Dual Descent Method

Sai Pravallika Danda
Department of Computer Science and Engineering
IIT HYDERABAD
cs20btech110013@iith.ac.in

Badavath Revanth Naik
Department of Computer Science and Engineering
IIT HYDERABAD
cs20btech11007@iith.ac.in

***Abstract*—Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression tasks. The main goal of it is to find the best separating hyperplane between the classes with maximum margin. The optimization problem is to maximize the margin with the constraint that no points lie in between the margin which simplifies to a quadratic programming problem. Lagrange multipliers is one of the strategies to solve optimization problems. Lagrange problem is typically solved in the dual form. One of the main challenges is the large computational cost of QPP. The long training time also prevents it from locating the optimal parameter set from a very fine grid of parameters over a large span. The Dual Coordinate Descent (DCD) is one of the efficient algorithms that directly solves the QPP by orderly optimizing the series of single-variable sub-problems. In this paper, we wish to explore the algorithm and it's efficiency.**

***Index Terms*—SVM, Primal, Dual, DCD - Dual Coordinate Descent, QPP, Lagrangian**

I. Introduction

Support Vector Machines are supervised learning algorithms for classification of the data. SVM though is a linear model, it can be extended to solve non-linear classification problems. In this paper we wish to explore dual coordinate descent, an optimisation algorithm, to solve linear SVM classification problem. We are applying coordinate descent technique to the dual optimisation problem of the Linear SVM, where we update one variable at a time by minimising a single variable sub-problem. In general implementations, accessing values per-instance is much easier than accessing data values of corresponding features. Hence, choosing the dual problem while implementing coordinate descent method is better option compared to the primal problem.

II. Optimisation Problem of Linear SVM

Formulation of linear SVM for classification is as follows: Consider a training set of data instances and labels $(x_i, y_i), i = 1, 2, ...l$, where $y_i \in \{-1, +1\}$ and $x_i \in \mathbb{R}^n$. And the classification function is $sgn(w^T x + b)$, where (w, b) are the weights and bias respectively.

A. Primal Optimisation Problem

$$\begin{aligned} \min_{\omega,b} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\xi_i \\ \text{subject to } y_i(\omega^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0, i = 1, 2, ...l \end{aligned} \tag{1}$$

where n is the number of features, $C \geq 0$ is the penalty parameter, **b** is the bias term and ξ is the loss function.

B. Dual Optimisation Problem

$$\begin{aligned} \min_{\alpha} f(\alpha) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{subject to } 0 \leq \alpha_i \leq C, i = 1, 2, ...l \\ \mathbf{y}^T\alpha = 0 \text{ (if bias is considered)} \end{aligned} \tag{2}$$

where $\boldsymbol{\alpha}$ are the Lagrangian multipliers, $\mathbf{e} = [1, ..., 1]^T$ and $Q_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ If the bias term is removed, then the dual problem does not have a linear constraint.
From the primal-dual relationship, we have $\boldsymbol{\omega} = \sum_{i=1}^{l} y_i \alpha_i \boldsymbol{x_i}$

III. Dual Coordinate Descent

In coordinate descent method, each coordinate of the variable vector is updated successively, one at a time, unlike Gradient Descent method where all the coordinates are updated at once. In the above stated linear SVM dual optimization problem, we are going to update each coordinate of $\boldsymbol{\alpha}$ (α_i). After one cycle of updating of coordinates is done, the process of updating repeats.

A. Outer Iteration:

Let us consider $\boldsymbol{\alpha^0} \in \mathbb{R}^l$ as the initial vector of Lagragians. In the optimisation process, sequence of vectors $\{\boldsymbol{\alpha^k}\}_{k=0}^{\infty}$ are generated. The process of generating $\boldsymbol{\alpha^{k+1}}$ from $\boldsymbol{\alpha^k}$ is called an outer iteration.

B. Inner Iteration:

Each outer iteration consists of l inner iterations, where we update each coordinate of $\boldsymbol{\alpha^k} - [\alpha_1^k, \alpha_2^k ..., \alpha_l^k]$ sequentially. After l inner iterations, we get $\boldsymbol{\alpha^{k+1}}$

$$\boldsymbol{\alpha^{k,i}} = [\alpha_1^{k+1}, ...\alpha_{i-1}^{k+1}, \alpha_i^k .....\alpha_l^k]^T \quad \forall i = 2, .., l \tag{3}$$

$\boldsymbol{\alpha^{k,i}}$ represents the vector generated at the end of k outer iterations followed by i inner iterations.

IV. Procedure

A. One-variable sub-problem

Since we update one coordinate at a time in an iteration, the above dual optimization problem (2) now reduces to a one-variable sub-problem in a variable d.
On updating $\boldsymbol{\alpha^{k,i}}$ to $\boldsymbol{\alpha^{k,i+1}}$, the i^{th} coordinate is updated

while keeping others the same. The optimisation problem we now have is,

$$\min_{d} \quad f(\boldsymbol{\alpha^{k,i}} + d\boldsymbol{e}_i)$$
$$\text{subject to } 0 \leq \alpha_i^k + d \leq C \tag{4}$$

B. Objective Function

The objective function for the above optimisation problem (4) is a quadratic function in variable d. It is given as,

$$f(\boldsymbol{\alpha^{k,i}} + d\boldsymbol{e}_i) = f(\boldsymbol{\alpha^{k,i}}) + \nabla_i f(\boldsymbol{\alpha^{k,i}})d + \frac{1}{2}Q_{ii}d^2 \tag{5}$$

where $\nabla_i f$ is the i^{th} component of ∇f

C. Projection Gradient

$\nabla^P f(\boldsymbol{\alpha})$ is the projected gradient defined as below,

$$\nabla_i^P f(\boldsymbol{\alpha}) = \begin{cases} \nabla_i f(\boldsymbol{\alpha}) & 0 < \alpha_i < C \\ \min(0, \nabla_i f(\boldsymbol{\alpha})) & \alpha_i = 0 \\ \max(0, \nabla_i f(\boldsymbol{\alpha})) & \alpha_i = C \end{cases} \tag{6}$$

If projected gradient at $\boldsymbol{\alpha}^{k,i}$ becomes 0, the updating of α_i is not necessary as the optimal α_i is reached, and move to the next index $i + 1$.

D. Solving the sub-problem

Without any constraints, optimal d for the above objective function

$$d = -\frac{\nabla_i f(\boldsymbol{\alpha^{k,i}})}{Q_{ii}} \tag{7}$$

From (4) subjecting d to the constraints, the obtained solution is

$$\alpha_i^{k,i+1} = \begin{cases} \alpha_i^{k,i} - \frac{\nabla_i f(\boldsymbol{\alpha^{k,i}})}{Q_{ii}} & 0 \leq \alpha_i^k + d \leq C \\ C & \alpha_i^k + d \geq C \\ 0 & \alpha_i^k + d \leq 0 \end{cases} \tag{8}$$

which can also be written as,

$$\alpha_i^{k,i+1} = \min(\max(\alpha_i^{k,i} - \frac{\nabla_i f(\boldsymbol{\alpha^{k,i}})}{Q_{ii}}, 0), C) \tag{9}$$

E. Finding the Gradients (∇f)

The i^{th} component of gradient of objective function can be obtained as,

$$\begin{aligned} \nabla_i f(\boldsymbol{\alpha}) &= \sum_{j=1}^{l} Q_{ij}\alpha_j - 1 \\ &= \sum_{j=1}^{l} y_i y_j \mathbf{x}_i^T \mathbf{x}_j \alpha_j - 1 \end{aligned} \tag{10}$$

The operations involved in the above expression to calculate gradient directly are computationally expensive, costs $O(ln)$. However, since for linear SVM we have

$$\boldsymbol{\omega} = \sum_{j=1}^{l} y_j \alpha_j \boldsymbol{x_j} \tag{11}$$

Gradient now can be calculated in $O(n)$ time as follows

$$\nabla_i f(\boldsymbol{\alpha}) = y_i \boldsymbol{\omega}^T \boldsymbol{x}_i - 1 \tag{12}$$

F. Updating the weights

From the primal-dual relationship, we have $\boldsymbol{\omega} = \sum_{i=1}^{l} y_i \alpha_i \boldsymbol{x_i}$. Calculating ω using it is computationally expensive, takes $O(ln)$ time. An alternate way for updating ω after each inner iteration, taking $O(n)$ time is as follows,

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + (\alpha_i^{k+1} - \alpha_i^k) y_i \boldsymbol{x}_i \tag{13}$$

At the end of all the outer iterations, we obtain the optimal value of ω of the primal problem (1)

V. Algorithm

Algorithm : Dual coordinate descent for linear SVM

- Given $\boldsymbol{\alpha} = \boldsymbol{\alpha^0}$ and initial weights $\boldsymbol{\omega} = \sum_{i=1}^{l} y_i \alpha_i^0 \boldsymbol{x_i}$
- While $\boldsymbol{\alpha}$ is not optimal (Outer Iteration)

  For i = 1, 2, ... l (Inner Iteration)

  (a) $\alpha_{temp} = \alpha_i$

  (b) $\nabla = y_i \boldsymbol{\omega}^T \boldsymbol{x_i} - 1$

  (c) $\nabla^P = \begin{cases} \min(\nabla, 0) & \alpha_i = 0 \\ \max(\nabla, 0) & \alpha_i = C \\ \nabla & 0 < \alpha_i < C \end{cases}$

  (d) If $\nabla^P \neq 0$

  $\alpha_i \leftarrow \min(\max(\alpha_i - \frac{\nabla}{Q_{ii}}, 0), C)$

  $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + (\alpha_i - \alpha_{temp}) y_i \boldsymbol{x}_i$

VI. Discussions

A. Bias Term

In our report, in the formulation of linear SVM optimisation problem, the bias term was not considered. With the inclusion of bias term, a linear constraint $\boldsymbol{y^T \alpha} = 0$ is introduced in the dual optimisation problem. It will then be carried to the sub-problem discussed in the Procedure (IV). According to the study in [2], Dual coordinate descent for linear SVM without the bias term converges faster when compared with that with a bias term. The stated reason, a linear constraint may make the sub-problem to be easily optimal already; αs are not moved/updated. It concludes that for better results with CD, dual SVM without linear constraint has to be used.

VII. Comparisions

We studied the experiments done in [1] to analyse the performance of DCD. The following observations were made. SVM^{perf} and Pegasos are competitive with DCD but however they have slower conversions. The learning rate of SGD used in them might be the reason for their decent performance.
PCD too is a coordinate descent method. Unlike DCD it is applied to the primal and thus does not have a closed form solution for the sub-problems. This results in a higher cost per iteration in PCD.
TRON is a newton method, has fast final convergence but does not perform well in early iterations. So it does not generate a good model quickly.

VIII. Experiment Results

We used LinearSVM from sklearn library in python over the dataset 'The Cleveland Heart Disease'. The class LinearSVM uses DCD when the parameter 'dual' is set to True. We experimented with fitting SVM model under different conditions: less and more data points, using dual or primal, exclusion of bias and randomising the order of sub-problems when solving dual problem. The time of computation for each of them is tabulated below.

Sample Size	Method	Time(s)
n = 30	Dual - DCD	0.00961
	Dual - Rand	0.00832
	Dual - Bias	0.00848
	Primal	0.01073
n = 300	DCD	0.1136
	Primal	0.0251

When fitting the dataset over a smaller number of samples, we observed that the time taken for the dual problem is lesser when compared to the primal problem. Further, we experimented with exclusion of bias term and random shuffling of sub-problems and they were computationally efficient. However, this wasn't the same when a larger sample size was used. We also observed that accuracy was the same for both dual and primal problems.

IX. Conclusion

We observed that DCD is significantly faster than state-of-the-art solvers like Pegasos, PCD, TRON, SVM^{perf}. For linear SVM, DCD is efficient when used without a bias term, thus avoiding linear constraint. For one-class SVMs and SVDD, which have an inherent linear constraint, novel coordinate descent methods have to be developed (discussed in [3]). Further studies shows that, the training speed increases by randomising the order of sub-problems in each iteration. Also, as the instances and number of features in each instance increase, the DCD method trains well over other solvers.

References

[1] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, A dual coordinate descent method for large-scale linear SVM, in ICML, 2008.

[2] C.-C. Chiu, P.-Y. Lin, and C.-J. Lin, Twovariable block dual coordinate descent methods for large-scale linear support vector machines, in SDM, 2020.

[3] Hung-Yi Chou, Pin-Yen Lin and Chih-Jen Lin, Dual Coordinate-Descent Methods for Linear One-Class SVM and SVDD

[4] https://aclanthology.org/Q13-1017.pdf

[5] https://en.wikipedia.org/wiki/Support-vector_machine

X. Github Repository

https://github.com/spdanda/EE5606