ASSESSMENT COVER SHEET

Assignment Details

Course: Introduction to stastical Machine Learning

Semester/Academic Year: 2020 Sem 2

Assignment title: SVM

Assessment Criteria

Assessment Criteria are included in the Assignment Descriptions that are published on each course's web site.

Plagiarism and Collusion

Plagiarism: using another person's ideas, designs, words or works without appropriate acknowledgement.

Collusion: another person assisting in the production of an assessment submission without the express requirement, or consent or knowledge of the assessor.

Consequences of Plagiarism and Collusion

The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include: the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion and/or receiving a financial penalty.

DECLARATION

I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others. I have read the University Policy Statement on Plagiarism, Collusion and Related Forms of Cheating:

http://www.adelaide.edu.au/policies/?230

I give permission for my assessment work to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted and retained in a form suitable for electronic checking of plagiarism.

Vandit
2/9/20

SIGNATURE AND DATE

2020 Introduction to Introduction to Statistical Machine Learning Assignment 1: Support Vector Machines

Vandit Jyotindra Gajjar
School of Computer Science, The University of Adelaide
vanditjyotindra.gajjar@student.adelaide.edu.au
Student ID: a1779153

Abstract

Machine Learning is one of the applications of Artificial Intelligence which provides a system/network to gather knowledge from the environment and learn from experience without hard-coding. This assignment/report will mainly focus on the Support Vector Machines (SVMs) [1] which is a supervised learning model used for Classification and Regression analysis. Here, for the assignment we are focusing on linear classification also there is a 'Kernel Trick' used for non-linear data which is out of the scope. One of the parts of the assignment covers is the comparison of an existing implementation in scikit-learn [2] and by using python tool CVXOPT [3]. The experiment was conducted on the dataset given which consists of 8500 training samples and 1500 testing samples. This report focuses on solving the assignment questions of support vector machines with detailed analysis. The source code is publicly available at here.

Support Vector Machines

Support Vector Machines is a linear classifier where W is the weight matrix, x is the input points, and b is the bias

$$f(x) = W^T x + b$$

This above expression can be formulated to solve as an optimization problem over weight matrix W:

$$min||w||^2 + C\sum_{i}^{N} \max(0, 1 - y_i f(x_i))$$

This optimization problem is known as the primal problem. Instead, the Support Vector Machines can be formulated to learn the linear classifier by solving an optimization problem over α.

$$f(x) = \sum_{i}^{N} \alpha_i y_i (x_i^T x) + b$$

This is known as the **dual** problem.

Primal and Dual forms:

N is the number of training data points and d is the dimension of feature vector x.

Primal: For $w \in R^d$

$$\min ||w||^2 + C \sum_{i}^{N} \max(0, 1 - y_i f(x_i))$$

Dual: For $\alpha \in R^d$

$$\max \sum_i \propto_i - \frac{1}{2} \sum_{jk} \propto_j \propto_k y_j y_k (x_j T x_k) \text{ subject to } 0 \leq \alpha_i \leq C$$

Primal version of classifier:

$$f(x) = W^T x + b$$

Dual version of classifier:

$$f(x) = \sum_i^N \propto_i y_i (x_i T x) + b$$

Support Vectors:

Support vectors are the points nearest to the hyperplane which we can see in Figure 1. Now, if the data points are removed which would cause alteration in the position of this dividing hyperplane. Thus, they are considered as the critical elements of the data. A hyperplane is a line which linearly separates and classifies a set of data.

Max Margin:

The distance between the hyperplane and the nearest data point from either set is known as the margin. In Figure 1, from the support vectors which is shown inline and from the line to the dotted line, the distance is called margin. Thus if the distance is W from line to red plane and W from line to blue plane, this will lead to the max-margin to $\frac{2}{W}$. So basically searching for the max-margin is somewhat equal to searching for the nearest data points between the convex hulls.

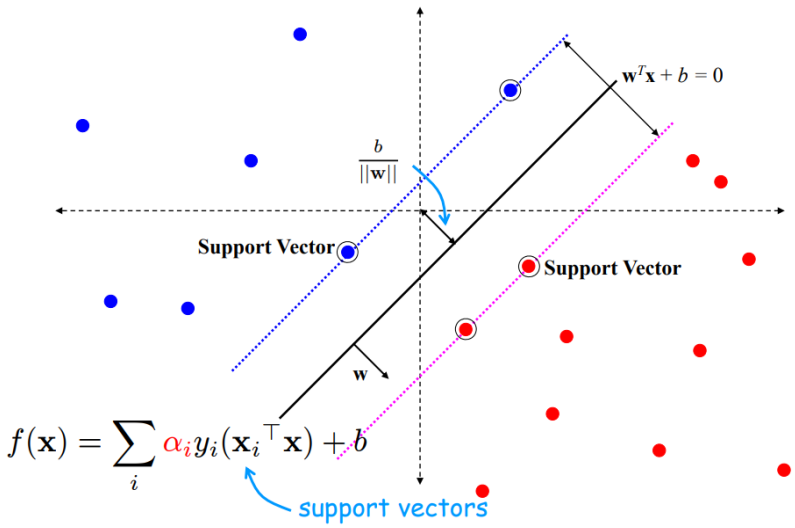

Figure 1. Support Vector Machines. Best viewed in color and magnification.

Soft Margin Classifier:

The constraint to maximize the margin of the line that differentiates the class must be relaxed and this is often called a soft margin. A tuning parameter called C which is the regularization parameter that defines the amount of violation of the margin allowed. During the learning of the hyperplane from data points, all the training data points lie within the distance of the margin and referred to as the support vectors. And as regularization parameter C affects the number of data points that are allowed to fall within the margin, so basically C influences the number of support vectors used by the model.

- The smaller the value of regularization parameter C, the higher the sensitivity of the algorithm to training set (high variance and low bias).
- The larger the value of regularization parameter C, the lower the sensitivity of the algorithm to training set(low variance and high bias).

Define Variables:

- **W:** Normal Vector to the hyperplane
- **x:** Input data points
- **b:** Bias
- **α:** Vector of the weights
- **d:** The dimension of feature vector x
- **C:** Regularization parameter

Implementation Details

In our experiment, we have performed the training on a workstation with the Intel i-7 core processor and accelerated by NVIDIA GTX 1060 6 GB GPU. All the experiments run in the Windows platform using Anaconda virtual environment. The training and testing data is 8500 and 1500 data points consisting of 200 features. For better experiments, we have further divided the training set to train and validation to see the algorithm performance before trying out the test set, thus we divided 7225 data points to train and the remaining data points to the validation set. For simplicity, we have converted the label 0 to -1, thus we get two classes -1 and 1, hence known as a binary classification.

Experiments

We perform experiments on the dataset and compare it with the existing application of the Scikit-learn library [2] and using the CVXOPT python tool-box [3]. First, we write the implementation for primal form and dual form, and for certain regularization parameter C, we compute the training accuracy and comparing with each other. In Table 1, we report the training accuracy comparison for different regularization parameter C. In Table 2, we report the testing accuracy comparison for fix number of regularization parameter C.

Table 1. Comparison of Training Accuracy for Primal, Dual form, and Scikit-learn library for different regularization parameter.

Regularization Parameter C	Training Accuracy Primal	Training Accuracy Dual	Training Accuracy SVC
1	0.9741	0.8494	0.9741
5	0.9749	0.8949	0.9749
10	0.9764	0.9113	0.9772
15	0.9756	0.9121	0.9756
20	0.9756	0.9160	0.9749
25	0.9741	0.9215	0.9741
30	0.9717	0.9254	0.9717
35	0.9717	0.9286	0.9725
40	0.9709	0.9301	0.9709
45	0.9709	0.9333	0.9701

50	0.9701	0.9341	0.9701
55	0.9709	0.9349	0.9709
60	0.9709	0.9356	0.9701
65	0.9694	0.9364	0.9701
70	0.9701	0.9364	0.9701
75	0.9701	0.9364	0.9701
80	0.9717	0.9356	0.9701
85	0.9709	0.9364	0.9709
90	0.9717	0.9372	0.9709
95	0.9709	0.9380	0.9709
100	0.9709	0.9380	0.9709

Table 2. Comparison of Testing Accuracy for Primal, Dual form, and Scikit-learn library for certain regularization parameter.

Regularization Parameter C	Testing Accuracy Primal	Testing Accuracy Dual	Testing Accuracy SVC
1	0.97	0.830	0.97
10	0.972	0.917	0.972
25	0.968	0.938	0.968
50	0.968	0.945	0.968
100	0.968	0.95	0.968

We also compare the weights and bias obtained from solving the primal problem, with the weight and bias reconstructed by the dual variables obtained via solving the dual problem. In Table 3, we compare the weights. When comparing these results we found that the bias and weight for primal results are significantly smaller than the dual. The reason behind this will be the calculation of optima using minima and maxima.

Table 3. Comparison of Bias and Weights for Primal and Dual problem.

	Primal	Dual
Bias	0.00011	2.00319
Max weight	1.1472	0.4228
Min weight	-0.3736	-0.3736
Mean weight	0.0046	-0.0005

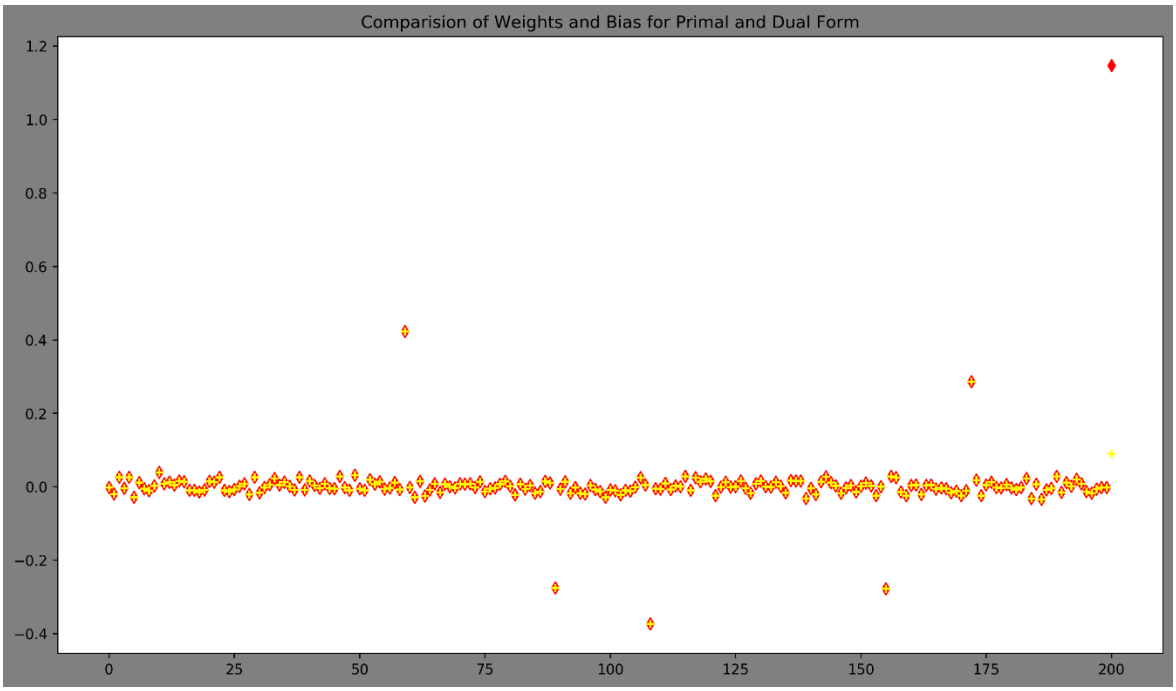

Figure 2. Comparison of weights and bias for primal and dual form. The + is for SVM Dual Form while the diamond is for SVM Primal Form. The graph has been plotted using Matplotlib library [4]. Best viewed in color and magnification.

We also performed one of the visualizations to check for outliers in weights and bias after performing the training. In Figure 2, the remaining data points which are not in the hyperplane are the outliers and which increases our error.

Finally, we calculated the confusion matrix in Table 4 and Table 5. This provides insights into how the test set is being categorized into its actual class and this also provides a brief insight that where the support vector machines go wrong in terms of classification.

Table 4. Confusion Matrix for Primal Form when regularization parameter C is set to 10.

Primal Form	Predicted: True	Predicted: False
Predicted: True	734	16
Predicted: False	11	724

Table 5. Confusion Matrix for Dual Form when regularization parameter C is set to 10.

Dual Form	Predicted: True	Predicted: False
Predicted: True	667	82
Predicted: False	43	708

References:

[1] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297. 1

[2] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830. 1, 4

[3] Diamond, Steven, and Stephen Boyd. "CVXPY: A Python-embedded modeling language for convex optimization." The Journal of Machine Learning Research 17.1 (2016): 2909-2913. 1, 4

[4] Hunter, John D. "Matplotlib: A 2D graphics environment." *Computing in science & engineering* 9.3 (2007): 90-95. 6