# Dual Coordinate Descent-Group 31

AKYAM DHATRI NANDA - AI20BTECH11002
VOJESWITHA GOPIREDDY -AI20BTECH11024
VAISHNAVI W - AI20BTECH11025
MUCHARLA SUPRIYA - EE19BTECH11022
AKUNURI VISHWA VEENA - ES19BTECH11026

April 30, 2022

- Abstract
- Introduction
- Linear SVM Optimization Problem
  - Primal Optimization Problem
  - Dual Formulation
  - Dual Optimization Problem
- Dual Coordinate Descent
  - Outer Iteration
  - Inner Iteration
- Procedure
- Comparisons
- References

# Abstract

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression tasks.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 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. We wish to explore the efficiency of the algorithm.

# 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.

# Linear SVM Optimisation Problem

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.

# Linear SVM Optimisation Problem

## Primal Optimisation Problem

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

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

# Dual Formulation

For the optimisation problem, the Lagrangian is given by:

$$L(\omega, b, \xi, \alpha, \beta) = \frac{1}{2}\omega^T\omega + c\sum_{i=1}^{l}\xi_i + \sum_{i=1}^{l}\beta_i(-\xi_i) + \sum_{i=1}^{l}\alpha_i[(1-\xi_i) - y_i(\omega^T x_i + b)] \quad (2)$$

The dual form of the problem is obtained by minimising Lagrangian with respect to $\omega$, b and $\xi$. Each lagrange multiplier i.e., $\alpha_i$, $\beta_i$ is constrained to be greater than or equal to zero.

$$\max_{\alpha,\beta} f_d(\alpha, \beta) = \max_{\alpha,\beta} \min_{\omega,b,\xi} L(\omega, b, \xi, \alpha, \beta) \quad (3)$$

# Dual Formulation

Setting derivative of Lagrangian wrt $\omega$ to zero we get,

$$\frac{\partial L}{\partial \omega} = \omega - \sum_{i=1}^{l} \alpha_i y_i x_i \tag{4}$$

$$\implies \omega = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{5}$$

Setting derivative of Lagrangian wrt b to zero we get,

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} \alpha_i y_i \tag{6}$$

$$\implies \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{7}$$

# Dual Formulation

Setting derivative of Lagrangian wrt $\xi_i$ to zero we get,

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i \tag{8}$$

$$\implies C = \alpha_i + \beta_i \tag{9}$$

Since, we have $\beta_i$ = C - $\alpha_i$, the constraint $\beta_i \geq 0$ becomes $\alpha_i \leq C$.
Therefore the constraints of the dual problem are

$$0 \leq \alpha_i \leq C \tag{10}$$

# Dual Formulation

On substituting (6), (8), (10) in (2), the lagrangian simplifies to

$$f_d(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \boldsymbol{x_i}^T \boldsymbol{x_j} \tag{11}$$

$$= -(\frac{1}{2} \alpha^T Q \alpha - e^T \alpha) \tag{12}$$

# Linear SVM Optimisation Problem

## Dual Optimisation Problem

$$\min_{\alpha} f(\alpha) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$
$$\text{subject to } 0 \leq \alpha_i \leq C, i = 1, 2, ....l$$
$$\mathbf{y}^T\alpha = 0 \text{ (if bias is considered)} \quad (13)$$

where $\mathbf{e} = [1, ....., 1]^T$, $Q_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ and $\alpha$ is the Lagrangian multiplier.

# 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 $\alpha$ ($\alpha_i$). Once all the coordinates are updated, the process of updating repeats.

## Outer Iteration

Let us consider $\boldsymbol{\alpha^0} \in \mathbb{R}^l$ as the initial vector of Lagrangians. 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.

## 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 $\boldsymbol{\alpha^{k+1}}$

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

$$\boldsymbol{\alpha^{k,i}} = [\alpha_1^{k+1}, .....\alpha_{i-1}^{k+1}, \alpha_i^k .....\alpha_l^k]^T \quad \forall i = 2, .., l \tag{14}$$

# Procedure

## One variable sub-problem

Since we update one coordinate at a time in an iteration, the above dual optimization problem now reduces to a one-variable sub-problem in a variable $d$.

On updating $\boldsymbol{\alpha}^{\boldsymbol{k,i}}$ to $\boldsymbol{\alpha}^{\boldsymbol{k,i+1}}$, the $i^{th}$ coordinate is updated while keeping others the same. The optimisation problem we now have is,

$$\begin{aligned} \min_{d} \quad & f(\boldsymbol{\alpha}^{\boldsymbol{k,i}} + d\boldsymbol{e}_i) \\ \text{subject to } & 0 \leq \alpha_i^k + d \leq C \end{aligned} \tag{15}$$

# Procedure

## Objective Function

The objective function for the above optimisation problem (15) is a quadratic function in variable d. It is given as,

$$f(\boldsymbol{\alpha}^{\boldsymbol{k,i}} + d\boldsymbol{e}_i) = f(\boldsymbol{\alpha}^{\boldsymbol{k,i}}) + \nabla_i f(\boldsymbol{\alpha}^{\boldsymbol{k,i}})d + \frac{1}{2}Q_{ii}d^2 \tag{16}$$

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

# Procedure

## Projection Gradient

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

$$\nabla_i^P f(\boldsymbol{\alpha}) = \begin{cases} \nabla_i f(\boldsymbol{\alpha}) & 0 < \alpha_i < C \\ \min(0, \nabla_i f(\boldsymbol{\alpha})) & \alpha_i = 0 \\ \max(0, \nabla_i f(\boldsymbol{\alpha})) & \alpha_i = C \end{cases} \tag{17}$$

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

# Procedure

## Solving the sub-problem

Without any constraints, optimal d for the above objective function

$$d = -\frac{\nabla_i f(\boldsymbol{\alpha}^{\boldsymbol{k,i}})}{Q_{ii}} \tag{18}$$

From (15) subjecting d to the constraints, the obtained solution is

$$\alpha_i^{k,i+1} = \begin{cases} \alpha_i^{k,i} - \frac{\nabla_i f(\boldsymbol{\alpha}^{\boldsymbol{k,i}})}{Q_{ii}} & 0 \leq \alpha_i^k + d \leq C \\ C & \alpha_i^k + d \geq C \\ 0 & \alpha_i^k + d \leq 0 \end{cases} \tag{19}$$

which can also be written as,

$$\alpha_i^{k,i+1} = \min(\max(\alpha_i^{k,i} - \frac{\nabla_i f(\boldsymbol{\alpha}^{\boldsymbol{k,i}})}{Q_{ii}}, 0), C) \tag{20}$$

# Procedure

## Finding the Gradients ($\nabla f$)

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

$$\begin{aligned}\nabla_i f(\boldsymbol{\alpha}) &= \sum_{j=1}^{l} Q_{ij}\alpha_j - 1 \\ &= \sum_{j=1}^{l} y_i y_j \mathbf{x}_i^T \mathbf{x}_j \alpha_j - 1 \end{aligned} \tag{21}$$

The operations involved in the above expression to calculate gradient directly are computationally expensive, costs $O(ln)$.

# Procedure

## Finding the Gradients ($\nabla f$)

However, since for linear SVM with soft margin we have

$$\boldsymbol{\omega} = \sum_{j=1}^{l} y_j \alpha_j \boldsymbol{x_j} \tag{22}$$

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

$$\nabla_i f(\boldsymbol{\alpha}) = y_i \boldsymbol{\omega}^T \boldsymbol{x}_i - 1 \tag{23}$$

# Procedure

## Updating the weights

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

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + (\alpha_i^{k+1} - \alpha_i^k) y_i \boldsymbol{x}_i \tag{24}$$

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

# Algorithm

- 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 $\nabla^P \neq 0$

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

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

# Discussions

## 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 is introduced in the dual optimisation problem i.e.,

$$\boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{\alpha} = 0 \tag{25}$$

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; $\alpha$s are not moved/updated.

# Results

We used LinearSVM from sklearn library in python over the dataset 'The Cleveland Heart Disease'. The class LinearSVM uses DCD when 'dual' is set to True. 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.

# Results contd.

| 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 |

# Conclusions

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. 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.