2nd Homework Assignment
Project on Support Vector Machines

Vasileios Papageorgiou

June 22, 2024

1 Introduction

The purpose of this project is to implement John Platt's Sequential Minimal Optimization Algorithm (SMO) to train a Support Vector Machine (SVM) for binary classification. Support Vector Machines are supervised learning models used for classification and regression. Their basic principle involves finding the hyperplane that best separates data points of different classes by maximizing the margin between them. This is achieved by solving an optimization problem to identify the support vectors, which are the data points closest to the hyperplane.
The fundamental principle of SMO is that it tries to solve the dual rather than the primal optimization problem. In this project, we will implement a basic version of the SMO algorithm to train an SVM binary classifier using the methodology from Platt's original paper [1]. We will explain each step of the process, providing the analytical background along with code snippets.

2 Problem Formulation

Given a dataset with m training examples and n features, where the i-th example is represented as $(x^{(i)}, y^{(i)})$ with $y^{(i)} \in \{-1, +1\}$ as the label, we aim to address the linear separation problem, potentially involving outliers. The primal problem involves finding a vector $\mathbf{w} = (w_1, w_2, \dots, w_n)$ and a scalar b under the following constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} s_i \\ \text{s.t.} \quad & y^{(i)}(\mathbf{w}^T x^{(i)} + b) \geq 1 - s_i, \quad i = 1, \dots, m \\ & s_i \geq 0, \quad i = 1, \dots, m \end{aligned}$$

Here, C is the regularization parameter that penalizes the outliers. We allow training samples to have a margin less than 1 and we pay the cost of Cs_i. The parameter controls the trade-off between maximizing the margin and minimizing the margin violations, balancing a wider margin with a smaller number of margin failures.
Using Lagrange duality we can derive the dual problem, corresponding to the primal one, which is formulated as follows:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \end{aligned}$$

We note that the slack variables s_i do not appear in the dual formulation. The optimization problem is now converted into a dual quadratic problem, where the objective function is solely dependent on a set of Lagrange multipliers α_i, under the box constraint $0 \leq \alpha_i \leq C$ and one linear equality constraint $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$.
The Karush-Kuhn-Tucker (KKT) conditions for the dual problem are for all i:

$$\alpha_i = 0 \iff y_i u_i \geq 1,$$

$$0 < \alpha_i < C \iff y_i u_i = 1,$$

$$\alpha_i = C \iff y_i u_i \leq 1.$$

where u_i is the output of the SVM for the i-th training example.
The Karush-Kuhn-Tucker (KKT) conditions reveal crucial insights for SVM training: Support vectors are points where $0 < \alpha_i < C$, situated on the margin's edge. These vectors play a critical role in defining the decision boundary. When $\alpha_i = 0$, the point lies outside the margin and has no impact on the prediction, serving as non-support. Points with $\alpha_i = C$ are inside the margin and may or may not be classified correctly, influencing the SVM's decision boundary.
There is a one-to-one relationship between each Lagrange multiplier α_i and each training example. Once the Lagrange multipliers are determined, the normal vector $\mathbf{w}$ can be derived from them:

$$\mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i \mathbf{x}_i$$

.

3 SMO Algorithm

(a) Algorithm Description

- Overview of the SMO algorithm.
- Key decisions in the implementation: variable selection, initial solution, termination criteria.

(b) Implementation Details

- Pseudocode of the implemented SMO algorithm.
- Discussion on hardwiring the constant C.

4 Experimental Setup

- Description of the dataset used (gisette dataset).
- Preprocessing steps and train-test split (using 4500-5000 points for training).

5 Results

- Presentation of the results obtained from the basic implementation.
- Evaluation metrics used for assessing the performance.

6 Optimization and Fine-Tuning

- Strategies for optimizing the choice of the constant C.
- Any other optimizations or improvements made to the algorithm.
- Comparative results before and after optimization.

7 Discussion

- Analysis of the results and their implications.
- Challenges faced during implementation and how they were addressed.
- Limitations of the current implementation and potential future work.

8 Conclusion

- Summary of the work done.
- Key findings and their significance.
- Final thoughts and potential directions for future research.

A Appendix

- Additional tables, figures, or code snippets.
- Detailed mathematical derivations if necessary.

Theoretical Background

We have the following non linear program:

$$\min\{F(x) = \frac{c^T x}{d^T x} : Ax = b;\ x \geq 0\} \tag{1}$$

Algorithm 1 Bisection Method for Optimal α

```
1: Given: interval $[L, U]$ that contains optimal $\alpha$
2: repeat
3:     $\alpha := \frac{u+l}{2}$
4:     Solve the feasibility problem:
5:         $c^T x \leq \alpha d^T x$
6:         $d^T x > 0$
7:         $Ax = b$
8:     Adjust the bounds
9:     if feasible then
10:        $U := \alpha$
11:    else
12:        $L := \alpha$
13:    end if
14: until $U - L \leq \epsilon$
```

```
    # Define parameter s
    s = y1 * y2

    # Compute L, H via equations (13) and (14) from Platt
    if y1 != y2:
    L = max(0, a2 - a1)
    H = min(self.C, self.C + a2 - a1)
    else:
    L = max(0, a2 + a1 - self.C)
    H = min(self.C, a2 + a1)

    if L == H:
    return 0
```

Problem 4

(a) Updating the Error Cache

When a Lagrange multiplier is non-bound after being optimized, its cached error is zero. The stored errors of other non-bound multipliers not involved in joint optimization are updated as follows.

$$E_k^{\text{new}} = E_k^{\text{old}} + u_k^{\text{new}} - u_k^{\text{old}} \tag{3.36}$$

$$E_k^{\text{new}} = E_k^{\text{old}} + u_k^{\text{new}} - u_k^{\text{old}} \tag{3.37}$$

For any k-th example in the training set, the difference between its new SVM output value and its old SVM output value, $u_k^{\text{new}} - u_k^{\text{old}}$, is due to the change in α_1, α_2 and the change in the threshold b.

$$u_k^{\text{new}} - u_k^{\text{old}} = y_1 \alpha_1^{\text{new}} k_{1k} + y_2 \alpha_2^{\text{new}} k_{2k} - b^{\text{new}} - \left(y_1 \alpha_1^{\text{old}} k_{1k} + y_2 \alpha_2^{\text{old}} k_{2k} - b^{\text{old}}\right) \tag{3.38}$$

Substituting equation (3.37) into equation (3.36), we have

$$E_k^{\text{new}} = E_k^{\text{old}} + y_1 \left(\alpha_1^{\text{new}} - \alpha_1^{\text{old}}\right) k_{1k} + y_2 \left(\alpha_2^{\text{new}} - \alpha_2^{\text{old}}\right) k_{2k} - \left(b^{\text{new}} - b^{\text{old}}\right) \tag{3.39}$$

References

[1] John Platt. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical Report MSR-TR-98-14, Microsoft, April 1998. https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/.

[2] Ginny Mak. The Implementation of Support Vector Machines Using the Sequential Minimal Optimization Algorithm. Master's thesis, McGill University, School of Computer Science, Montreal, Canada, April 2000.