SVM Optimisation using Primal-Dual Interior Points Method and SMO

Erna Ceka
Department of Computer Science, UCL
Word Count: 2779

May 18, 2021

1 Introduction

Support vector machines (SVMs) are a well established and rigorously funded technique for solution of classification problems in machine learning. In this report, we define the optimisation problem of the SVM binary classification model.

We solve the obtained optimisation problem using two appropriate methods: i) Primal-Dual Interior Points Method and ii) Sequential Minimal Optimisation Method. We discuss the theoretical and empirical convergence results and compare the performance of these two methods both in terms of computational efficiency and classification accuracy.

2 SVM Optimisation Problem

2.1 Defining the SVM Optimisation Problem

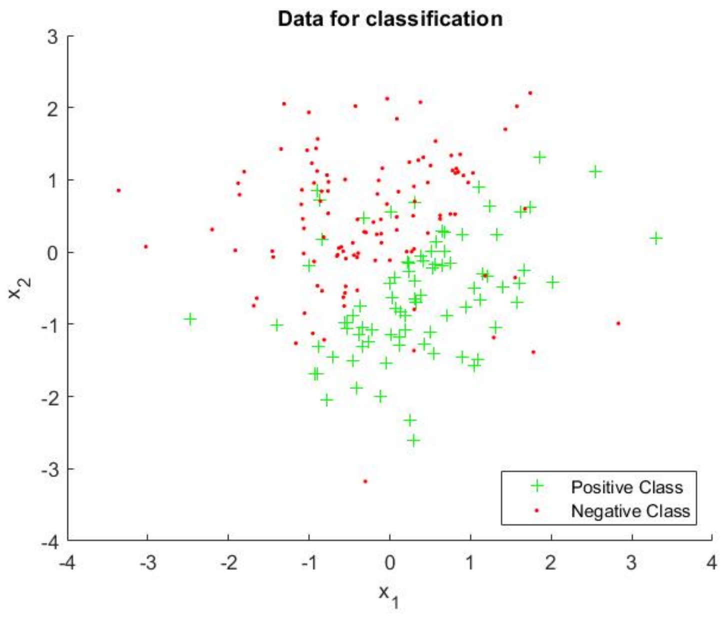
In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin. In the linear case, the margin is defined by the distance of the hyperplane to the nearest of the positive and negative examples. The formula for the output of a linear SVM is $f(x) = \beta^T x + \beta_0$, where β is the normal vector to the hyperplane and x is the input vector.

Given a training set $S = \{x_i; y_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^p$ and $y_i \in \{1, -1\}$, the aim of SVM is to induce a classifier which has good classification performance on future unseen examples.

We want to build an SVM model applied to a synthetic data to classify observations in two categories: the positive class $+1$, and the negative class -1, based on two features $x^{(1)}$ and $x^{(2)}$.. Our data consist of 200 observations in the training set and 100 observations in the test set.

A visual summary of our data is provided in the figure below.

Figure 1: Data for SVM classification



We first define a hyperplane as follows: $\{x : f(x) = x^T\beta + \beta_0 = 0\}$. A classification rule induced by $f(x)$ is $G(x) = sign\left[x^T\beta + \beta_0\right]$. As mentioned above, the idea of the SVM algorithm in binary classification problems relies on the principle of **maximal margin classifier**. Our aim is to find the hyperplane that creates the biggest margin between the training points for class 1 and -1. [1]. Maximising the margin is equivalent to solving the following optimisation problem:

$$\begin{array}{ll} \min\limits_{\beta_0,\beta} & \frac{1}{2}\|\beta\|^2 \\ \text{s.t.} & y_i\left(x_i^T\beta + \beta_0\right) \geq 1 \forall i \end{array} \tag{1}$$

where the constraints represent the linearly separable property. The large margin principle is reflected by minimizing $\frac{1}{2}\|\beta\|^2$ with $\frac{2}{\|\beta\|}$ being the margin between two lines $\beta^T \cdot x + \beta_0 = 1$ and $\beta^T \cdot x + \beta_0 = -1$. [2]
However, in most practical cases the classes are not completely linearly separable, instead they overlap in the feature space.This is the case for our data presented in Figure 1.
When the data is not completely linearly separable but we would still like to learn a linear classifier, a loss on the violation of the linearly separable constraints has to be introduced.

$$y_i\left(x_i^T\beta + \beta_0\right) \geq M(1-\xi_i) \tag{2}$$

In this formulation, the value ξ_i in the constraint $y_i\left(x_i^T\beta + \beta_0\right) \geq M(1-\xi_i)$, is the proportional amount by which the prediction $f(x_i) = x_i^T\beta + \beta_0 = 0$ is on the wrong side of the margin. Misclassifications occur when $\xi_i > 1$ so by bounding the sum $\sum_i \xi_i$ at a value C, we bound the total amount of training misclassifications at C. [3]

It is computationally convenient to now write the optimisation problem in the following form:

$$\begin{aligned} \min_{\beta_0,\beta,\xi} \quad & \tfrac{1}{2}\|\beta\|^2 + C\textstyle\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i\left(x_i^T\beta + \beta_0\right) \geq 1-\xi_i, \quad \xi_i \geq 0 \end{aligned} \tag{3}$$

where the cost parameter C acts as a penalisation term to restrict the total number of misclassifications. The optimal value for C can be estimated by cross-validation; x_i is the $i-th$ training example, and y_i is the correct output of the SVM for the $i-th$ training example. The value y_i is +1 for the positive class and –1 for the negative class.

The constraint $y_i\left(x_i^T\beta+\beta_0\right) \geq 1-\xi_i$ and $\xi_i \geq 0$ is equivalent to $\xi_i = \max\left(0, 1-y_i\left(\beta^T x+\beta_0\right)\right)$ **This is defined as the hinge loss.** Hence the learning problem defined in (3) is equivalent to the following unconstrained optimization problem:

$$\min_{\beta_0,\beta} \quad \tfrac{1}{2}\|\beta\|^2 + C\textstyle\sum_{i=1}^{N} max\left(0, 1-y_i\left(\beta^T x+\beta_0\right)\right) \tag{4}$$

Our choice of the hinge loss function is motivated by the following reasons:

- The optimisation problem defined in (3) has the form loss + penalty, which is a familiar paradigm in function estimation.
- Empirical examination of the "hinge" loss function shows that it is reasonable for two-class classification, when compared to other more traditional loss functions such as log-likelihood loss or squared error loss.[3]
- This definition of the cost function is convex, therefore a locally optimal point is globally optimal (provided the optimization is over a convex set, which it is in our case).This is a quadratic optimization problem subject to linear constraints and there is a unique minimum.[3]

Problem (3) is a differentiable convex problem with affine constraints and it satisfies refined Slater's condition i.e., the strict feasibility in Slater's condition for these constraints can be relaxed to feasibility .[1].This allows us to make two important observations, that we subsequently use to define and solve the dual optimisation problem:

- If Slater's condition is satisfied, then strong duality holds and the dual optimum is attained, i.e. x^* is optimal **iff** there exists α^*, ν^* such that the KKT conditions are satisfied at (x^*, α^*, ν^*). This implies that KKT conditions are sufficient and necessary.
- As such, optimising our primal Constraint Optimisation Problem (COP) is equivalent to solving the dual problem. [4]

2.2 Defining the dual optimisation problem

The Lagrange primal function for our COP in (3) is:

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(x_i^T\beta+\beta_0\right) - (1-\xi_i)\right] - \sum_{i=1}^{N}\mu_i\xi_i \tag{5}$$

where $\alpha = [\alpha_1, \alpha_2 ... \alpha_N]^T$ and $\mu = [\mu_1, \mu_2 ... \mu_N]^T$ are the associated Lagrange multipliers.
To solve this optimisation problem we make use of the fact that KKT conditions are both necessary

and sufficient. So we first, minimise the Lagrange primal function with respect to β, β_0, ξ_i. Setting the respective derivatives to zero we get:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{6}$$

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

$$\alpha_i = C - \mu_i, \forall i \tag{8}$$

By substituting (6),(7),(8) in (5) we obtain the Lagrangian dual objective function:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{9}$$

As mentioned above, for our COP we have strong duality therefore minimising the primal function is equivalent to maximising the dual function.
Maximizing the dual function is a simpler convex quadratic programming problem than the primal, and can be solved with standard techniques (See Murray et al.,for example)[5].
The advantage of solving the dual problem over the Lagrange primal problem is that the objective function now only depends on the Lagrangian multipliers, which is easier to be solved analytically. We maximise L_D subject to $0 \leq \alpha_i \leq C$, and $\sum_{i=1}^{N} \alpha_i y_i = 0$. See [3] for further information on the derivation.

We can compactly write the dual optimisation problem as follows:

$$\begin{aligned} \max_{\alpha} \quad & g(\alpha) = \alpha^T \mathbf{1} - \frac{1}{2} \alpha^T D \alpha \\ \text{s.t.} \quad & \alpha^T y = 0 \\ & \alpha \succeq 0 \\ & \alpha \preceq C\mathbf{1} \end{aligned} \tag{10}$$

where y is the vector: $y = (y_1, y_2 \ldots y_N)^T$, D is a symmetric NxN matrix with entries, $y_i y_j x_i^T x_j$, $\mathbf{1}$ is an $N - dimensional$ vector of ones.

For simplicity, we will solve the following equivalent Constraint Optimisation Problem (**COP**):

$$\begin{aligned} \min_{\alpha} \quad & g(\alpha) = \frac{1}{2} \alpha^T D \alpha \ \text{-}\alpha^T \mathbf{1} \\ \text{s.t.} \quad & \alpha^T y = 0 \\ & \alpha \succeq 0 \\ & \alpha \preceq C\mathbf{1} \end{aligned} \tag{11}$$

Once the optimal Lagrange multipliers α^* are determined by solving (11), we can easily obtain the optimal parameters of the SVM decision boundary. From (6) we see that the solution for β has the form:

$$\beta^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i \tag{12}$$

The training examples for which $\alpha_i^* > 0$ are called support vectors which determine the orientation of the optimal hyper-plane, because other examples with $\alpha_i^* = 0$ can be omitted from the expression. Because β^* can be computed via equation (12) from the training data before use, the amount of computation required to evaluate a linear SVM is constant in the number of non-zero support vectors.[6]

To determine the optimal β_0^* we will use the relationship implied by the complementary slackness condition for our primal COP, which requires that:

$$\alpha_i^* \left[y_i \left(x_i^T \beta^* + \beta_0^* \right) - 1 + \xi_i^* \right] = 0, \forall i \tag{13}$$

$$\mu_i^* \xi_i^* = 0, \forall i \tag{14}$$

This enables us to solve $\beta_0^* = y_i - x_i^T \beta^*$ for any support vector x_i with $0 < \alpha i^* < C$

Once we have the hyperplane from solving the optimisation problem above, the decision function for our SVM classification problem can be written as:

$$\widehat{G}_x = sign\left[\widehat{f}(x)\right] = sign\left[x^T \beta^* + \beta_0^*\right] = sign[\sum_{i=1}^{N} \alpha_i^* y_i x^T x_i + \beta_0^*] \tag{15}$$

3 Primal-dual method to solve the SVM optimisation problem

3.1 Rationale

An interior point is a pair of primal and dual variables which satisfy the primal and dual constraints, respectively. Below we provide the main procedure of the primal-dual interior point algorithm for SVM optimization, given the fundamental importance of this optimization technique.

We use this method because for small and moderately sized problems, interior point algorithms are reliable and accurate optimization methods to choose. In particular our problem can be solved efficiently by interior Primal-Dual method which often achieves super-linear convergence. . [1]

Furthermore, Interior Point Methods such as PD provide a guarantee to solve optimization problems in $\mathcal{O}(\sqrt(n) \log(1/\varepsilon))$ iterations and, in practice, display a fast convergence in merely a few iterations almost independent of the problem size. Primal-dual interior-point methods often exhibit super-linear asymptotic convergence.[7]

Interior point methods for optimization exploit the following three tools[2]:

- they use logarithmic barrier functions to “replace” the inequality constraints,
- they apply duality theory to derive the first order optimality conditions which take the form of a system of nonlinear equations, and
- they apply the Newton method to solve this system of nonlinear equations.

3.2 Primal-dual interior point algorithm

Overview of the PD method

- The search directions in a primal-dual interior-point method are obtained from Newton's method, applied to modified KKT equations. [8]
- Primal-dual interior-point methods usually take one Newton step per iteration (no additional loop for the centering step).
- At each iteration, both the primal and dual variables are updated.
- In this method, the primal and dual iterates are not necessarily feasible. Therefore an advantage of the primal-dual algorithms over the barrier method is that they can work when the problem is feasible, but not strictly feasible.
- Primal-dual interior-point methods are often more efficient than the barrier method, especially when high accuracy is required, since they can exhibit better than linear convergence.

Pseudo-code for the Primal-Dual interior point algorithm

Based on the above, we can outline the primal-dual interior-point algorithm for the SVM optimisation problem as follows:

Algorithm 1: Pseudo-code for the Primal-Dual interior point algorithm (SVM Dual Optimisation Problem)

```
Input  : $\alpha_0$ that satisfies the inequalities $\alpha_0 - C\mathbf{1} < 0$ and $\alpha_0 > 0$,
         $\lambda_0 > 0, \mu > 1, \varepsilon_{feas} > 0, \varepsilon > 0$
Output : Solution of the COP (13): $\alpha^* \in \mathbb{R}^{\mathbb{N}}$
1 repeat
2     Determine t. Set $t := \mu m / \widehat{\eta}$
3     Compute primal-dual search direction $\Delta y_{pd}$
4     Line search and update: Determine step length $s > 0$ and set $y := y + s\Delta y_{pd}$
5 until ;
6 $\|r_{prim}\|_2 \leq \varepsilon_{feas}$, $\|r_{dual}\|_2 \leq \varepsilon_{feas}$ and $\widehat{\eta} \leq \varepsilon$
```

After obtaining α^*, we can derive the optimal hyperplane that determines the decision boundary for the SVM problem as follows:$\beta^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i$, while β_0 is obtained as: $\beta_0^* = y_i - x_i^T \beta^*$

Now, we will reformulate the SVM dual optimization problem, in a way that is suitable for the Primal Dual Optimization algorithm. We are solving the COP set out in Equation (11). Introducing the Lagrange multiplier s, z and ν — where s and z are the inequality multiplier consisting of $Nx1$ vectors such that $s \succeq 0$ and $z \succeq 0$, while ν is a scalar representing the equality multiplier—we define the Lagrangian of our COP as follows[1]:

$$L(\alpha; s, z, \nu) = \frac{1}{2}\alpha^T D\alpha - \alpha^T \mathbf{1} + s^T(\alpha - C\mathbf{1}) - z^T\alpha + \nu\alpha^T y \tag{16}$$

Our COP is a convex quadratic problem, i.e., an optimal α^* exists and Slater's constraint qualification holds, so there exist dual optimal $s^* \in \mathbb{R}^{\mathbb{N}}$, $z^* \in \mathbb{R}^{\mathbb{N}}$ and the scalar ν^*, which together with α^* satisfy the The KKT conditions:

$$D\alpha^* - 1 + s^* - z^* + \nu^* y = 0 \text{ (Dual Feasibility)} \tag{17}$$

$$\alpha^{*T}y = 0 \text{ (Primal Feasibility)} \tag{18}$$

$$\alpha^* - C\mathbf{1} \leq 0 \text{ (Primal Feasibility)} \tag{19}$$

$$\alpha^* \geq 0 \text{ (Primal Feasibility)} \tag{20}$$

$$s^{*T}(\alpha^* - C\mathbf{1}) = 0 \text{ (Complementary slackness)} \tag{21}$$

$$z^{*T}\alpha^* = 0 \text{ (Complementary slackness)} \tag{22}$$

$$s \geq 0; z \geq 0 \tag{23}$$

Interior-point methods solve the COP (or the KKT conditions defined above) by applying Newton's method to a sequence of equality constrained problems, or to a sequence of modified versions of the KKT conditions.**The modified KKT conditions** are closely related to the concept of **central path**.

An important property of the central path is that: **Every central point yields a dual feasible point, and hence a lower bound on the optimal value** p^*. More specifically, define $s^*(t) = -1/t(\alpha - C\mathbf{1})$; $z^*(t) = -1/t(-\alpha)$ and $\nu^*(t) = \widehat{\nu}/t$. From the definition of the Central Path, we see that $\alpha^*(t)$ minimises the Lagrangian in (16) for $s = s^*(t)$,$z = z^*(t)$ and $\nu = \nu^*(t)$, which means that $(s = s^*(t), z = z^*(t), \nu^*(t))$ are dual feasible. See [9] for further information on the derivation. Therefore the dual function of our COP is finite and:

$$h(s^*(t), z^*(t), \nu^*(t)^*) = g(\alpha^*(t)) - m/t \tag{24}$$

In particular the duality gap associated with (24) is m/t. As an important consequence we have that $\alpha^*(t)$ is no more than m/t sub-optimal. This confirms the intuitive idea that $\alpha^*(t)$ converges to an optimal point as $t \to \infty$.

We can interpret the central path conditions as a continuous deformation of the KKT optimality conditions. A point α is equal to $\alpha^*(t)$ iff there exists s, z and ν such that:

$$D\alpha^* - 1 + s^* - z^* + \nu^* y = 0 \text{ (Dual Feasibility)} \tag{25}$$

$$\alpha^{*T}y = 0 \text{ (Primal Feasibility)} \tag{26}$$

$$\alpha^* - C\mathbf{1} \leq 0 \text{ (Primal Feasibility)} \tag{27}$$

$$\alpha^* \geq 0 \text{ (Primal Feasibility)} \tag{28}$$

$$-s^{*T}(\alpha^* - C\mathbf{1}) = 1/t \text{ (Modified CS)} \tag{29}$$

$$z^{*T}\alpha^* = 1/t \text{ (Modified CS)} \tag{30}$$

$$s \geq 0; z \geq 0 \tag{31}$$

The only difference to (KKT) is the complementarity slackness condition being replaced by $-s^{*T}(\alpha^* - C\mathbf{1}) = 1/t$ and $z^{*T}\alpha^* = 1/t$. Consequently, for large t, $\alpha^*(t)$,$s^*(t)$,$z^*(t)$ and $\nu^*(t)$ almost satisfy the KKT conditions.

Primal dual search direction

The search directions in a primal-dual interior-point algorithm are obtained from the Newton's method, applied to modified KKT equations.[10] For simplicity we concatenate the two inequality multipliers-namely $s \in \mathbb{R}^{\mathbb{N}}$, $z \in \mathbb{R}^{\mathbb{N}}$- in one single vector λ which is defined as: $\lambda = \begin{bmatrix} s \\ z \end{bmatrix}$. We start with compactly writing the modified KKT conditions for our SVM COP, expressed as $r_t(\alpha, \lambda, \nu)$=0.

$$0 = r_t(\alpha, \lambda, \nu) = \begin{bmatrix} \nabla g(\alpha) + J^T(\alpha)\lambda + A^T \nu \\ \text{-diag}(\lambda)F(\alpha) - (1/t)1 \\ A\alpha - b \end{bmatrix} := \begin{bmatrix} r_{dual} \\ r_{cent} \\ r_{prim} \end{bmatrix} \tag{32}$$

where $t > 0$, $\nabla g(\alpha) = D\alpha - \mathbf{1}$, $F(\alpha) = \begin{bmatrix} \alpha - C\mathbf{1} \\ -\alpha \end{bmatrix}$,$J^T(\alpha) = \begin{bmatrix} \text{diag}(\mathbf{1}) \\ \text{-diag}(\mathbf{1}) \end{bmatrix}$, $A = y^T$ and $b = 0$.

If α,λ ν satisfy $r_t(\alpha, \lambda, \nu) = 0$ and inequality constraints are strictly satisfied then $\alpha = \alpha^*(t)$, $\lambda = \lambda^*(t)$ and $\nu = \nu^*(t)$.In particular α is primal feasible and λ, ν are dual feasible with duality gap m/t.

Newton step for the solution of $r_t(\alpha, \lambda, \nu) = 0$ for a fixed t, at $y = (\alpha, \lambda, \nu)$, is determined as follows:

$$r_t(y + \Delta y) \approx r_t(y) + Dr_t(y)\Delta y = 0 \tag{33}$$

where $\Delta y = (\Delta\alpha, \Delta\lambda, \Delta\nu)$ is the primal dual search directions, i.e. $\Delta y = -(Dr_t(y))^{-1} r_t(y)$. In terms of α, λ, ν we have:

$$\begin{bmatrix} \nabla^2 g(\alpha) + \lambda \nabla^2 F(\alpha) & J(\alpha)^T & A^T \\ \text{-diag}(\lambda)J(\alpha) & \text{-diag}(F(\alpha)) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = - \begin{bmatrix} r_{dual} \\ r_{cent} \\ r_{prim} \end{bmatrix} \tag{34}$$

where, $\nabla^2 g(\alpha) \in \mathbb{R}^{\mathbb{N} * \mathbb{N}} = D$. The primal-dual search direction $\Delta y_{pd} = (\Delta\alpha_{pd}, \Delta\lambda_{pd}, \Delta\nu_{pd})$ is defined as the solution of equation (35).

The surrogate duality gap[11]

- In the primal-dual interior point methods, the iterates $(\alpha^k, \lambda^k, \nu^k)$ are not necessarily feasible, except in the limit as the algorithm converges.Hence, we cannot easily evaluate duality gap $\eta(k)$ in the k-th step.

- Instead, we define the surrogate duality gap as: $\eta(\alpha, \lambda) = -F(\alpha)^T \lambda$

- The surrogate gap is the duality gap if α were primal feasible and λ, ν were dual feasible i.e. if $r_{prim} = 0$, $r_{dual} = 0$. Note that value of t corresponds to the surrogate duality gap $\eta \approx m/t \rightarrow t = m/\eta$.

- The parameter t is set to a factor $\mu m/\eta$, which is the value of t associated with the current surrogate duality gap . If α, λ, ν were central, with parameter t (and therefore with duality gap m/t), then we would increase t by the factor μ. We have assumed $\mu = 10$,as a reasonable empirical value.

- The primal-dual interior-point algorithm terminates when α is primal feasible and λ, ν are dual feasible (within the tolerance ε_{feas}) and the surrogate gap is smaller than the tolerance ε.

Line Search

- The line search in the primal-dual interior point method is a standard backtracking line search, based on the norm of the residual and modified to ensure that the inequality constraints are strictly satisfied.

- We denote the current iterate as α,λ,ν and the next iterate as α^+,λ^+,ν^+ i.e. $\alpha^+ = \alpha + s\Delta\alpha_{pd}$, $\lambda^+ = \lambda + s\Delta\lambda_{pd}$, $\nu^+ = \nu + s\Delta\nu_{pd}$. The residual evaluated at y^+ will be denoted as r^+. We first compute the largest positive step not exceeding one, that gives $\lambda^+ \succeq 0$ i.e.:

$$s^{max} = sup(s \in [0,1] \mid \lambda + s\Delta\lambda \succeq 0) = min[1, min(-\lambda_i/\Delta\lambda_i \mid \Delta\lambda_i < 0] \tag{35}$$

- We start the backtracking with $s = 0.99s^{max}$, and multiply s by $\beta \in (0,1)$ until we have $\alpha^+ - C\mathbf{1} < 0$ and $-\alpha^+ < 0$.We continue multiplying s by β until we have:

$$\|r_t(\alpha^+,\lambda^+,\nu^+)\|_2 \leq (1-\widehat{\alpha}s)\|r_t(\alpha,\lambda,\nu)\|_2 \tag{36}$$

- Common choices for the backtracking parameter $\widehat{\alpha}$ is typically in the range of 0.01 to 0.1, and β is chosen in the range 0.3 to 0.8. The backtracking parameters in our model are set to 0.1 and 0.8 respectively.

- One iteration of the primal-dual interior-point algorithm is the same as one step of the infeasible Newton method, applied to solving $r_t(\alpha,\lambda,\nu) = 0$, but modified to ensure $\lambda > 0$, $\lambda^+ - C\mathbf{1} < 0$ and $-\lambda^+ < 0$.

3.3 Global and Local Convergence

Theoretical Local Convergence

- Primal-dual interior-point methods often exhibit super-linear asymptotic convergence. [8] Suppose that an inexact Newton method generates the sequence y_k that converges to y^* superlinearly .Under the additional assumption of the invertibility of the Jacobian $Dr(y^*)$ at the solution, the **superlinear convergence of the iteration sequence is achieved**. See [7] for further information on the proof.

- **The line search for the primal-dual method always terminates in a finite number of steps.**[8] This is based on the fact that once the norm of the residual is small enough, the algorithm takes full steps (which implies that feasibility is achieved), and convergence is subsequently quadratic towards the end. The norm of the residual is reduced by at least a fixed amount in each iteration before the region of quadratic convergence is reached. Since the norm of the residual cannot be negative, this shows that within a finite number of steps, the residual will be small enough to guarantee full steps, and quadratic convergence.[9] However, this is rarely observed in practice because it holds under very strict assumptions (see [9]), and it is a very local statement which is more likely observed when closely approaching the solution.

- Interior Point Methods such as PD provide a **guarantee to solve optimization problems in** $\mathcal{O}(\sqrt(n)\log(1/\varepsilon))$ iterations and, in practice, display a fast convergence in merely a few iterations almost independent of the problem size.[12]

Theoretical Global Convergence

The global convergence of the Primal Dual Interior Point method has been theoretically established, under the assumption that the derivative of the function $r(y)$ is Lipschitz continuous, [7] i.e. for (α,λ,ν), $(\tilde{\alpha},\tilde{\lambda},\tilde{\nu}) \in S$, $Dr(\alpha,\lambda,\nu)$ satisfies the Lipschitz condition:

$$\left\|Dr(\alpha,\lambda,\nu) - Dr(\tilde{\alpha},\tilde{\lambda},\tilde{\nu})\right\|_2 \leq L\left\|(\alpha,\lambda,\nu) - (\tilde{\alpha},\tilde{\lambda},\tilde{\nu})\right\|_2 \tag{37}$$

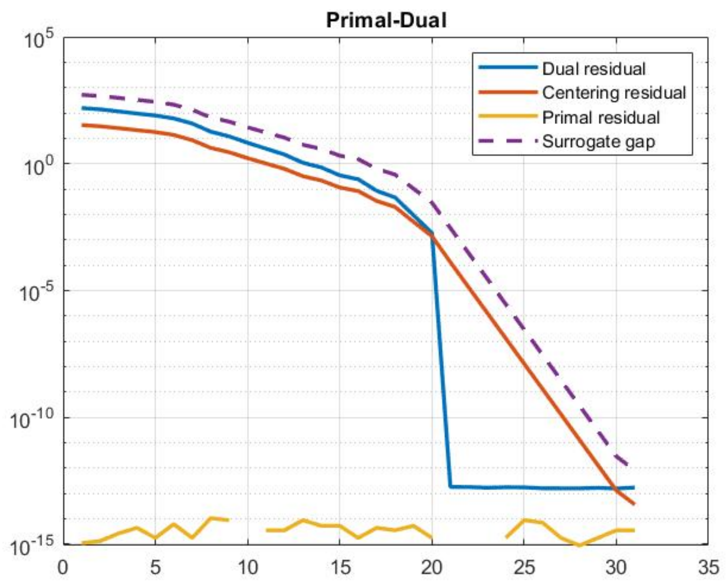
This is equivalent to $\nabla^2 g(\alpha)$ satisfying a Lipschitz condition.[9]

3.4 Optimisation results for the Primal-Dual Method

Our SVM optimisation problem set out in Equation (11) consists of: i) 400 inequalities: we have two pairwise inequalities (i,e, $\alpha \succeq 0$ and $\alpha \preceq C\mathbf{1}$) for a total of 200 observations in the training set;ii) 1 equality: $\alpha^T y = 0$ and two variables.

Figure 2 shows the progress of the primal-dual interior-point method for the SVM COP applied to our data. The plot shows the surrogate gap η_k and the norm of the primal residual, dual residual and centering residual for each iteration.
We initialise our method, starting from a point that satisfies the equality constraint i.e. our initial α_0 is primal feasible. The plot shows that the norm of the residuals converge to zero rapidly, and becomes zero to numerical precision in 31 iterations. The surrogate gap also converges rapidly. In line with the theoretical expectation, the primal-dual interior-point algorithm terminates when α is primal feasible and λ, ν are dual feasible (within the tolerance ε_{feas}) and the surrogate gap is smaller than the tolerance ε.

Figure 2: Progress of the Primal-Dual Interior point method for the SVM COP



3.4.1 Convergence results

It is theoretically expected that under the assumption of the invertibility of the Jacobian $Dr(y^*)$ at the solution, the primal-dual interior point method should converge superlinearly.

In our SVM COP we observe that the Jacobian $Dr(y^*) = \begin{bmatrix} \nabla^2 g(\alpha^*) + \lambda^* \nabla^2 F(\alpha^*) & J(\alpha^*)^T & A^T \\ \text{-diag}(\lambda^*) J(\alpha^*) & \text{-diag}(F(\alpha^*)) & 0 \\ A & 0 & 0 \end{bmatrix}$, is invertible as indicated by the non-zero eigenvalues of the KKT matrix. **Therefore we expect superlinear convergence.**

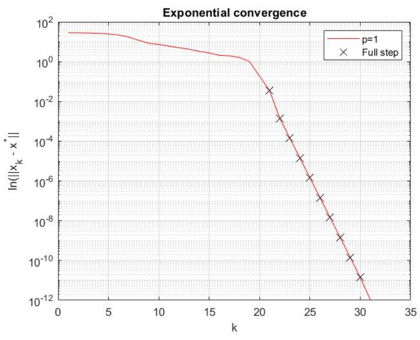
Figure 3 plots the Exponential and Q-convergence of the Primal-Dual method in the semi-log scale. We observe an almost linear relationship with a negative slope in the exponential convergence plot. Exponential convergence is essentially measured as : $\left\|\alpha^k - \alpha^*)\right\| = cq^k$, where q is bounded in (0.1).

The declining trend and more importantly the fact that the line goes to zero as the number of iterations increases, supports the theoretical expectation of the super-linear asymptotic convergence of the Primal-Dual method.

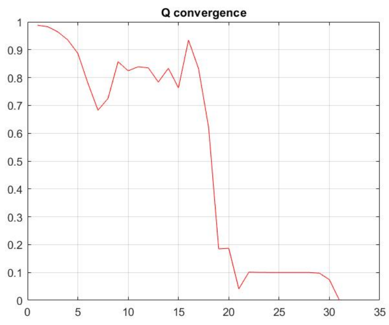
The same finding is corroborated by he examination of the Q-convergence: $\frac{\left\|\alpha^{k+1} - \alpha^*\right\|}{\left\|\alpha^k - \alpha^*\right\|^p} \leq r$ for p=1, where r is the convergence rate and p is the convergence order.

Figure 3: Exponential Convergence and Q-convergence of the PD method for SVM COP

(a) Exponential Convergence



(b) Q-Convergence



To prove global convergence we need to show that the derivative of the function $r(y)$ is Lipschitz continuous i.e. $\left\|Dr(\alpha, \lambda, v) - Dr(\tilde{\alpha}, \tilde{\lambda}, \tilde{v})\right\|_2 \leq L \left\|\alpha, \lambda, v) - (\tilde{\alpha}, \tilde{\lambda}, \tilde{v})\right\|_2$. We know that $Dr(y)$ satisfies a Lipschitz condition if and only if $\nabla^2 g(\alpha)$ is Lipschitz continuous[9].

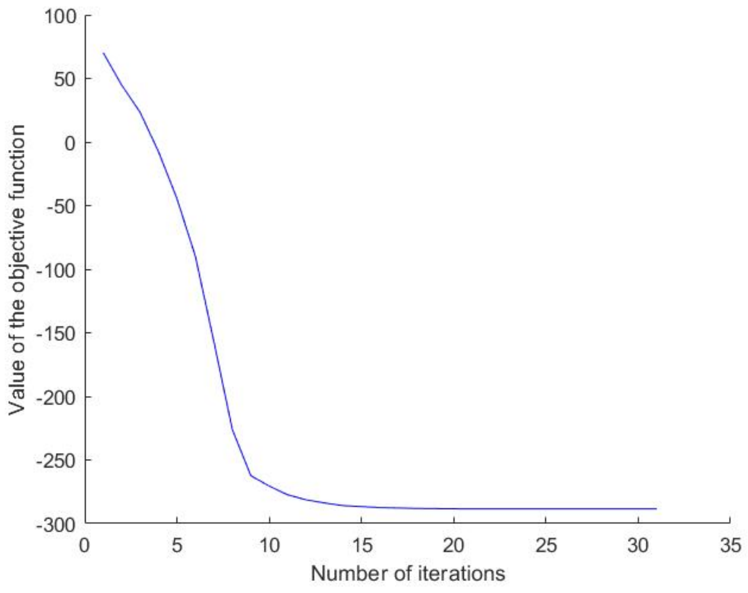
Given that $\nabla^2 g(\alpha)$ is equal to matrix D it satisfies by definition the Lipschitz condition. Therefore we conclude that $Dr(y)$ **is Lipschitz continuous and global convergence is expected for this method**.

3.4.2 Performance of the Primal-Dual method for SVM Optimisation

We make the following observations regarding the performance of the Primal-Dual interior point method for SVM Optimisation:

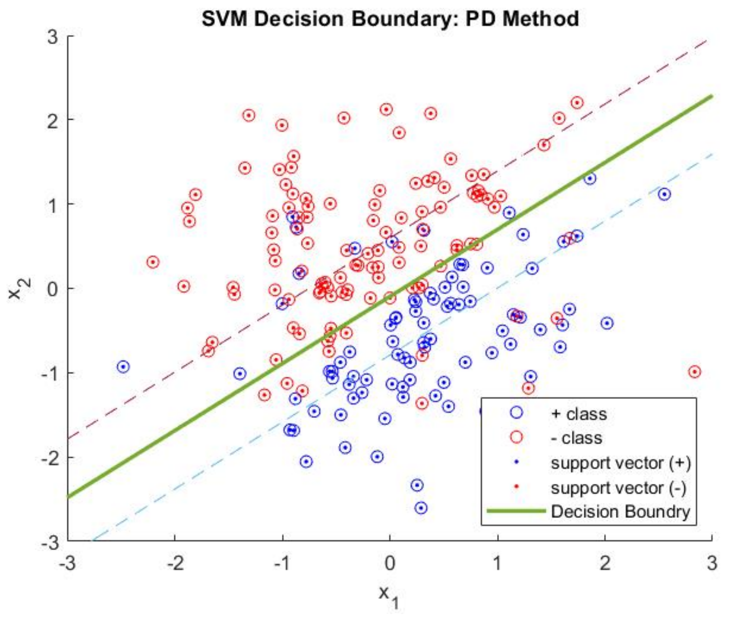
- **Computational Efficiency**: This method converges to the optimal solution in **31 iterations**. It requires **8.27** seconds to complete the optimisation. The minimum value of the objective function is (-288.4). Figure below shows the progress of the PD method in terms of minimising the value of the objective function.

Figure 4: Progress of the objective function based on Primal-Dual Interior Point Method



- **Optimal Hyperplane**: The optimal hyperplane is defined by the following coefficients: $\beta^* = [1.1460; -1.4417]$ and $\beta_0^* = -0.1400$. The number of support vectors corresponding to non-zero α^*-s is 200. The optimal decision boundary of the SVM and the classification performance is visualised in the figure below.

Figure 5: Optimal Decision boundary of the SVM COP based on the Primal-Dual Method



- **Classification performance**: We have measured the classification performance of our trained model in the test set, which consists of 100 observations. We observe that the accuracy of the SVM based on Primal-Dual optimisation is **75%** while the F-measure is **71.3%**. The accuracy measures the number of correct classifications out of the total number of predictions. Whereas, the F-score is calculated from the precision and recall score. *(Precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive).*

4 Sequential Minimal Optimisation Method (SMO)

4.1 Rationale

Training a support vector machine requires the solution of a very large quadratic programming (QP) optimization problem. SMO breaks this large QP problem into a series of smallest QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets. Because matrix computation is avoided, SMO scales somewhere between linear and quadratic in the training set size. SMO's computation time is dominated by SVM evaluation, hence SMO is fastest for linear SVMs and sparse data sets. [6]

At every step, SMO chooses two Lagrange multipliers to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values. The advantage of SMO lies in the fact that solving for two Lagrange multipliers can be done analytically. Thus, numerical QP optimization is avoided entirely.In addition, SMO requires no extra matrix storage.

The KKT conditions can be used in the SMO method, to check for convergence to the optimal point. We are restating the KKT conditions as follows:

$$\alpha_i = 0 \Leftrightarrow y_i(\beta^T x^{(i)} + \beta_0) \geq 1 \tag{38}$$

$$0 < \alpha_i < C \Leftrightarrow y_i(\beta^T x^{(i)} + \beta_0) = 1 \tag{39}$$

$$\alpha_i = C \Leftrightarrow y_i(\beta^T x^{(i)} + \beta_0) \leq 1 \tag{40}$$

where $(\beta^T x^{(i)} + \beta_0)$ is the output of the SVM for the $i-th$ training example. Notice that the KKT conditions can be evaluated on one example at a time, which is useful in the construction of the SMO algorithm. In other words, any α_i-s that satisfy these properties for all i will be an optimal solution to the SVM optimization problem. The SMO algorithm iterates until all these conditions are satisfied (to within a certain tolerance) thereby ensuring convergence.

4.2 SMO Algorithm

SMO decomposes the overall QP problem into QP sub-problems, using Osuna's theorem to ensure convergence.[13] The theorem proves that the large QP problem can be broken down into a series of smaller QP sub-problems. As long as at least one example that violates the KKT conditions is added to the examples for the previous sub-problem, each step will reduce the overall objective function and maintain a feasible point that obeys all of the constraints. Therefore, a sequence of QP sub-problems that always add at least one violator will be guaranteed to converge.

There are two components to SMO: 1. an analytic method for solving for the two Lagrange

multipliers and 2. a heuristic for choosing which multipliers to optimize.
The SMO algorithm consists of three main steps:

- The SMO algorithm selects two parameters α_i and α_j;
- It optimises the objective value jointly for both α-s;
- Finally,it adjusts the b parameter based on the new α-s;

This process is repeated until α-s converge.

Step 1: Selecting α parameters

In order to solve for the two Lagrange multipliers, SMO first computes the constraints on these multipliers and then solves for the constrained minimum. Much of the full SMO algorithm is dedicated to heuristics for choosing which α_i and α_j to optimize. For large data sets, this is critical for the speed of the algorithm, since there are $Nx(N-1)$ possible choices for α_i and α_j, and some will result in much less improvement than others. However, our data consists of only 200 observation in the training set therefore we will use a simplified version of the SMO algorithm which employs a much simpler heuristic.[14]

- We iterate over all α_i for $i = 1,2...N$. The choice of the first heuristic provides the outer loop of the SMO algorithm. The outer loop first iterates over the entire training set, determining whether each example violates the KKT conditions. If an example violates the KKT conditions, it is then eligible for optimization.
- If α_i does not fulfill the KKT conditions to within some numerical tolerance, we select α_j at random from the remaining $N-1$ α-s and attempt to jointly optimise α_i and α_j.
- If none of the α-s are changed after a few iterations over all the α_i-s, then the algorithm terminates.
- It is important to note that by employing this simplification, the algorithm is no longer guaranteed to converge to the global optimum (since we are not attempting to optimize all possible α_i, α_j pairs).

Step 2: Optimising α_i and α_j

Having chosen the Lagrange multipliers α_i and α_j to optimize, we first compute constraints on the values of these parameters, then we solve the constrained maximization problem. First we want to find bounds L and H such that $L \leq \alpha_j \leq H$ must hold in order for α_j to satisfy the constraint that $0 \leq \alpha_j \leq C$. These are given by:

$$\text{if } y_i \neq y_j, L = max(0, \alpha_j - \alpha_i),\ H = min(C, C + \alpha_j - \alpha_i) \tag{41}$$

$$\text{if } y_i = y_j, L = max(0, \alpha_i + \alpha_j - C),\ H = min(C, \alpha_i + \alpha_i) \tag{42}$$

Now we want to find α_j, so as to minimise our objective function. If this value ends up lying outside the bounds L and H, we simply adjust the value of α_j to lie within this range. The optimal α_j is given by:

$$\alpha_j := \alpha_j - \frac{y_j(E_i - Ej)}{\eta} \tag{43}$$

where: $E_k = f(x_k) - y_k$. We can think of E_k as the error between the SVM output on the $k-th$ example and the true label y_k; η is defined as : $\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j$.

Then we adjust the value of α_j to lie within the range $[L, H]$ in the following way:

$$\alpha_j = \begin{cases} H \text{ if } \alpha_j > H \\ \alpha_j \text{ if } L \leq \alpha_j \leq H \\ L \text{ if } \alpha_j < L \end{cases} \tag{44}$$

Finally, having solved for α_j we want to find the value for α_i. This is given by:

$$\alpha_i := \alpha_i + y_i y_j (\alpha_j^{(old)} - \alpha_j) \tag{45}$$

where $\alpha_j^{(old)}$ is the value before α_j optimisation. .
For our purposes, if $\eta = 0$, we treat this as a case where we cannot make progress on this pair of α-s.

Step 3: Computing the b threshold

After optimising α_i and α_j, we select the threshold b such that the KKT conditions are satisfied for the i-th and j-th examples. If after optimizing, α_i is not at the bounds i.e.$(0 < \alpha_i < C)$, then the following threshold b_1 is valid, since it forces the SVM to output y_i when the input is x_i:

$$b_i = b - E_i - y_i(\alpha_i - \alpha_i^{(old)}) x_i \cdot x_i - y_j(\alpha_j - \alpha_j^{(old)}) x_i \cdot x_j \tag{46}$$

Similarly, the following threshold b_2 is valid if $(0 < \alpha_j < C)$

$$b_j = b - E_j - y_i(\alpha_i - \alpha_i^{(old)}) x_i \cdot x_j - y_j(\alpha_j - \alpha_j^{(old)}) x_j \cdot x_j \tag{47}$$

If both $(0 < \alpha_i < C)$ and $(0 < \alpha_j < C)$, then both these thresholds are valid and they will be equal. If both new α-s are at the bounds (i.e. $\alpha_i = 0$ or $\alpha_i = C$ and $\alpha_j = 0$ or $\alpha_j = C$) then all the thresholds between b_1 and b_2 satisfy the KKT conditions, so we let $b := (b_1 + b_2)/2$. This gives the complete equation for b.

$$b := \begin{cases} b_1 \text{ if } 0 < \alpha_i < C \\ b_2 \text{ if } 0 < \alpha_j < C \\ (b_1 + b_2)/2 \text{ otherwise} \end{cases} \tag{48}$$

4.3 Global and Local Convergence

Theoretical Local Convergence Under the assumptions that the kernel matrix D is positive definite and the optimization problem is non-degenerate, we expect the SMO method to display **linear convergence**. It is well known that the dual function value sequence generated by SMO has a linear convergence rate when the kernel matrix is positive definite and sublinear convergence is also known to hold for a general matrix.[15]

Assumptions for linear convergence[16]

- **Assumption 1**: Kernel matrix D is positive definite. i.e. $min(\text{eig}(D)) > 0$;

- **Assumption 2 (Non-degeneracy)**: For the optimal solution α^*, we have $\nabla g(\alpha^*)_i + h^* y_i \neq 0$ if $\alpha_i^* = 0$ or C.

The following two theorems are the main results on the linear convergence of SMO. They require two technical Lemmas which are stated in [16].
Theorem 1: There is $c < 1$ such that after k is large enough:

$$(\alpha^{(k+1} - \alpha^*)^T D(\alpha^{(k+1} - \alpha^*)^T \leq c(\alpha^{(k)} - \alpha^*)^T D(\alpha^{(k)} - \alpha^*) \tag{49}$$

The linear convergence of the objective function is as follows:
Theorem 2: There is $c < 1$ such that after k is large enough:

$$g(\alpha^{(k+1)}) - g(\alpha^*) \leq c(g(\alpha^k) - g(\alpha^*)) \tag{50}$$

What if matrix *D* is not positive definite?

As shown in [15], sublinear convergence is known to hold for a general matrix D. The authors in [15] have also shown that when applied to hard–margin, i.e., linearly separable SVM problems, a linear convergence rate holds for the SMO algorithm without any condition on the kernel matrix. However, the latter is not particularly relevant for our *SVM soft margin* problem.

Theoretical Global Convergence
Given that D is positive definite and COP is a strictly convex programming problem, there exists unique global optimum α^*. This implies that the whole sequence of iterates α^k of the SMO method converges to α^*.[16]

What if matrix *D* is not positive definite?

Authors in [17] have shown that the conclusion of global convergence holds for a general matrix too and it is not necessary to show that D is positive definite i.e. $min(\text{eig}(D)) > 0$. Therefore for the SMO method there exists a unique global optimum α^* of the SVM COP even for a general matrix D. This implies that the whole sequence of iterates α^k of the SMO method converges to α^*. However, it is important to note that by employing the simplified SMO Algorithm describes in (4.2), the method presented in this report is no longer guaranteed to converge to the global optimum (since we are not attempting to optimize all possible α_i, α_j pairs.

4.4 Optimisation results for the SMO Method

4.4.1 Convergence results

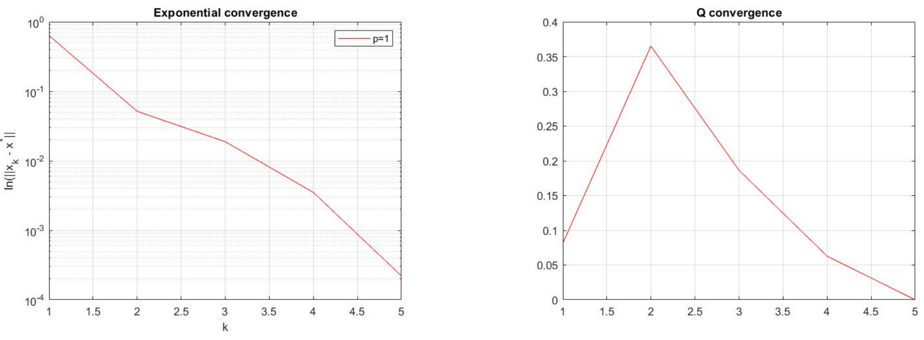
It is theoretically expected that for a general matrix D SMO should display sublinear convergence and if matrix D is positive definite then the convergence should be linear.
For our SVM COP we observe that matrix D is positive semi-definite (i.e. it is a symmetric matrix and all its eigenvalues are non-negative). Therefore, we expect sublinear as opposed to linear convergence for our data.

Figure 6, plots the exponential and Q-convergence of the SMO method in the semi-log scale. When examining the exponential convergence, we observe a sub-linear relationship (consistent with the theory) for the first iterations which becomes slightly linear with a negative slope towards the end. In terms of the Q-convergence we observe that the convergence order is consistently less than 1.

Figure 6: Exponential Convergence and Q-convergence of the SMO method

(a) Exponential Convergence

(b) Q-Convergence



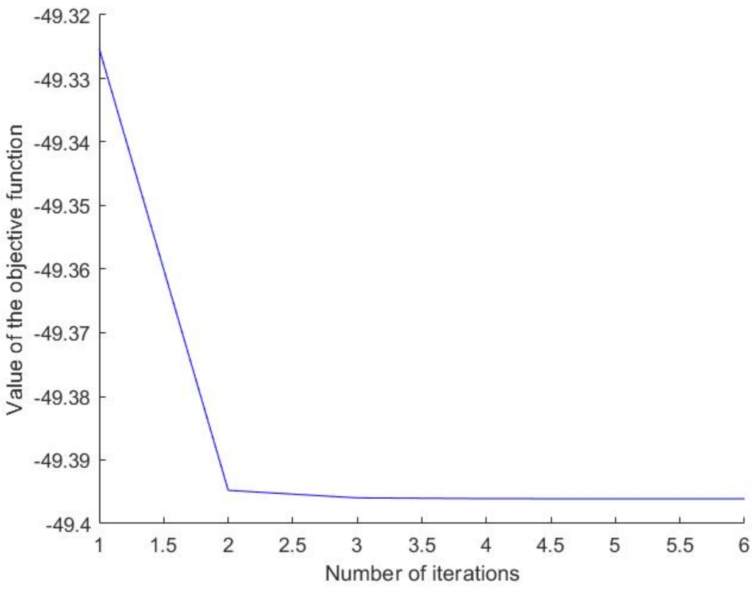
In terms of global convergence, theoretically there exists a unique global optimum α^* of the SVM COP even for a general matrix D. This implies that the whole sequence of iterates α^k of the SMO method converges to α^*. However, for our simplified SMO algorithm global convergence is no longer guaranteed since we are not attempting to optimize all possible α_i, α_j pairs. [14]

4.4.2 Performance of the SMO method for SVM Optimisation

We make the following observations regarding the performance of the SMO method for SVM Optimisation:

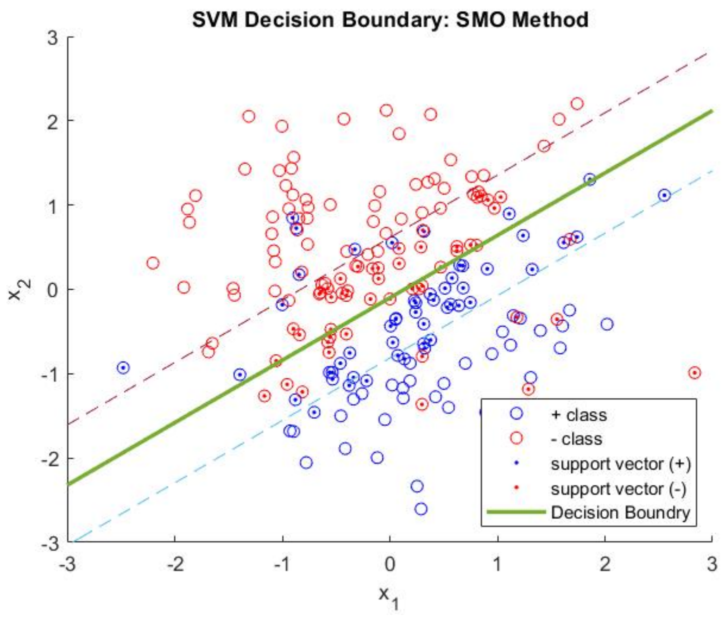
- **Computational Efficiency**: This method converges to the optimal solution in **6 iterations**. It requires **39.1** seconds to complete the optimisation. The minimum value of the objective function is (-49.4). Figure 7 below shows the progress of the SMO method in terms of minimising the value of the objective function.

Figure 7: Progress of the objective function based on SMO Method



- **Optimal Hyperplane**: The optimal hyperplane is defined by the following coefficients: $\beta^* = [1.0372; -1.4007]$ and $\beta_0^* = -0.1400$. The number of support vectors corresponding to non-zero α^*-s is 102. The optimal decision boundary of the SVM and the classification performance is visualised in the figure below.

Figure 8: Optimal Decision boundary of the SVM COP based on the SMO Method



- **Classification performance**: We have measured the classification performance of our trained model in the test set. We observe that the accuracy of the SVM based on the SMO method is **72%** while the F-measure is **66.7%**.

5 Conclusion

We observe that the Primal-Dual method displays a better performance than the SMO for our SVM optimisation problem. The performance is evaluated in terms of computational efficiency (i.e. how expensive each method is in terms of computational time and numbers of iterations) and classification accuracy. The corresponding results for each method are described in detail in sections 3.4 and 4.4.

The relative over-performance of the primal dual method is attributed to the following factors:

- Primal-dual methods are proved to be reliable and accurate optimisation techniques for small and moderately sized problems. Whereas, the SMO method is more applicable for large-scale problems as the main advantage of the SMO lies in the fact that solving for two Lagrange multipliers can be done analytically, numerical optimization is avoided entirely.
- Our optimisation problem is solved efficiently by the primal dual method because this method asymptotically achieves super-linear convergence. Whereas, for a general kernel matrix D, the SMO method achieves only sublinear convergence.
- Finally,Primal-Dual methods provide a guarantee to solve optimization problems in $\mathscr{O}(\sqrt(n)\log(1/\varepsilon))$ iterations and, in practice, display a fast convergence in merely a few iterations almost independent of the problem size. Global convergence is also guaranteed for the SMO method with a general kernel matrix, however in our implementation we are using a simplified version of the SMO algorithm for which the global convergence is no longer guaranteed since we are not attempting to optimize all possible α_i, α_j pairs.

6 Appendix A: Matlab Implementation of the PD Method

```
clear all;clc;
data=csvread('data_train2.csv');
%z-score normalization
data(:,1:end-1)=zscore(data(:,1:end-1));
%%Initial Visualisation of data
x = data(:,1:end-1);
y = data(:,end);

[N,M]=size(x);
%% Setting up the Primal Dual objective function
% Finding D
C=3; %penalty term for mis-classifications

D = zeros(length(x));
for i = 1: length(x)
    for j = 1:length(x)
        D(i,j) = y(i)*y(j)*x(i, 1:2)*x(j, 1:2)';
    end
end

eig(D)
% Objective function
F.f = @(alpha_pd) 1/2*alpha_pd'*D*alpha_pd - alpha_pd'*ones(length(alpha_pd),1);
F.df = @(alpha_pd) D*alpha_pd - ones(length(alpha_pd),1);
F.d2f = @(alpha_pd) D;

% Inequality constraints
constraint2.f = @(alpha_pd) [alpha_pd-C*ones(length(alpha_pd),1); -alpha_pd];
constraint2.df = @(alpha_pd) [diag(ones(length(alpha_pd),1)'); diag(-ones(length(
    alpha_pd),1)')];
for i =1:2*N
    HessConst(:,:,i) = zeros(N, N);
end
constraint2.d2f = @(alpha_pd) HessConst;

% Structure to store inequality constraints
ineqConstraint = constraint2; %

% Equality constrain x^Ty = 0
eqConstraint.A = y';
eqConstraint.b = zeros(1,1);
% Structure to store equality constraints
eqConstraint = eqConstraint;



%% Parameters
%initialise alpha_pd0 for the dual optimisation problem so that it satisfies the
    equality constraint.
alpha_pd0_temp = 0.1*ones(length(x)-1,1); % choose n-1 parameters arbitrarily and
     compute the last one so that the equality constraint is satisfied
alpha_pd0_end=(-alpha_pd0_temp'*y(1:length(x)-1))/y(end)
alpha_pd0=[alpha_pd0_temp; alpha_pd0_end]
% -sum(x0(2:length(x0))'*y(2:length(x0)))/y(1)
lambda0 = ones(length(x)*2,1);% ensure: lambda0 > 0. This is the vector of
    inequality multipliers
```

```
nu0 = ones(1,1); %scalar for equality multiplier
mu = 10; % in (3, 100);
t = 1;
tol = 1e-12; %stopping criteria
tolFeas = 1e-12; %stopping criteria
maxIter = 500;

% Backtracking options
opts.maxIter = 30;
opts.alpha = 0.1; %1e-4; %0.1;
opts.beta = 0.8; %0.5;

%% Minimisation using the InteriorPoint_PrimalDual function
tic
[alpha_pd_k, fPD, tPD, nIter, infoPD] = interiorPoint_PrimalDual(F,
    ineqConstraint, eqConstraint, alpha_pd0, lambda0, nu0, mu, tol, tol, maxIter,
     opts);
% Assign values
alpha_pd_Min = alpha_pd_k;
fMin = F.f(alpha_pd_k);
toc
%% Iterate plot
iterations=1:nIter;
Fval=[]
for i=1:nIter
    input=infoPD.xs(:,i);
    Fval=[Fval,F.f(input)];
end
figure;
hold on
plot(nIter,Fval,'b')
xlabel('Number of iterations')
ylabel('Value of the objective function')
hold off
%% Primal-Dual: residuals, surrogate gap
figure;
semilogy(infoPD.r_dual, 'LineWidth', 2); title('Primal-Dual')
hold on;
semilogy(infoPD.r_cent, 'LineWidth', 2);
semilogy(infoPD.r_prim, 'LineWidth', 2);
%semilogy(infoPD.r_prim + infoPD.r_dual + infoPD.r_cent, 'LineWidth', 2);
semilogy(infoPD.surgap, 'LineWidth', 2, 'Linestyle','--');
legend('Dual residual', 'Centering residual', 'Primal residual', 'Surrogate gap')
    ;
grid on;

%% Empirical convergence rates

% repeat for PD and barrier

conPD = convergenceHistory(infoPD, [], F, 2); % using 2-norm.
%conPD = convergenceHistory(infoPD, xBar, F, 2); % using 2-norm. (for C, PD
    struggles to reduce the dual residual, one of the eigevalues of KKT matrix
    close to 0/)


con = conPD; %{conPD, conBar}

```

```
% Q convergence: $||x_{k+1} - x^*|| / ||x_{k} - x^*||^p <= r$, convergence rate
    $r \in (0,1)$, convergence order $p$
p = 1; % convergance rate
rs1 = con.x(2:end)./(con.x(1:end-1).^p);
%p=2; rs2 = con.x(2:end)./(con.x(1:end-1).^p);
figure, plot(rs1, 'r'); hold on; grid on; title('Q convergence');
%hold on, plot(find(info.FRsteps(2:end)), rs1(logical(info.FRsteps(2:end))), 'xk
    ', 'MarkerSize', 10);
%legend('p=1', 'FR step')
% 0 < r < 1 indicates linear convergence (after initial phase) for feasible x0

% Exponential convergence: ||x_{k} - x^*|| = c q^k, c>0, bounded, q in (0,1)
% ln ||x_{k} - x^*|| = ln c + (ln q)*k    [linear function in semilogy with slope
     (ln q)<0 and offset (ln c)]
% Note that exponential convergence is essentially Q convegence ||x_{k+1} - x^*||
     / ||x_{k} - x^*||^p <= r for r=q and p=1
figure, semilogy(con.x, 'r'); grid on; title('Exponential convergence'); xlabel('
    k'); ylabel('ln(||x_{k} - x^*||')
if isfield(con,'x_nw') % Barrier
figure, semilogy(con.x_nw, 'r'); grid on; title('Exponential convergence (Newton
    iterates)'); xlabel('k'); ylabel('ln(||x_{k} - x^*||')
hold on, semilogy(find(infoBar.alphas_nw > 1-eps), con.x_nw(infoBar.alphas_nw >
    1-eps), 'xk', 'MarkerSize', 10);
legend('p=1', 'Full step')
else %PD
hold on, semilogy(find(infoPD.s > 1-eps), con.x(infoPD.s > 1-eps), 'xk', '
    MarkerSize', 10);
legend('p=1', 'Full step')
end
% Linear convergence after initial phase for feasible x0

% This is not a right way of looking at suspected linear convergence.
% Algebraic convergence: ||x_{k} - x^*|| = c k^{-p}, c>0, bounded
% ln ||x_{k} - x^*|| = ln c + -p*(ln k)     [linear function in log log with
    slope -p and offset (ln c)]
%figure, loglog(con.x, 'r'); grid on; title('Algebraic convergence'); xlabel('ln(
    k)'); ylabel('ln(||x_{k} - x^*||')

%% Visualization of Data
figure
hold on
scatter(x(y==1,1),x(y==1,2),'+g')
scatter(x(y==-1,1),x(y==-1,2),'.r')
xlabel('{x_1}')
ylabel('{x_2}')
legend('Positive Class','Negative Class','Location','southeast')
title('Data for classification')
hold off

%% Retriving betas
Beta=zeros(M,1)';
alpha=alpha_pd_Min;
Xs=x(alpha>0,:); Ys=y(alpha>0);
Support_vectors=size(Xs,1);

Xs
Beta=(alpha(alpha>0).*Ys)'*Xs

```

```
% Solving for any support vector to obtain beta0
Beta_0=mean(Ys-(Xs*Beta'))

data_test=csvread('data_test.csv');

Xtest=data_test(:,1:end-1);
Ytest=data_test(:,end);
% f~ (Predicted labels)
f=sign(Xtest*Beta'+Beta_0);
%[F_measure, Accuracy] = confusion_mat(Ytest,f)
[~, Accuracy, F_measure ] = confusionMatrix(Ytest,f )

ft=x*Beta'+Beta_0;
zeta=max(0,1-y.*ft);
Non_Zero_Zeta=sum(zeta~=0);

X=x;
Y=y;

figure
hold on
scatter(X(Y==1,1),X(Y==1,2),'b')
scatter(X(Y==-1,1),X(Y==-1,2),'r')
scatter(Xs(Ys==1,1),Xs(Ys==1,2),'.b')
scatter(Xs(Ys==-1,1),Xs(Ys==-1,2),'.r')
syms x
fn=vpa((-Beta_0-Beta(1)*x)/Beta(2),4);
fplot(fn,'Linewidth',2);
fn1=vpa((1-Beta_0-Beta(1)*x)/Beta(2),4);
fplot(fn1,'--');
fn2=vpa((-1-Beta_0-Beta(1)*x)/Beta(2),4);
fplot(fn2,'--');
axis([-3 3 -3 3])
xlabel('{x_1}')
ylabel('{x_2}')
title('SVM Decision Boundary: PD Method')
legend('+ class','- class','support vector (+)','support vector (-)','Decision
    Boundry','Location','southeast')
hold off
```

7 Appendix B: Matlab Implementation of the SMO Method

```
clear all;clc;
data=csvread('data_train2.csv');
%z-score normalization
data(:,1:end-1)=zscore(data(:,1:end-1));
x = data(:,1:end-1);
y = data(:,end);

%%Parameters

n = length(data); %Total number of data points
C = 0.5; %Regularization parameter, C (optimal) using Cross-Validation techniques
tol = 10e-5; %Tolerence
```

```
%Initializing Lagrange Multipliers alpha to the same alpha_0 as the PD method
alpha_temp = 0.1*ones(length(x)-1,1); % choose n-1 parameters arbitrarily and
    compute the last one so that the equality constraint is satisfied
alpha_end=(-alpha_temp'*y(1:length(x)-1))/y(end);
alpha=[alpha_temp; alpha_end];
bias = 0; %Initialise bias
info.xs = [];
nIter=0;
%% Setting up the SMO Algorithm for SVM optimisation
tic
while (1)

    changed_alphas=0;
    N=size(y,1);
    fprintf('Possible support vectors: %d',N)

    for i=1:N
        Inputs1=x*x(i,:)';
        Ei=sum(alpha.*y.*Inputs1)-y(i);

        if ((Ei*y(i) < -tol) && alpha(i) < C) || (Ei*y(i) > tol && (alpha(i) > 0)
    )

                for j=[1:i-1,i+1:N]
                    Inputs2=x*x(j,:)';
                    Ej=sum(alpha.*y.*Inputs2)-y(j);
                      alpha_iold=alpha(i);
                      alpha_jold=alpha(j);
                      if y(i)~=y(j)
                          L=max(0,alpha(j)-alpha(i));
                          H=min(C,C+alpha(j)-alpha(i));
                      else
                          L=max(0,alpha(i)+alpha(j)-C);
                          H=min(C,alpha(i)+alpha(j));
                      end
                      if (L==H)
                          continue
                      end
                      Dotxixj=x(j,:)*x(i,:)';
                      Dotxixi=x(i,:)*x(i,:)';
                      Dotxjxj=x(j,:)*x(j,:)';
                      eta = 2*Dotxixj-Dotxixi-Dotxjxj;

                      if eta>=0
                          continue
                      end

                      alpha(j)=alpha(j)-( y(j)*(Ei-Ej) )/eta;

                      if alpha(j) > H
                          alpha(j) = H;
                      elseif alpha(j) < L
                          alpha(j) = L;
                      end
                      if norm(alpha(j)-alpha_jold) < tol
                          continue
                      end
                      alpha(i)=alpha(i) + y(i)*y(j)*(alpha_jold-alpha(j));
                      b1 = bias - Ei - y(i)*(alpha(i)-alpha_iold)*Dotxixi...
                          -y(j)*(alpha(j)-alpha_jold)*Dotxixj;

                      b2 = bias - Ej - y(i)*(alpha(i)-alpha_iold)*Dotxixj...
                          -y(j)*(alpha(j)-alpha_jold)*Dotxjxj;

                      if 0<alpha(i)<C
                          bias=b1;
                      elseif 0<alpha(j)<C
                          bias=b2;
                      else
                          bias=(b1+b2)/2;
                      end

                      changed_alphas=changed_alphas+1;
                end
        end
    end

    if changed_alphas==0
        break
    end
    info.xs = [info.xs alpha];
    nIter=nIter+1;
end
nIter
toc

%% Retriving betas

%beta
alpha=info.xs(:,end)
Beta=sum(alpha.*y.*x)
%beta_0
Beta_0 =mean( y - x*Beta')
%Support vectors
Xs=x(alpha>0,:); Ys=y(alpha>0);
fprintf('Number of support Vectors : %d',size(Xs,1))

%Accuracy and F-measure
data_test=csvread('data_test.csv');
Xtest=data_test(:,1:end-1);
Ytest=data_test(:,end);
% f~ (Predicted labels)
f=sign(Xtest*Beta'+Beta_0);
%[F_measure, Accuracy] = confusion_mat(Ytest,f)
[~, Accuracy, F_measure ] = confusionMatrix(Ytest,f )

X=x; Y=y;
figure
hold on
scatter(X(Y==1,1),X(Y==1,2),'b')
scatter(X(Y==-1,1),X(Y==-1,2),'r')
scatter(Xs(Ys==1,1),Xs(Ys==1,2),'.b')
scatter(Xs(Ys==-1,1),Xs(Ys==-1,2),'.r')
syms x
fn=vpa((-Beta_0-Beta(1)*x)/Beta(2),4);
```

```
fplot(fn,'Linewidth',2);
fn1=vpa((1-Beta_0-Beta(1)*x)/Beta(2),4);
fplot(fn1,'--');
fn2=vpa((-1-Beta_0-Beta(1)*x)/Beta(2),4);
fplot(fn2,'--');
axis([-3 3 -3 3])
xlabel('{x_1}')
ylabel('{x_2}')
title('SVM Decision Boundary: SMO Method')
legend('+ class','- class','support vector (+)','support vector (-)','Decision
    Boundry','Location','southeast')
hold off


%% Objective function
info.xs
xMin=info.xs(:,end);
x = data(:,1:end-1);
y = data(:,end);
D = zeros(length(x));
for i = 1: length(x)
    for j = 1:length(x)
        D(i,j) = y(i)*y(j)*x(i,:)*x(j,:)';
    end
end
% Objective function
F.f = @(x) 1/2*x'*D*x - x'*ones(length(x),1);
F.df = @(x) D*x - ones(length(x),1);
F.d2f = @(x) D;
fMin = F.f(xMin);

eig(D); %to verify if assumption of positive-definiteness is met for linear
    convergence
info.xs;
iterations=1:nIter;
Fval=[];
for i=1:nIter
    input=info.xs(:,i);
    Fval=[Fval,F.f(input)];
end
figure;
hold on
plot(iterations,Fval,'b')
xlabel('Number of iterations')
ylabel('Value of the objective function')
hold off


%% Convergence plots (Same script as above)
```

References

[1] J.Taylor and Sh.Sun. "A review of optimization methodologies in support vector machines", p.8. 2011.

[2] F.Potra and S. Wright. "Interior-point methods". Journal of Computational and Applied Mathematics Volume 124, Issues 1–2, 1 December 2000, Pages 281-302.

[3] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. "The Elements of Statistical Learning,Second Edition".

[4] Marta Betcke. "Numerical Optimisation Course: Constraint Optimisation Lecture". 2021.

[5] W. Murray and M. H. Wright. "Practical Optimization". 1981.

[6] J.Platt. "Fast Training of Support Vector Machines using Sequential Minimal Optimization". MIT Press (1998).

[7] S.Bellavia. "Inexact Interior-Point Method". Journal of Optimization Theory and Applications volume 96, pages109–121 (1998).

[8] S.Boyd and L.Vandenberghe. "Convex Optimization". Chapter 11.7, p.607.

[9] S.Boyd and L.Vandenberghe. "Convex Optimization". Chapter 10.3.3.

[10] S.Boyd and L.Vandenberghe. "Convex Optimization". Chapter 11.7.3.

[11] S.Boyd and L.Vandenberghe. "Convex Optimization". Chapter 11.7.3.

[12] J. Gondzio. "Interior Point Methods 25 Years Later". 2011.

[13] R. Freund E. Osuna and F. Girosi. " An improved training algorithm for support vector machines". 1997.

[14] Stanford University. "The Simplified SMO Algorithm". (2009).

[15] J. López and J. R. Dorronsoro. "The convergence rate of linearly separable SMO". The 2013 International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1-7.

[16] Chih-Jen Lin. "Linear Convergence of a Decomposition Method for Support Vector Machines".

[17] Chih-Jen Lin. "Asymptotic Convergence of an SMO Algorithm Without any Assumptions".