classifier.py as its superlcass. Implement three functions in svm_basic.py:
    a. __init__(self) — i.e. the constructor
    b. fit(self, X, Y)
    c. predict(self, X)

    You may do this by changing Harrington's svmMLIA.py (Non-Kernel version) from the URL above.
5. Use a separate (non-OO) module to test your implementation against the data. A plot of this data should look like Figure 2.

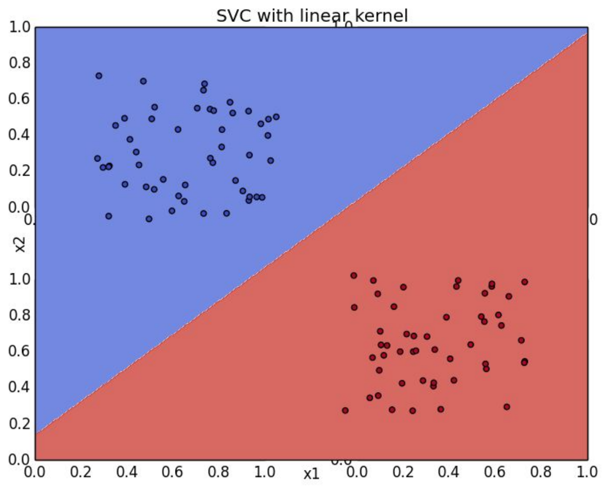

Figure 2: SVM class separation for linearly_separable.csv

Submission

Submit your source code or link to your github repository on Canvas.

Grading

Your grade for this assignment will be based equally on the logger implementations as follows:

- 100% = Implementation works correctly.
- 75% = Implementation works but contains minor errors (eg. the log level does not work correctly).
- 50% = Implementation works but contains major errors (eg. does not log correctly.

- 0% = Implementation not attempted or not submitted on time.