Assignment 4 Report

Xiaopei Zhang A20302816

I. Problem Statement

This assignment requires to implement various techniques in support vector machines (SVM) from scratch, evaluate the performance of different algorithms, and figure out how to predict class which each given test case belongs to. Line 209 to line 301 focus on linear SVM. It includes separable and non-separable datasets. Python skilearn linear svm and confusion matrix are used to compare and evaluate performance. Line 305 to line 458 is mainly about polynomial and Gaussian SVM. Confusion matrices are calculated, and Python sklearn polynomial and radical svm is used. The rest tests the unbalanced separable dataset with linear SVM and its corresponding Python linear svm.

II. Proposed Solution

I use the following functions to implement my SVM.

1. Solve dual to get Lagrange multipliers (C=10000 for hard margin).

$$L_D = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}X^{(i)T}X^{(j)} + \sum_{i=1}^{m}\alpha_i \quad \text{s.t.} \quad 0 \le \alpha_i \le C \quad \text{and} \quad \sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

2. Identify support vector set by $\alpha > 0$.

3. Compute w and w0.

$$w = \sum_{i=1}^{m}\alpha_i y^{(i)}X^{(i)} \qquad w_0 = \frac{1}{\#SV}\sum_{X^{(i)}\in SV}(y^{(i)} - w^T X^{(i)})$$

Classification is different between linear and nonlinear.

1. Linear: compute $w^T X + w_0$.

2. Nonlinear: compute $\sum_{X^{(i)}\in SV}\alpha_i y^{(i)}K(X^{(i)}, X) + w_0$.

If the result is larger than 0, x belongs to class 1;otherwise, x belongs to class -1.

Written up steps for solving primal objective function of soft margins SVM:

$$\frac{\partial L_P}{\partial w_i} = \frac{1}{2} \times 2 \times w_i - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0 \qquad w_i = \sum_{i=1}^{m} \alpha_i y^{(i)} X^{(i)}$$

$$\frac{\partial L_P}{\partial w_0} = -\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \qquad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = c - \alpha_i - \beta_i = 0 \qquad c - \alpha_i - \beta_i = 0$$

$$L_P = \frac{1}{2}(\sum_{i=1}^{m} \alpha_i y^{(i)} X^{(i)T})(\sum_{i=1}^{m} \alpha_i y^{(i)} X^{(i)}) + c\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i y^{(i)} (\sum_{j=1}^{m} \alpha_j y^{(i)} X^{(j)T}) X^{(i)} - \sum_{i=1}^{m} \alpha_i y^{(i)} w_0$$

$$+ \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \alpha_i \xi_i - \sum_{i=1}^{m} \beta_i \xi_i$$

$$= -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} X^{(i)T} X^{(j)} + \sum_{i=1}^{m} \alpha_i$$

III. Implementation

1. Program Design

My implementations of all the functions are highly modular. According to the sequence of my implementation, I explained most related functions below.

plot2D: take in data and scatter according to the first two features

gram: take in two datasets and return the dot product

polynomial: take in two datasets and return the polynomial with degree 3

gaussian: take in two datasets and return its gaussian value with sigma 0.5

getKernel: take X and kernel, and compute kernel values for each pair of values in X

my_svm: take in X, y, kernel and C, and compute weight, intercept, support vectors' alpha, x and y values

nonlinearDecisionBoundary: take in X, support vectors' alpha, x and y values and kernel, and return the sum of all the support vectors' kernel values multiplied by Lagrange and y

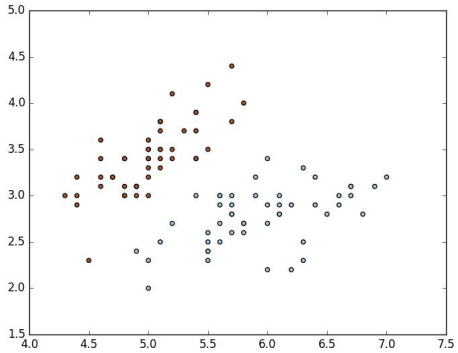
predict: take in X, weight, intercept, support vectors' alpha, x and y values, and kernel, return

the classes X belong to

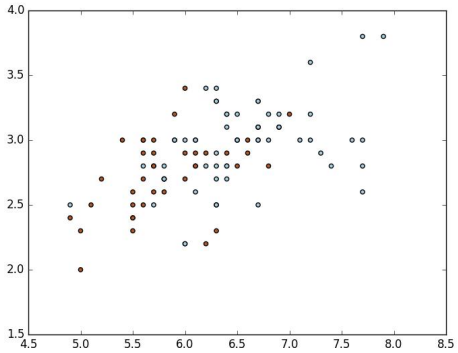
plotLinearSVM: it plots training data with decision line for linear model

plotNonlinearSVM: it plots training data with decision line for nonlinear model

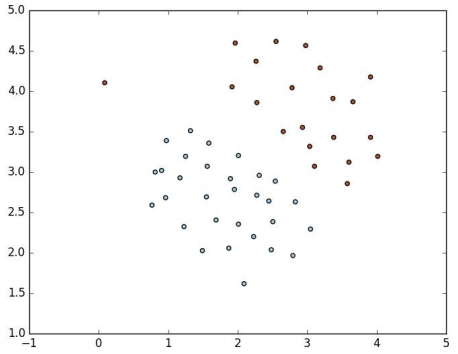
In the main function, "#" printout separates all the sections. In each section, I create global confusion matrices for results from my own functions and Python functions. Under 10 fold cross validation, I perform linear SVM (hard margin and soft margin), polynomial SVM and Gaussian SVM on iris_reorganized.data (the first two classes constitute the separable data, and the last two classes constitute the non-separable data), polynomial SVM and Gaussian SVM on ex6data1.mat and ex6.data3.mat (binary data from Andrew Ng's machine learning class).



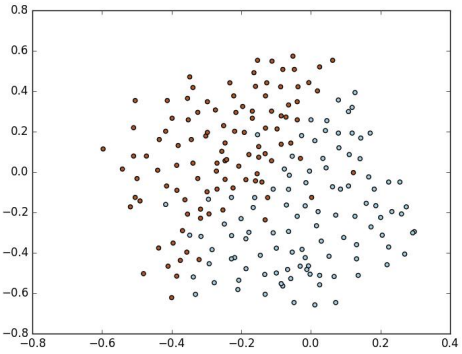
First 2 classes from iris (separable)



Last 2 classes from iris (non-separable)



ex6data1.mat



ex6data3.mat

2. Problems Encountered

The performance of my function is sometimes better and sometime worse than Python made function. I suspect that is has something to do with the initial guess. My linear SVM and Python linear SVM both cannot separate the non-separable data from iris. I guess that it is because the model is not suitable.

3. Instructions

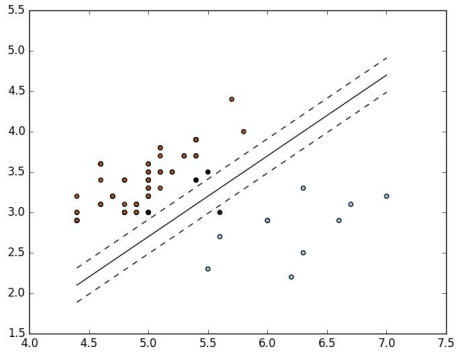
My script is written and tested under Python 2.7. My sklearn version is 0.17.1. Please keep my script and data under one directory before execution. Pictures will pop up during execution, and closing them would continue the program.

IV. Results and Discussion

1. Results

Linear SVM with hard margin for separable data

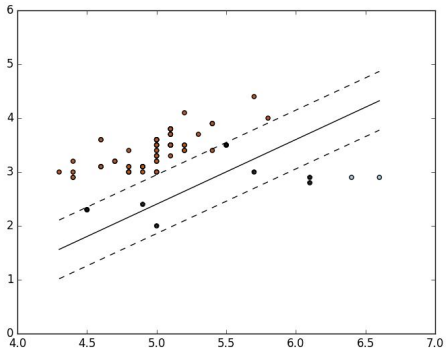
My linear SVM with hard margin:
[[15 0]
 [0 85]]
Python linear SVM with hard margin:
[[15 0]
 [0 85]]



Separable with hard margin

for non-separable data

My linear SVM with hard margin:
[[0 11]
 [0 89]]
Python linear SVM with hard margin:
[[0 11]
 [0 89]]



Separable with soft margin

Linear SVM with soft margin for separable data

My linear SVM with soft margin:

for non-separable data

My linear SVM with soft margin (non-separable):

[[4 0]
 [0 96]]
Python linear SVM with soft margin:
[[4 0]
 [0 96]]

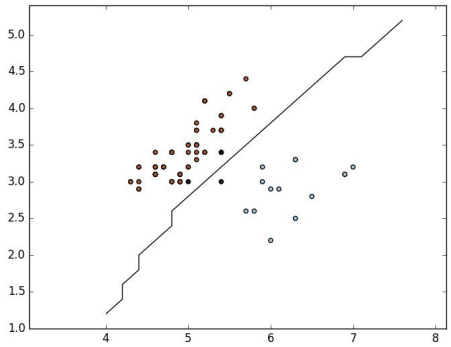
[[0 4]
 [0 96]]
Python linear SVM with soft margin (non-separable):
[[0 4]
 [0 96]]

Polynomial SVM for separable data

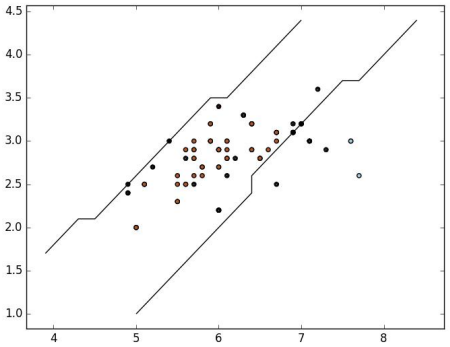
for non-separable data

My polynomial SVM (separable):
[[17 0]
 [0 83]]
Python polynomial SVM (separable):
[[17 0]
 [0 83]]

My polynomial SVM (non-separable):
[[2 15]
 [12 71]]
Python polynomial SVM (non-separable):
[[6 11]
 [0 83]]



Separable with polynomial



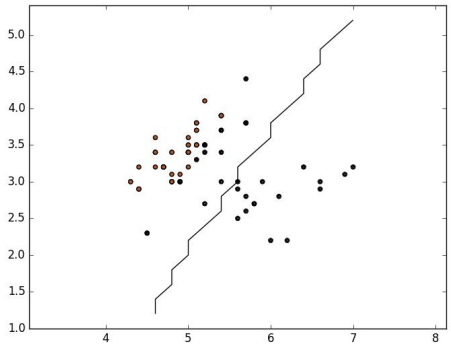
Non-separable with polynomial

Gaussian SVM for separable data

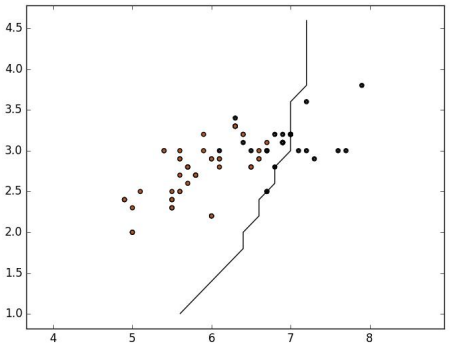
for non-separable data

My Gaussian SVM (separable):
[[13 5]
 [0 82]]
Python Radial SVM (separable):
[[16 2]
 [0 82]]

My Gaussian SVM (non-separable):
[[1 17]
 [0 82]]
Python radial SVM (non-separable):
[[1 17]
 [0 82]]



Separable with Gaussian

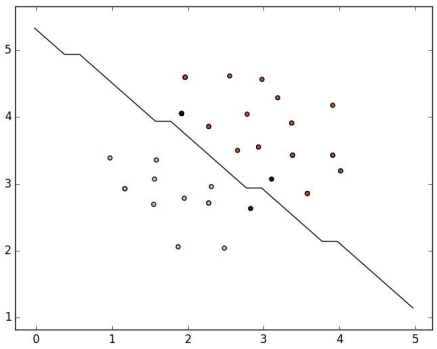


Non-separable with Gaussian

Polynomial SVM for ex6data1.mat

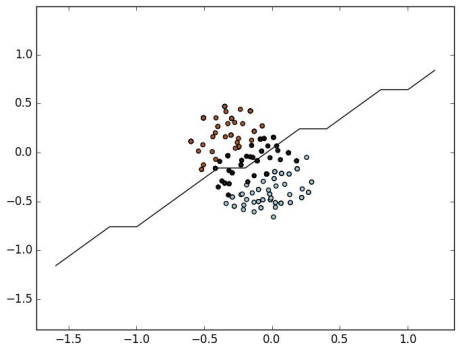
for ex6data3.mat

My polynomial SVM (ext):
[[14 0]
 [0 37]]
Python polynomial SVM (ext):
[[14 0]
 [0 37]]



Ex6data1 with polynomial

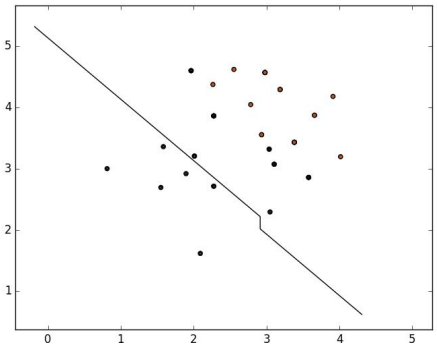
My polynomial SVM (ext):
[[95 11]
 [11 94]]
Python polynomial SVM (ext):
[[61 45]
 [64 41]]



Ex6data3 with polynomial

Gaussian SVM for ex6data1.mat

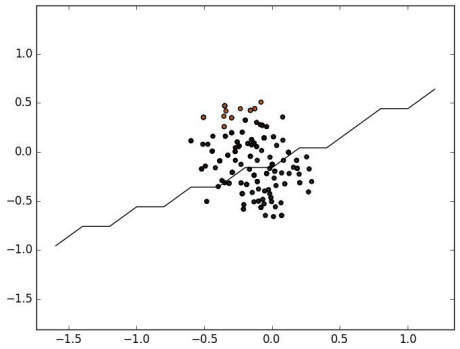
My Gaussian SVM (ext):
[[8 3]
 [0 40]]
Python Radial SVM (ext):
[[10 1]
 [0 40]]



Ex6data1 with Gaussian

for ex6data3.mat

My Gaussian SVM (ext):
[[78 15]
 [13 105]]
Python Radial SVM (ext):
[[79 14]
 [12 106]]



Ex6data3 with Gaussian

Linear SVM with soft margin for separable-unbalanced data

My linear SVM with soft margin (separable_unbalanced):
[[2 1]
 [0 62]]
Python linear SVM with soft margin (separable_unbalanced):
[[1 2]
 [0 62]]

2. Demonstration of Correctness

My implementations of SVM with linear, polynomial and Gaussian kernels are correct since my confusion matrices are similar with confusion matrices derived from Python made functions. Also, my implementations are based on the given functions in section II, so they are theoretically correct.

3. Performance Evaluation

(1) Strength. All of my implementations are theoretically derived, and my SVM has almost the same high accuracy as Python made function.

(2) Weakness. My implementation runs slower than Python made functions.

4. Effect of Various Parameters on Results

I change C and the number of examples to see how SVM performs. I found that when C gets larger than 1, the accuracy becomes lower when the two classes cannot be totally separated. Also, when the number of examples from two classes are unbalanced, the accuracy becomes lower as well.

References

[1]http://archive.ics.uci.edu/ml/datasets/Iris
[2]http://scikit-learn.org/stable/auto_examples/svm/plot_svm_margin.html
[3]https://gist.github.com/mblondel/586753
[4]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
[5]http://stackoverflow.com/questions/22294241/plotting-a-decision-boundary-separating-2-classes-using-matplotlibs-pyplot