MACHINE LEARNING ALGORITHMS: LAB EXERCISES

Lanshi FU, Zhe HUANG, Linghao ZENG
Faculté des Sciences d'Orsay
Université Paris-Saclay

ABSTRACT

The aim of this lab exercise is to code three different algorithms to train a binary SVM classifier. The SVM classifier is used to separate data points into two classes, and the algorithms are designed to optimize the hyperplane that separates the data points with the highest margin. The three algorithms are sub-gradient descent on the primal, projected gradient ascent on the dual, and box constrained coordinate ascent on the dual.

***K*eywords** SVM · dual SVM · sub-gradient descent · projected gradient ascent · box constrained coordinate ascent

1 Primal Problem of SVM

1.1 Problem Definition

Support Vector Machines (SVMs) are a type of supervised learning algorithm used for classification and regression analysis. The primary goal of SVMs is to find a hyperplane that separates the data points into different classes, with the largest possible margin between the hyperplane and the closest data points.

The objective function for the primal training problem of SVM is defined as follows:

$$\min_a P(a) = \sum_i \max\left(0, 1 - X^{(i)} a y^{(i)}\right) + \frac{c}{2}\|a\|_2^2 \tag{1}$$

where X is a matrix of sample points of shape (200, 3), y is the ground truth of shape (200, 1), a is the parameters of model of shape (3, 1), c is the regularizaton weight. For X and a, the last column is the bias term.

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & 1 \\ \vdots & \vdots & \vdots \\ x_1^{(200)} & x_2^{(200)} & 1 \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(200)} \end{bmatrix} \qquad a = [a_1 \quad a_2 \quad b] \tag{2}$$

1.2 Sub-gradient

1.2.1 Explanation

We cannot directly compute the gradient of the objective function. Specifically, the hinge loss term $\max(0, 1-X^{(i)}ay^{(i)})$ is not differentiable at $X^{(i)}a = y^{(i)}$. Because the objective function is not differentiable at these points, we cannot use traditional gradient-based optimization algorithms to minimize it. However, we can still use sub-gradient descent to find a solution that minimizes the objective function.

1.2.2 Details

For the objective function of an SVM, we can compute a sub-gradient for each data point that has a non-zero hinge loss term. The sub-gradient of the objective function is:

$$\begin{aligned}
\nabla_a P(a) &= \nabla_a(\sum_i 1 - X^{(i)}ay^{(i)} + \frac{c}{2}\|a\|_2^2) \\
&= \sum_i \nabla_a(1 - X^{(i)}ay^{(i)}) + ca \\
&= \begin{bmatrix} \nabla_{a_1}(1 - (X_1^{(i)}a_1 + X_2^{(i)}a_2 + b)y_i) \\ \nabla_{a_2}(1 - (X_1^{(i)}a_1 + X_2^{(i)}a_2 + b)y_i) \\ \nabla_b(1 - (X_1^{(i)}a_1 + X_2^{(i)}a_2 + b)y_i) \end{bmatrix} + ca \\
&= \begin{bmatrix} \sum_i -X_1^{(i)}y_i \\ \sum_i -X_2^{(i)}y_i \\ \sum_i -y_i \end{bmatrix} + ca \\
&= -X^T@y + ca
\end{aligned} \tag{3}$$

where $1 - X^{(i)}ay^{(i)} > 0$, @ is the matrix product implemented in numpy. When $1 - X^{(i)}ay^{(i)} < 0$, the sub-gradient is ca. Therefore, the sub-gradient function is

$$\nabla_a P(a) = \begin{cases} -X^T@y + ca & \text{if } 1 - X^{(i)}ay^{(i)} > 0 \\ ca & \text{if } 1 - X^{(i)}ay^{(i)} \leq 0 \end{cases} \tag{4}$$

Then the sub-gradient step is

$$a^{(t+1)} = a^{(t)} - \epsilon\nabla_a P(a^{(t)}) \tag{5}$$

where ϵ is the learning rate.

2 Dual Problem of SVM

2.1 Problem Definition

Here, we change notation a little. Let $Y = \begin{bmatrix} y^{(1)} & \dots & \dots & 0 \\ 0 & y^{(2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & y^{(200)} \end{bmatrix}$ be a diagonal matrix containing the labels. Let $\mathcal{L}(w) = \max(0, 1 - w)$ and $h(w) = \frac{1}{2}\|w\|_2^2$. Then, we can rewrite the primal problem as:

$$\min_a \sum_i \mathcal{L}([YXa]_i) + ch(a) \tag{6}$$

In addition, we know that

- $f(w) = \max(0, 1 - w)$, then the conjugate function of f is $f^*(u) = u + \delta_{[-1,0]}(u)$, $[\sum f(w)]^* = \sum f^*(w)$
- $h(a) = \frac{1}{2}\|a\|_2^2$, then the conjugate function of h is $h^*(a) = \frac{1}{2}\|a\|_2^2$, $[ch(a)]^* = ch^*(a)$
- the dual problem of $\min_v F(Ua) + H(a)$ is $\max_\lambda -F^*(\lambda) - H^*(-U^T\lambda)$m , where $U = YX$
  Proof. Let $F : \mathbb{R}^m \to \mathbb{R}, H : \mathbb{R}^n \to \mathbb{R}, U \in \mathbb{R}^{m\times n}$, and assume the following problem:

  $$\min_{a\in\mathbb{R}^n} F(Ua) + H(a)$$

  Let $t = Ua$, we write the problem as:

  $$\min_{a\in\mathbb{R}^n, t\in\mathbb{R}^m} F(t) + H(a) \qquad \text{PRIMAL}$$

  We relax the constraint using dual vars $\lambda \in \mathbb{R}^m$ and build the Lagrangian dual:

  $$\geq \max_{\lambda\in\mathbb{R}^m} \min_{a\in\mathbb{R}^n, t\in\mathbb{R}^m} F(t) + H(a) + \lambda^\top(Ua - t) \qquad \text{DUAL}$$

$$
\begin{aligned}
&= \max_{\lambda \in \mathbb{R}^m} \left(\min_{T \in \mathbb{R}^m} F(t) - \lambda^\top t \right) + \left(\min_{a \in \mathbb{R}^n} H(a) + \lambda^\top U a \right) \\
&= \max_{\lambda \in \mathbb{R}^m} - \left(\max_{t \in \mathbb{R}^m} \lambda^\top t - F(t) \right) - \left(\max_{a \in \mathbb{R}^n} -\lambda^\top U a - H(a) \right) \\
&= \max_{\lambda \in \mathbb{R}^m} -F^*(\lambda) - H^*(-\lambda^T U)
\end{aligned}
$$

End.

Let $F(w) = \sum f(w)$, $H(a) = ch(a)$. Thus, we have the dual problem of SVM as:

$$
\begin{aligned}
&\max_{\lambda} - \sum_i \lambda_i - \frac{c}{2} \frac{\lambda^T Y X}{c} \frac{X^T Y \lambda}{c} \\
&\text{s.t. } -1 \leq \lambda_i \leq 0, \forall 1 \leq i \leq n
\end{aligned} \tag{7}
$$

We can recover optimal primal variables from optimal dual variables. From the KKT conditions, for optimal primal and dual variables we have

$$
\begin{aligned}
\nabla_a (F(t) + H(a) + \lambda^\top (Ua - t)) &= 0 \\
\nabla_a \left(\sum_{i=1}^n \max(0, 1 - t) + \frac{c}{2} \|a\|_2^2 + \lambda^\top (YXa - t) \right) &= 0 \\
ca + \left(\lambda^\top Y X \right)^\top &= 0 \\
a &= -\frac{1}{c} X^\top Y \lambda
\end{aligned} \tag{8}
$$

2.2 Projected Gradient Ascent

2.2.1 Explanation

In the dual formulation of the SVM optimization problem, we seek to maximize the Lagrangian function with respect to the Lagrange multipliers, subject to certain constraints (here $[-1, 0]$). One way to solve the dual problem is to use projected gradient ascent, which is an iterative optimization algorithm that updates the Lagrange multipliers in the direction of the gradient of the Lagrangian function and then projects the result onto the feasible region of the optimization problem.

2.2.2 Details

For the SVM dual problem, we have objective function eq. (7). Then we compute the gradient

$$
\begin{aligned}
\nabla f \left(\lambda^{(t)} \right) &= -1 - \frac{1}{2c} \left(Y^\top X X^\top Y + \left(Y^\top X X^\top Y \right)^\top \right) \lambda^{(t)} \\
&= -1 - \frac{1}{c} Y^\top X X^\top Y \lambda^{(t)}
\end{aligned} \tag{9}
$$

The project gradient ascent step is:

$$
\lambda^{(t+1)} = \text{proj}_{[-1,0]^n} \left[\lambda^{(t)} + \epsilon \nabla f \left(\lambda^{(t)} \right) \right] \tag{10}
$$

where proj is the projection operator, ϵ is the learning rate. Projection into the convex set $[-1, 0]^n$ id to clip each coordinate to $[-1, 0]$, i.e.:

$$
\text{Clip}_{[0,1]}(w) = \begin{cases} -1 & \text{if } w \leq -1 \\ 0 & \text{if } w \geq 0 \\ w & \text{otherwise} \end{cases}
$$

2.3 Box Constrained Coordinate Ascent

2.3.1 Explanation

Box Constrained Coordinate Ascent updates the Lagrange multipliers one at a time, in any order, and only updates the Lagrange multiplier for the current coordinate while fixing the other Lagrange multipliers.

2.3.2 Details

Here, we do a notation change for eq. (7). Let $Q = -YXX^TY$, $b = -1$, $l = -1$, $u = 0$, then we have

$$\begin{aligned} &\max_{\lambda} \frac{1}{2}\lambda^\top \boldsymbol{Q}\lambda + \boldsymbol{b}^\top \lambda \\ &\text{s.t. } l \leq \lambda \leq u \end{aligned} \tag{11}$$

Then we can rewrite the objective function as

$$\begin{aligned} f(\lambda) &= \frac{1}{2c}\lambda^\top \boldsymbol{Q}\lambda + \boldsymbol{b}^\top \lambda \\ &= \frac{1}{2c}\sum_{i=1}^{n}\lambda_i \sum_{j=1}^{n}\lambda_j Q_{j,i} + \sum_{i=1}^{n} b_i\lambda_i \\ &= \frac{1}{2c}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j Q_{j,i} + \sum_{i=1}^{n} b_i\lambda_i \\ &= \frac{1}{2c}\sum_{i\neq j}\lambda_i\lambda_j Q_{j,i} + \frac{1}{2c}\sum_{i}\lambda_i^2 Q_{i,i} + \sum_{i=1}^{n} b_i\lambda_i \end{aligned} \tag{12}$$

Solve the partial derivative

$$\frac{\partial}{\partial\lambda_k} f(\lambda) = \frac{1}{c}\sum_{i\neq k}\lambda_i Q_{k,i} + \frac{1}{c}\lambda_k Q_{k,k} + b_k = 0 \tag{13}$$

then we have

$$\lambda_k = \frac{-cb_k - \sum_{i\neq k}\lambda_i Q_{k,i}}{Q_{k,k}},\ \text{s. t. } l \leq \lambda_k \leq u \tag{14}$$

therefore

$$\lambda_k = \text{Clip}_{[l,u]}\left[\frac{-cb_k - \sum_{i\neq k}\lambda_i Q_{k,i}}{Q_{k,k}}\right] \tag{15}$$

where $l = -1$, $u = 0$, $b = -1$.

3 Experiments

3.1 Experimental Setup

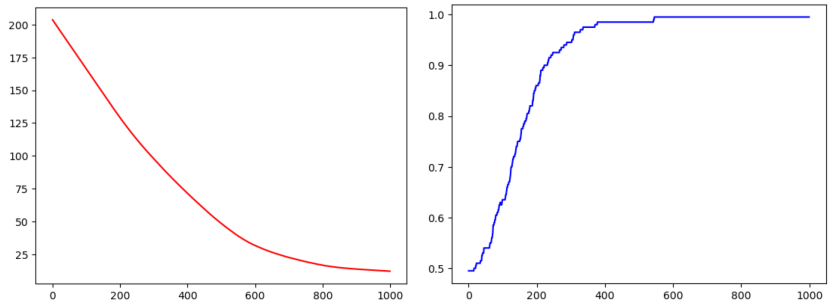
For this experiment, we used the NumPy library in Python to implement the three different algorithms for training a binary SVM classifier: sub-gradient descent on the primal, projected gradient ascent on the dual, and box constrained coordinate ascent on the dual. We also used the scikit-learn library to generate a toy dataset to test the performance of the algorithms.

The toy dataset was generated using scikit-learn's make_classification function, which creates a random multivariate dataset with a specified number of samples, features, and classes. We set the number of samples to 200, the number of features to 2, and the number of classes to 2, to create a binary classification problem. We also set the class separation to 3 to make the dataset well separated.

3.2 Results

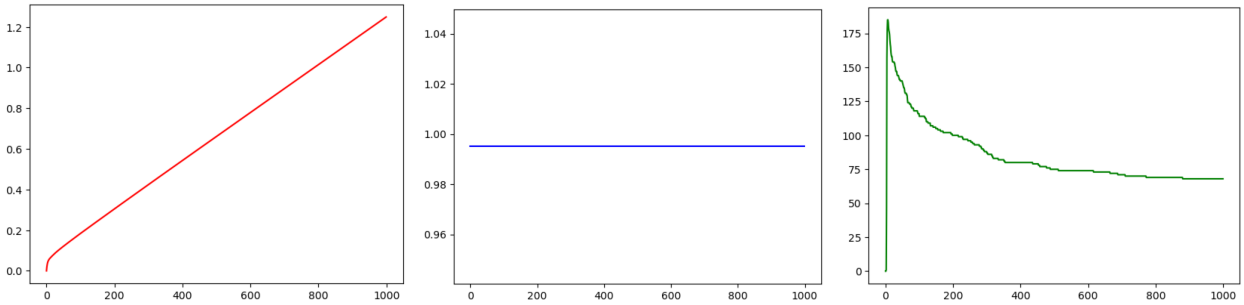
3.2.1 Hyperparameters

The learning curves of our three algorithms are illustrated in Fig. 1, 2 and 3. The regularization weight is set to 1 in these algorithms. The step size of sub-gradient ascent is set to 1e-6, and the step size of projected gradient ascent is set to 1e-4. We found that the optimal value of the regularization parameter c varied depending on the algorithm used, with larger values of c leading to better performance for the projected gradient ascent and box constrained coordinate ascent algorithms.



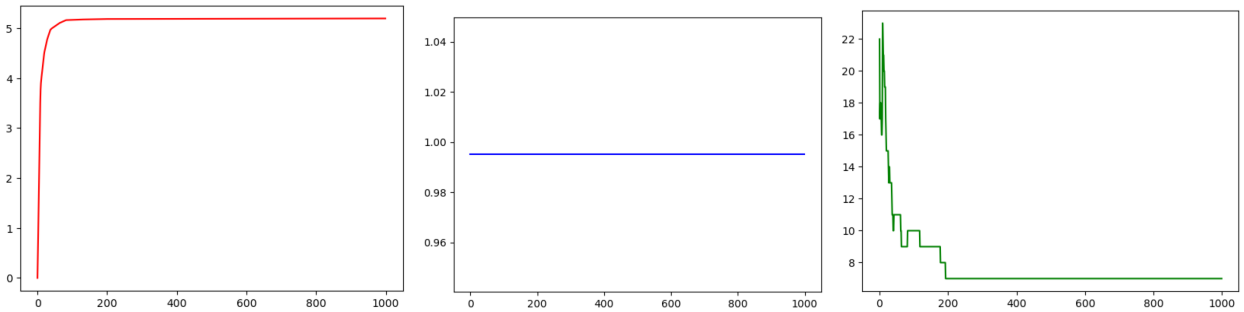
(a) Objective function (b) Classification accuracy

Figure 1: Learning curve of sub-gradient descent on the primal problem.



(a) Objective function (b) Classification accuracy (c) Support vectors

Figure 2: Learning curve of projected gradient ascent on the dual problem.



(a) Objective function (b) Classification accuracy (c) Support vectors

Figure 3: Learning curve of box constrained coordinate ascent on the dual problem.

3.2.2 Analysis and Comparison

Because we only use a toy dataset for experiments, the final accuracy after 1000 epochs are all nearly 100%, which are 0.995, 0.995, 0.995, respectively.

For the first algorithm, the output of objective function is decreased over epochs smoothly. For the latter two algorithms, the output of objective function is increased over epochs. The difference between projected gradient ascent and box constrained coordinate ascent is that coordinate ascent does not need a step size, so the output of its objective function increases more quickly.

The curve of first algorithm's classification accuracy increases over epochs, while the other two don't and have high accuracy at the beginning. After code review, we believe that it's because the parameters of the first algorithm are initialized randomly, but the parameters of other two are all initialized to 0, which may be very close to the ideal best value.

Support vectors are the data points that lie on the margin or are misclassified by the hyperplane. For the two gradient ascent algorithms, we can see that number of support vectors decrease over epochs. As step size is not needed in box constrained coordinate ascent, its number of support vectors decrease more quickly than projected gradient ascent. After 1000 epochs, the number of support vectors of projected gradient ascent is 68, while the number of support vectors of box constrained coordinate ascent is 7, which indicates that the performance of box constrained coordinate ascent is better in this situation.

In a sentence, the sub-gradient descent algorithm was the slowest to converge, while the projected gradient ascent and box constrained coordinate ascent algorithms were faster and more stable

4 Conclusion

In this exercise, we have conducted an experiment to compare the performance of three different algorithms for training a binary SVM classifier on a toy dataset. The three algorithms we used were sub-gradient descent on the primal, projected gradient ascent on the dual, and box constrained coordinate ascent on the dual. Overall, our experiment provides insights into the strengths and weaknesses of different algorithms for training a binary SVM classifier, and can help guide the choice of algorithm and parameter settings. We believe that the methods we used in this experiment can be applied to other machine learning problems and provide valuable insights into the performance of different algorithms.