Numerical Optimisation Project
Binary Classification Task on Spiral Data based on SVM

Bowen Ding(21118015)

April 1, 2022

Abstract

In this Project, we particularly explore a binary classification task on Spiral Data(a non-linear separable data set) based on Support Vector Machine(SVM). Under this case, we introduce kernel method and convert the primal SVM format into dual format to improve the computation efficiency. In the process of solving the optimal parameters, we utilise Sequential Minimal Optimisation(SMO) algorithm with two different working set selection tactics: SMO-Simplified and SVM-Light , afterwards we benchmark and compare the performance and convergence of them.

Contents

1 Introduction

In this project, all of our following work is based on spiral data set. The spiral data set contains m two-dimensional samples and all of them are tagged with binary label. Label y_i of a specified sample $\boldsymbol{x}_i = [x_{i1}, x_{i2}]^T$ is tagged as '+1' if $\begin{cases} x_{i1} = r_i \sin 2r_i + \varepsilon \\ x_{i2} = r_i \cos 2r_i + \varepsilon \end{cases}$ while is '-1' if $\begin{cases} x_{i1} = r_i \sin\left(2r_i + \pi\right) + \varepsilon \\ x_{i2} = r_i \cos\left(2r_i + \pi\right) + \varepsilon \end{cases}$, where $r_i = 1 + (i-1)\frac{2\pi}{m-1}$, $i = 1, ..., m$ and ε is noise term submitted to $\mathcal{N}(0, 0.2)$.

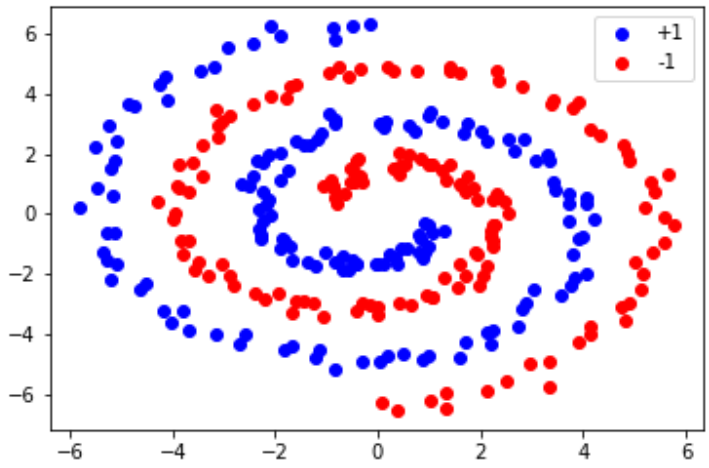

Figure 1: Visualisation of Spiral Data Set

In order to benchmark the binary classification performance of SVM, we separate the generated data into training set (300 samples) and test set (300 samples) uniformly. However, original primal SVM format implementation on this data set does not work due to some challenges [**?**]:

- It is obvious that this data set is not linear separable, which means we are unlikely to find a hyper-plane to classify '+1' and '-1' accurately. However, kernel trick helps solve this problem subtly by mapping the original sample points into a linear separable high dimensional space. So, we will implement kernel SVM on this problem.
- High or infinite dimension of mapped samples leads to the learnable parameter $\boldsymbol{w}$ being high dimensional or infinite for primal format SVM, making the training process quite low efficiency or sometimes infeasible(for infinite dimension situation). Therefore, dual format SVM is utilised to guarantee the learnable parameter into finite dimensional(the size of train set).
- The dual kernel SVM essentially abstracts the original problem into a large dense quadratic programming problem with constraint qualification. In other words, the kernel matrix is an $m \times m$ fully dense matrix, where m is the size of training set. When m goes quite large, it will cost quite large computer memory to store the kernel matrix, let alone calculating Hessian matrix of the kernel matrix. Hence, Newton and Quasi Newton method cannot be applied in the optimisation stage of SVM straight-forwardly.

We organise the report as below. In section 2, we will give a brief introduction to SVM format used on this problem, as well as theoretical fundamentals of optimisation methods applied in SVM. Section 3 presents the comparison between two working set selection in terms of applicability, convergence, CPU time and memory efficiency based on numerical experiments. Section 4 gives a summary of our experiment result.

2 Theory

2.1 Support Vector Machine

2.1.1 Hard-margin SVM

Given training set $D = \{(\boldsymbol{x_k}, y_k)\}_{k=1}^{m}, y_k \in \{1, -1\}$. We want to have a hyper-plane $(\boldsymbol{w}, b) : \boldsymbol{w}^T\boldsymbol{x} + b = 0$ to separate all data accurately, where $\boldsymbol{w}$ is normal vector of the hyper-plane and b is bias determining the intercept. Our aim is to find an optimal hyper-plane to separate all data correctly with the maximum margin. The margin refers to the minimal distance from samples in the data set to the hyper-plane, which is denoted as $r = \frac{|\boldsymbol{w}^T\boldsymbol{x}+b|}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$ and the data points lying on the margin are called support vectors. Therefore, the binary task can be described as a quadratic programming problem(primal hard-margin SVM format) [3]:

$$\min_{\boldsymbol{w},b} \frac{\boldsymbol{w}^T\boldsymbol{w}}{2}$$

$$s.t. \qquad y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1, i = 1, .., m$$

Note: $\max\limits_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|} \Leftrightarrow \min\limits_{\boldsymbol{w},b} \frac{\boldsymbol{w}^T\boldsymbol{w}}{2}$

2.1.2 Soft-margin SVM

Basically, primal hard-margin SVM aims to make sure the selected hyper-plane to classify all the data correctly, which required data space to be absolutely linearly separable. But for most data set, linearly separable condition is hard to satisfy. For such cases, we allow training errors occur. So we introduce slack variables $\xi_i \geq 0$ to weaken the constraints. Therefore, the soft-margin SVM can be shown as below:

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & i = 1, ..., m \end{aligned}$$

The slack variable ξ_i measure in what extent the sample not observe the primal constraints.For example, $\boldsymbol{x}_i$ will not be on the accurate side of the hyper-plane when $\xi_i \geq 1$. C refers to a large penalty parameter, making sure the optimal ξ_i tends to be 0 after several iterations of numerical optimisation.

2.1.3 Kernel SVM

The introduction of soft margin can tolerate training error to some extent, but it still utilises a linear boundary to divide data points in original data space. But for the data with a distribution like the spiral data set where it is impossible to use a linear boundary to separate the data points into correct side. So, we have to introduce the kernel trick to map the data into a higher linearly dimensional data space where we can find a proper hyper-plane. Here, we define a mapping function $\boldsymbol{\Phi}(\boldsymbol{x}) : R^{\boldsymbol{n}} \to R^{\boldsymbol{d}}$,$d > n$ and d can be taken as ∞. Hence, the primal soft-margin SVM can be shown as:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{w}},b,\boldsymbol{\xi}} \ \frac{1}{2}\tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{w}} + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y_i\left(\tilde{\boldsymbol{w}}^T\boldsymbol{\Phi}(\boldsymbol{x_i}) + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & i = 1, ..., m \end{aligned}$$

where $\tilde{\boldsymbol{w}} \in R^{\boldsymbol{d}}$ and $\boldsymbol{\Phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), ..., \phi_d(\boldsymbol{x})]^T$. However, when dimension of $\boldsymbol{\Phi}(\boldsymbol{x})$ is very high even infinite, making quite hard to get optimal $\tilde{\boldsymbol{w}}$ because of dimensional curse. So we need another method to guarantee our learnable parameter's dimension is not quite high. In order to solve the problem above, we define the Lagrangian:

$$L(\tilde{\boldsymbol{w}}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda}) = \frac{1}{2}\tilde{\boldsymbol{w}}^{\boldsymbol{T}}\tilde{\boldsymbol{w}} + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(\tilde{\boldsymbol{w}}^T\boldsymbol{x}_i + b\right) - \xi_i\right) + \sum_{i=1}^{n}\lambda_i(-\xi_i)$$

And we can get KKT conditions for primal soft-margin:

$$\begin{cases} \left.\begin{aligned} y_i\left(\tilde{\boldsymbol{w}}^T\boldsymbol{\Phi}(\boldsymbol{x}_i) + b\right) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}\right\} \textit{primal feasible condition} \\ \left.\begin{aligned} \alpha_i &\geq 0 \\ \lambda_i &\geq 0 \end{aligned}\right\} \textit{dual feasible condition} \\ \left.\begin{aligned} \alpha_i\left(1 - y_i\left(\tilde{\boldsymbol{w}}^T\boldsymbol{x}_i + b\right) - \xi_i\right) &= 0 \\ \lambda_i\xi_i &= 0 \end{aligned}\right\} \textit{complementary slackness} \\ \left.\begin{aligned} \tfrac{\partial L}{\partial \tilde{\boldsymbol{w}}}{}_{d\times 1} = \tilde{\boldsymbol{w}} - \textstyle\sum_{i=1}^{m}\alpha_i y_i\boldsymbol{\Phi}(\boldsymbol{x}_i) &= 0 \\ \tfrac{\partial L}{\partial b} = \textstyle\sum_{i=1}^{m}\alpha_i y_i &= 0 \\ \tfrac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i &= 0 \end{aligned}\right\} dual - inner\ optimal\ condition \end{cases}$$

$$i = 1, ..., m$$

The KKT conditions are sufficient and necessary for this optimisation problem because primal soft-margin SVM is a quadratic programming problem essentially. Also, we know that the strong duality of the optimality holds when original problem is convex with feasible solutions,whose constraints are linear. Consequently, the dual format soft-margin SVM is deduced by strong duality as below

$$\min_{\tilde{\boldsymbol{w}},b,\boldsymbol{\xi}}\left(\max_{\boldsymbol{\alpha},\boldsymbol{\lambda}\geq 0} L(\tilde{\boldsymbol{w}}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda})\right) = \max_{\boldsymbol{\alpha},\boldsymbol{\lambda}\geq 0}\left(\min_{\tilde{\boldsymbol{w}},b,\boldsymbol{\xi}} L(\tilde{\boldsymbol{w}}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda})\right)$$

Apply dual-inner optimal condition into above format which is equivalent to primal soft-margin SVM, then we get

$$\max_{\boldsymbol{\alpha},\boldsymbol{\lambda}\geq 0}\left(\min_{\tilde{\boldsymbol{w}},b,\boldsymbol{\xi}} L(\tilde{\boldsymbol{w}}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda})\right) = \max_{\boldsymbol{\alpha},\boldsymbol{\lambda}\geq 0}\left(L(\tilde{\boldsymbol{w}}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda})|_{\tilde{\boldsymbol{w}}=\sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\alpha_i y_i\Phi(\boldsymbol{x}_i), \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}}\alpha_i y_i = 0, C=\alpha_i+\lambda_i}\right)$$

which is equivalent to the optimality:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{i=1}^{\boldsymbol{m}}\sum_{j=1}^{\boldsymbol{m}}\alpha_i y_i \boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)^T \boldsymbol{\Phi}\left(\boldsymbol{x}_j\right)\alpha_j y_j - \sum_{i=1}^{\boldsymbol{m}}\alpha_i$$

$$\begin{aligned} s.t. \qquad \alpha_i &\leq 0 \\ \alpha_i - C &\leq 0 \\ \sum_{i=1}^{\boldsymbol{m}} \alpha_i y_i &= 0 \\ i &= 1, ..., m \end{aligned}$$

Finally, we get the dual soft-margin format SVM. At the same time, we can calculate the kernel matrix $[K_{ij}]_{m\times m} = \left[\boldsymbol{\Phi}\left(\boldsymbol{x}_i\right)^T \boldsymbol{\Phi}\left(\boldsymbol{x}_j\right)\right]_{m\times m}$ previously to get rid of repeated computing on inner product of two high dimensional vectors. Hence, the kernel SVM has been deduced successfully, and can be shown as:

$$\min_{\boldsymbol{\alpha}} f\left(\boldsymbol{\alpha}\right) = \frac{1}{2}\sum_{i=1}^{\boldsymbol{m}}\sum_{j=1}^{\boldsymbol{m}}\alpha_i y_i K_{ij} \alpha_j y_j - \sum_{i=1}^{\boldsymbol{m}}\alpha_i$$

$$\begin{aligned} s.t. \qquad \alpha_i &\leq 0 \\ \alpha_i - C &\leq 0 \\ \sum_{i=1}^{\boldsymbol{m}} \alpha_i y_i &= 0 \\ i &= 1, ..., m \end{aligned}$$

Similarly, we can also construct Lagrangian for kernel format above:

$$L\left(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\mu}, b\right) = \frac{1}{2}\sum_{i=1}^{\boldsymbol{m}}\sum_{j=1}^{\boldsymbol{m}}\alpha_i y_i K_{ij} \alpha_j y_j - \sum_{i=1}^{\boldsymbol{m}}\alpha_i - \sum_{i=1}^{\boldsymbol{m}}\lambda_i\alpha_i + \sum_{i=1}^{\boldsymbol{m}}\mu_i\left(\alpha_i - C\right) + b\sum_{i=1}^{\boldsymbol{m}} y_i\alpha_i$$

Hence, we can get the corresponding KKT conditions:

$$\begin{cases} 0 \leq \alpha_i \leq C, i = 1...m \\ \sum_{i=1}^{m} y_i\alpha_i = 0 \\ \left.\begin{array}{r} \lambda_i \geq 0, i = 1...m \\ \mu_i \geq 0, i = 1...m \end{array}\right\} dual\ feasible\ condition \\ \left.\begin{array}{r} \lambda_i\alpha_i = 0, i = 1...m \\ \mu_i\left(C - \alpha_i\right) = 0, i = 1...m \end{array}\right\} complementary\ slackness \\ \frac{\partial L}{\partial \alpha_i} = \nabla f\left(\boldsymbol{\alpha}\right)_i - \lambda_i - \mu_i + by_i = 0, i = 1...m \end{cases}$$

Actually, KKT conditions can serve as stopping condition when solving this problem because they are sufficient and necessary conditions for optimal point in this problem, which will be discussed more afterwards.

2.2 Numerical Optimisation Method for SVM

Kernel SVM optimality problem can be represented as a quadratic optimization problem with linearly and bound constraints:

$$\min_{\boldsymbol{\alpha}} f\left(\boldsymbol{\alpha}\right) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{e}$$

$$\begin{aligned} s.t. \quad \boldsymbol{y}^T\boldsymbol{\alpha} &= 0 \\ 0 < \alpha_i &< C, i = 1...m \end{aligned}$$

where Hessian matrix $[Q_{ij}]_{m\times m} = [y_i K_{ij} y_j]_{m\times m}$, which is a positive semi-definite dense matrix; $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_m]^T$ and each component of it is a Lagrangian multiplier for a primal feasible condition with a range from 0 to C(penalty parameter for slack variables). We can find that the dimension of Hessian matrix is highly depended on the data number m of the data set. So some traditional optimisation algorithms like Newton or Quasi Newton methods are not suitable here attributed to heavy intensive computation based on Q and overload memory to store them[A two level]. Here we consider a decomposition problem—Sequential Minimal Optimisation(SMO) to solve this optimality problem.

2.2.1 Sequential Minimal Optimisation Procedure

SMO algorithm is a decompisition method. Instead of solving a quadratic programming problem depended on $m \times m$ fully dense matrix Q and optimise m dimension . In each iteration, SMO only focus on optimising two parameters α_i, α_j, where $i, j \in B, B \subseteq \{1, 2, ..., m\}, |B| = 2$. B is called as working set and keeps updating with iteration until no suitable elements can be selected. The remaining of $\{1, 2, ..., m\}$ is denoted as $N = \{1, 2, ..., m\} \backslash B$. Hence, the original densely quadratic programming problem can be reduced as a sub-problem in each iteration:

$$\min_{\alpha_i, \alpha_j} W(\alpha_i, \alpha_j) = \frac{1}{2} Q_{ii} {\alpha_i}^2 + \frac{1}{2} Q_{jj} {\alpha_j}^2 + Q_{ij} \alpha_i \alpha_j - (\alpha_i + \alpha_j) + \left(\boldsymbol{\alpha}_N^k\right)^T Q_{NB} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix}$$

$$s.t \quad 0 < \alpha_i < C$$
$$0 < \alpha_j < C$$
$$y_i \alpha_i + y_j \alpha_j = -\boldsymbol{y}_N^T \boldsymbol{\alpha}_N^k$$

where α_i and α_j are selected based on the elements of working set B , Q_{ij} is the (i, j) element in Q matrix while Q_{NB} is the sub-matrix of Q based on the component in set N and set B. $\boldsymbol{\alpha}_N^k$ is the corresponding parameter vector of set N in the last iteration and it serves as a constant vector in current optimality problem. Consequently, the sub-problem is a constrained quadratic programming problem with only two parameter and it can be analytically solved[Platt, 1998], greatly saving the memory and time for computation in each iteration. The algorithm process of Generalised SMO algorithm is as below:

Generalised SMO algorithm

Input: training set $D = \{(\boldsymbol{x}_p, y_p)\}_{p=1}^m$, $\boldsymbol{x}_p \in R^n, y_p \in \{-1, 1\}$
Initialization: $\boldsymbol{\alpha}^0 = \left[\alpha_1^0, \alpha_2^0, ..., \alpha_m^0\right]^T = [0, 0, ..., 0]^T$
Repeat:
Select working set: Select working set $\left\{i^k, j^k\right\} \subseteq \{1, 2, ..., m\}$,
and the corresponding $\alpha_{i^k}^{k-1}, \alpha_{j^k}^{k-1}$ to be updated.
Update: By solving sub-problem analytically, we can get the optimal $\boldsymbol{\alpha}^k$, where
$\alpha_{i^k}^k = update\left(\alpha_{i^k}^{k-1}\right), \alpha_{j^k}^k = update\left(\alpha_{j^k}^{k-1}\right)$
$\alpha_p^k = \alpha_p^{k-1}$when $p \in \{1, 2, ..., m\} / \left\{i^k, j^k\right\}$
Until: Stopping conditions(KKT conditions) are satisfied with tolerance ε
or when $\boldsymbol{\alpha}'s$ change is quite small.

In fact, selection of working set highly affects the convergence of the algorithm, so we mainly explore and compare two selecting strategies shown below.

2.2.2 Working Set Selection

1. SMO-Simple Algorithm
SMO-Simple algorithm is dedicated to employ a simpler heuristics strategy to choose element α_i and α_j by iterating over all α_i. At the same time, we select the corresponding α_j among the $m - 1$ elements in $\boldsymbol{\alpha}$ except α_i. We stop iteration until current α_i satisfies the KKT condition with a ε tolerance. However, this algorithom cannot guarantee the global optimal becuase we do not iterate over all the pair of α_i and α_j so that some more optimised working set is missed [2].

2. SVM-Light Algorithm Actually, SVM-Light [1] is a popular decomposition methods which can select working set more systematically. When the size of working set equals to two, it is equivalent to a SMO modification format proposed by Keerthi et al. [4]. Working set of this algorithm is selected by KKT violation. The KKT conditions for Kernel format SVM can be rewritten as:

$$\begin{cases} \nabla f(\boldsymbol{\alpha})_i + by_i \geq 0, if \ \alpha_i < C \\ \nabla f(\boldsymbol{\alpha})_i + by_i \leq 0, if \ \alpha_i > 0 \end{cases} \quad (KKT\ 1)$$

According to $y_i = \pm 1$, we can rewrite the $(KKT\ 1)$ above as:

$$\begin{cases} \nabla f(\boldsymbol{\alpha})_i + b \geq 0, if \ \alpha_i < C, y_i = 1 \\ \nabla f(\boldsymbol{\alpha})_i - b \geq 0, if \ \alpha_i < C, y_i = -1 \\ \nabla f(\boldsymbol{\alpha})_i + b \leq 0, if \ \alpha_i > 0, y_i = 1 \\ \nabla f(\boldsymbol{\alpha})_i - b \leq 0, if \ \alpha_i > 0, y_i = -1 \end{cases} \quad (KKT\ 2)$$

From $(KKT\ 2)$, we can easily deduce the range for b:

$$\begin{cases} \max\limits_{i \in I_{up}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i \leq b \leq \min\limits_{j \in I_{low}(\boldsymbol{\alpha})} -y_j \nabla f(\boldsymbol{\alpha})_j \\ I_{up}(\boldsymbol{\alpha}) \equiv \{i | \alpha_i < C, y_i = 1 \ \ or \ \alpha_i > 0, y_i = -1\} \\ I_{low}(\boldsymbol{\alpha}) \equiv \{i | \alpha_i < C, y_i = -1 \ \ or \ \alpha_i > 0, y_i = 1\} \end{cases}$$

Therefore, we concludes that $\boldsymbol{\alpha}$ is optimal if and only if

$$\max_{i\in I_{up}(\boldsymbol{\alpha})} -y_i\nabla f(\boldsymbol{\alpha})_i \le \min_{j\in I_{low}(\boldsymbol{\alpha})} -y_j\nabla f(\boldsymbol{\alpha})_j$$

Consequently, $\boldsymbol{\alpha}$ is not optimal when $\exists i\in I_{up}(\boldsymbol{\alpha}), j\in I_{low}(\boldsymbol{\alpha}), -y_i\nabla f(\boldsymbol{\alpha})_i > -y_j\nabla f(\boldsymbol{\alpha})_j$. Such (i,j) is called as a "violating pair" [4]. Hence, $\boldsymbol{\alpha}$ is optimal when no "violating pairs" are found. To guarantee the asymptotic convergence, we select the working set containing the maximal violating pair in each iteration. i.e the indices of two elements in the working set should be $i\in arg\max_{t\in I_{up}(\boldsymbol{\alpha})} -y_t\nabla f(\boldsymbol{\alpha})_t$ and $j\in arg\min_{t\in I_{low}(\boldsymbol{\alpha})} -y_t\nabla f(\boldsymbol{\alpha})_t$. And here we will give a brief interpretation about why the maximal violating pair can ensure the global asymptotic convergence.

Theorem (Asymptotic Convergence of SVM-Light Algorithm) If $\{\boldsymbol{\alpha}_k\}$ is the sequence generated by the decomposition method (like SVM-Light based SMO algorithm), any convergent subsequence converges to the optimal point $\boldsymbol{\alpha}^*$ of kernel SVM format optimality [5].

Proof [5] Solving the sub-problem quadratic programming problem for kernel SVM based on SVM-Light essentially means to find the minimisation in a linear equality constraint $\alpha_i y_i + \alpha_j y_j = -\boldsymbol{y_N}\boldsymbol{\alpha_N^k} := \zeta$ within a rectangle area due to $0\le\alpha_i,\alpha_j\le C$. We use $\boldsymbol{\alpha}(t) = [\alpha_1(t),\alpha_2(t),...,\alpha_m(t)]^T$ to denote the change of $\boldsymbol{\alpha}$ in the linear equality constraint where

$$\alpha_h(t) = \begin{cases} \alpha_h^k + \frac{t}{y_h}, h = i \\ \alpha_h^k - \frac{t}{y_h}, h = j \\ \alpha_h^k, Otherwise \end{cases}$$

Hence, the previous sub-problem with parameter α_i and α_j can be converted to a optimality problem with only parameter t:

$$\begin{aligned} \min_t\ & \varphi(t) := f(\boldsymbol{\alpha}(t)) \\ s.t.\quad & 0\le\alpha_i(t)\le C \\ & 0\le\alpha_j(t)\le C \end{aligned}$$

We assume $\bar{t}$ is the optimal point of the problem and $\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}(\bar{t})$, $\bar{t} = \frac{\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|}{\sqrt{2}}$. We use the second-order Taylor expansion to approximate the $\varphi(t)$: $\varphi(t) = \varphi(0) + \varphi'(0)t + \varphi''(0)\frac{t^2}{2}$.

Because

$$\begin{aligned} \varphi'(t) &= \sum_{h=1}^{m} \nabla f(\boldsymbol{\alpha}(t))_h \alpha_h'(t) \\ &= y_i\nabla f(\boldsymbol{\alpha}(t))_i - y_j\nabla f(\boldsymbol{\alpha}(t))_j \\ &= y_i\left(\sum_{h=1}^{m} Q_{ih}\alpha_h(t) - 1\right) - y_j\left(\sum_{h=1}^{m} Q_{jh}\alpha_h(t) - 1\right) \\ \varphi''(t) &= Q_{ii} + Q_{jj} - 2y_iy_jQ_{ij} \end{aligned}$$

Therefore, when $t = 0$,

$$\varphi'(0) = y_i\nabla f\left(\boldsymbol{\alpha}^k\right)_i - y_j\nabla f\left(\boldsymbol{\alpha}^k\right)_j$$

$$\begin{aligned} \varphi''(0) &= K_{ii} + K_{jj} - 2K_{ij} \\ &= \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_i) + \Phi(\boldsymbol{x}_j)^T\Phi(\boldsymbol{x}_j) - 2\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j) \\ &= \left\|\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\right\|^2 \end{aligned}$$

So, considering Q is S.P.D and $\varphi''(0)\ge 0$, we talk about the following cases.

Case 1 $\varphi''(0) = 0$, i.e $\Phi(\boldsymbol{x}_i) = \Phi(\boldsymbol{x}_j)$, hence

$$\begin{aligned} \varphi'(0) &= y_i\sum_{h=1}^{m} Q_{ih}\alpha_h^k - y_j\sum_{h=1}^{m} Q_{jh}\alpha_h^k \\ &= y_i\left(\sum_{h=1}^{m} y_iy_h\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_h)\alpha_h^k - 1\right) - y_j\left(\sum_{h=1}^{m} y_jy_h\Phi(\boldsymbol{x}_j)^T\Phi(\boldsymbol{x}_h)\alpha_h^k - 1\right) \\ &= y_j - y_i \end{aligned}$$

Potentially, the descent is guaranteed when

$$\varphi'(0) = y_i\nabla f\left(\boldsymbol{\alpha}^k\right)_i - y_j\nabla f\left(\boldsymbol{\alpha}^k\right)_j \ne 0$$

In this circumstance, $y_i\ne y_j$ and $\left|\varphi'(0)\right| = 2$. According to $\varphi''(0) = \varphi''(t)|_{t=0} = 0$, it is shown that $\varphi'(t)$ is a linear function, so $\varphi(\bar{t})\le\varphi(0)$ with $|\bar{t}|\le C$. Then we get

$$\varphi(\bar{t}) - \varphi(0) = -\left|\varphi'(0)\bar{t}\right| \le -\frac{2}{C}\bar{t}^2 = -\frac{\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|^2}{C}\ (*1)$$

Case 2 $\varphi^{''}(0) > 0$. Assume $\bar{t} = \gamma t^*$ where $0 < \gamma < 1$ and $t^* = -\frac{\varphi^{'}(0)}{\varphi^{''}(0)}$ is the unconstrained minimum of φ. So we can get that

$$\varphi(\bar{t}) - \varphi(0) = \gamma \frac{-\varphi^{'}(0)^2}{\varphi^{''}(0)} + \frac{\gamma^2}{2}\frac{\varphi^{'}(0)^2}{\varphi^{''}(0)} \leq -\frac{\gamma^2}{2}\frac{\varphi^{'}(0)^2}{\varphi^{''}(0)} = -\frac{\varphi^{''}(0)}{2}\bar{t}^2 = -\frac{\varphi^{''}(0)}{4}\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\right\|^2 C \ (*2)$$

In summary, note that $\varphi(0) = f\left(\boldsymbol{\alpha}^k\right)$, $\varphi(\bar{t}) = f\left(\boldsymbol{\alpha}^{k+1}\right)$ and with use of $\bar{t} = \frac{\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\right\|}{\sqrt{2}}$,$\varphi^{''}(t) = Q_{ii} + Q_{jj} - 2y_i y_j Q_{ij}$ and $(*1)$, we can get $\sigma \equiv \min\left\{\frac{2}{C}, \min_{i,j}\left\{\frac{Q_{ii}+Q_{jj}-2y_i y_j Q_{ij}}{2} : Q_{ii} + Q_{jj} - 2y_i y_j Q_{ij} > 0\right\}\right\}$ to make sure $f\left(\boldsymbol{\alpha}^{k+1}\right) - f\left(\boldsymbol{\alpha}^k\right) \leq -\frac{\sigma}{2}\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\right\|^2$. Hence, the asymptotic convergence of SVM-Light is proved.

Q.E.D

2.2.3 Updating Parameters

After selecting the proper working set $B = \{i, j\}$, we solve the sub-problem mentioned in section **1.2.1** to update $\boldsymbol{\alpha}$ in each iteration, which means the aim is to find the required minimisation taking place in the rectangle area $S = [0, C] \times [0, C]$ along a line $\alpha_i y_i + \alpha_j y_j = -\boldsymbol{y_N}\boldsymbol{\alpha_N^k} := \zeta$, where ζ is a constant in current iteration $k+1$ and $y_i, y_j = \pm 1$. Hence, α_i can be represented as $\alpha_i = (\zeta - \alpha_j y_j) y_i$ and the sub-problem can be re-written as:

$$\min_{\alpha_i} \psi(\alpha_i) = \frac{1}{2}Q_{ii}{\alpha_i}^2 + \frac{1}{2}Q_{jj}(\zeta - \alpha_i y_i)^2 + y_j Q_{ij}\alpha_i(\zeta - \alpha_i y_i) - (\zeta - \alpha_i y_i) y_j - \alpha_i + \left(\boldsymbol{\alpha}_N^k\right)^T Q_{NB}\begin{bmatrix} \alpha_i \\ (\zeta - \alpha_i y_i) y_j \end{bmatrix}$$

$$s.t. \quad L \leq \alpha_i \leq H$$

$$\begin{cases} L = \max\left(0, \alpha_i^k - \alpha_j^k\right); H = \min\left(C, C + \alpha_i^k - \alpha_j^k\right), y_i \neq y_j \\ L = \max\left(0, \alpha_i^k + \alpha_j^k - C\right); H = \min\left(C, \alpha_i^k + \alpha_j^k\right), y_i = y_j \end{cases}$$

By setting $\frac{\partial \psi}{\partial \alpha_i} = 0$, we can deduce that

$$\alpha_i^{k+1,unc} = \alpha_i^k + \frac{y_i(E_i - E_j)}{\eta}$$

where $E_t = \left(\sum_{j=1}^N \alpha_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}_t) + b\right) - y_t = \left(\frac{[Q\boldsymbol{\alpha}]_t}{y_t} + b\right) - y_t$, $t = 1, 2$ implying the distance between prediction and truth, and $\eta = K_{ii} + K_{jj} - 2K_{ij} = \frac{Q_{ii}}{y_i^2} + \frac{Q_{jj}}{y_j^2} - 2\frac{Q_{ij}}{y_i y_j}$. Then, considering the constraint $L \leq \alpha_i \leq H$, we clip the result $\alpha_i^{k+1,unc}$:

$$\alpha_i^{k+1} = \begin{cases} H \ , \alpha_i^{k+1,unc} > H \\ \alpha_i^{k+1,unc} \ , L \leq \alpha_i^{k+1,unc} \leq H \\ L \ , \alpha_i^{k+1,unc} < L \end{cases}$$

Then, $\alpha_j^{k+1} = \alpha_j^k + y_i y_j\left(\alpha_i^k - \alpha_i^{k+1}\right)$ can be updated. At last, we upgrade b :

$$b^{k+1} = \begin{cases} b_i^{k+1} & , 0 < \alpha_i, \alpha_j < C \\ \frac{b_i^{k+1} + b_j^{k+1}}{2} & , otherwise \end{cases}$$

where

$$b_i^{k+1} = -E_i - y_i Q_{ii}\left(\alpha_i^{k+1} - \alpha_i^k\right) - y_j\frac{Q_{ji}}{y_j y_i}\left(\alpha_j^{k+1} - \alpha_j^k\right) + b^k$$

$$b_j^{k+1} = -E_j - y_i\frac{Q_{ij}}{y_i y_j}\left(\alpha_i^{k+1} - \alpha_i^k\right) - y_j Q_{jj}\left(\alpha_i^{k+1} - \alpha_i^k\right) + b^k$$

Therefore, the updating process is finished.The algorithm keeps iterating until KKT conditions are satisfied with tolerance ε.

3 Experiment and Analysis

In this section, we implement proper SVM format based on SMO algorithm with two working set selection tactics and then test on spiral data set. We select one of the strategy with a better performance in convergence and then train the algorithm with different regularisation parameter C and kernel parameter μ to benchmark convergence performance influenced by parameters. Also, we talk about the performance of the selected method in terms of complexity, CPU time and memory used. At last, we use accuracy as the metric to verify the validity of the implemented SVM.

In this part, we generate 600 sample points in total and split half of the spiral data set as training set (size 300) and set left samples as test set(size 300). It is obvious that the spiral data set is a non-linearly separable data set and we decide to use the kernel format SVM with RBF kernel for this task. RBF kernel $K_\mu(\boldsymbol{p}, \boldsymbol{q}) = e^{-\mu\|\boldsymbol{p}-\boldsymbol{q}\|^2}$ possesses the great advantage for mapping the data samples into the infinity dimensional space, showing that RBF kernel is with a strong and flexible mapping ability than other kernels.

3.1 Analysis on Working Set Selection Strategy

We compare the convergence situation when we utilise different working set selection strategies: SMO-Simple algorithm and SVM-Light algorithm. In this experiment, we set the regularisation parameter C as 0.5 , kernel parameter μ as 10, tolerance ε as 0.001 and maximum iteration number as 1000. Then we plot the convergence plot for $\boldsymbol{\alpha}$ based on the posterior $\boldsymbol{\alpha}^*$.

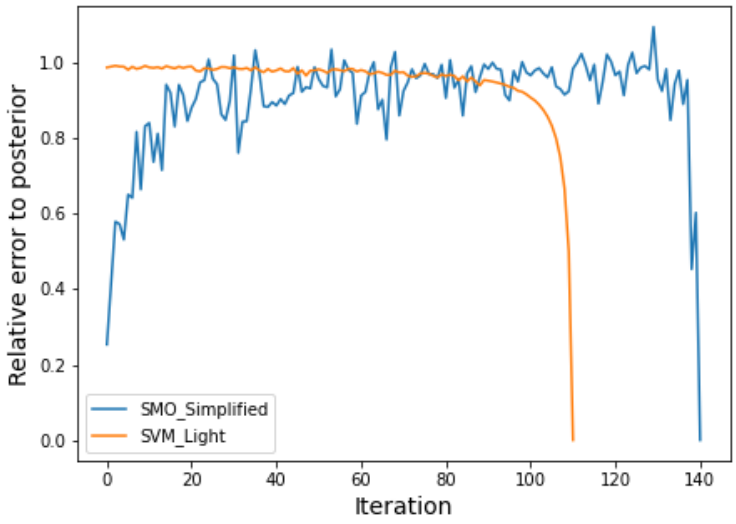

Figure 2: Convergence plot based on different working set selection strategies

The relative error to $\boldsymbol{\alpha}^*$ is measured based on Q matrix, which is $\frac{(\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^*)^T Q(\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^*)}{(\boldsymbol{\alpha}^{k}-\boldsymbol{\alpha}^*)^T Q(\boldsymbol{\alpha}^{k}-\boldsymbol{\alpha}^*)}$. From the plot above, we can easily find that SVM-Light algorithm possesses a faster convergence, which converges after 112 iterations compared with converging after 143 iterations of SMO-Simplified algorithm. Also, the converging process of SMO-Simplified highly fluctuates because of the randomness with choosing j element in working set in every iteration, which is avoided in SVM-Light algorithm. Therefore, SVM-Light's working set selection strategy is more robust so the experiments afterwards are mainly based on SMO-Light algorithm.

3.2 Analysis on Local Convergence Property

We have known that for SVM-Light algorithm, the asymptotic convergence can be assured. Now we talk about the local convergence rate of the algorithm. In this part, we still use the parameters set in section **3.1**.

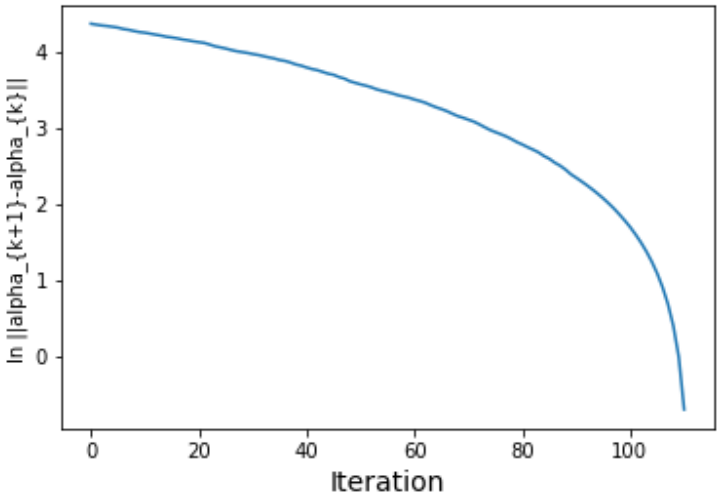

Figure 3: Exponential Convergence plot for SMO-Light

From Figure 2, we find that $\frac{\left(\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^*\right)^T Q\left(\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^*\right)}{\left(\boldsymbol{\alpha}^{k}-\boldsymbol{\alpha}^*\right)^T Q\left(\boldsymbol{\alpha}^{k}-\boldsymbol{\alpha}^*\right)} < 1$ as $k \to \infty$ and from Figure 3, we find that when $k \to \infty$, the exponential convergence plot approximates to a straight line. Hence, we empirically estimate this algorithm possesses linear convergence. Theoretically, the linear convergence can be guaranteed based on two assumptions [4]:

Assumption 1: The Kernel Matrix K is positive definite.

This assumption ensure that Q matrix is positive definite as well and the optimality problem becomes a strictly convex programming problem with a unique global optimum $\boldsymbol{\alpha}^*$.

Assumption 2: Nondegeneracy For the optimal solution $\boldsymbol{\alpha}^*$, $\nabla f\left(\boldsymbol{\alpha}^*\right)_i + b^* y_i \neq 0$ holds when $\alpha_i^* = 0, C$.

b^* denotes the value of $\max\limits_{i \in I_{up}(\boldsymbol{\alpha}^*)} -y_i \nabla f\left(\boldsymbol{\alpha}^*\right)_i \leq b \leq \min\limits_{j \in I_{low}(\boldsymbol{\alpha}^*)} -y_j \nabla f\left(\boldsymbol{\alpha}^*\right)_j$, if tbe algorithm takes infinite iterations. In fact, only elements whose $-y_i \nabla f\left(\boldsymbol{\alpha}^*\right)_i$ are b^* can still form violating pairs. This assumption guarantees that when k is larger enough, all bounded variables are constants, so at the optimal point $\alpha^* = arg\min\limits_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \boldsymbol{e}$ with constraint $y^T \boldsymbol{\alpha}^* = \Delta$ where Δ is a constant. By using Lagrange multiplier b^* to solve the optimal point, we can get a linear system like:

$$\begin{bmatrix} Q & y \\ y^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}^* \\ b^* \end{bmatrix} = \begin{bmatrix} e \\ \Delta \end{bmatrix} \quad (LS1)$$

In order to guarantee the decent direction, we solve the following problem in each iteration

$$\min_{\boldsymbol{d}} \frac{1}{2}\boldsymbol{d}^T Q_{BB} \boldsymbol{d} + \nabla f\left(\boldsymbol{\alpha}^k\right)_B^T \boldsymbol{d} s.t. \qquad y_B^T \boldsymbol{d} = 0$$

Similarity, we can get

$$\begin{bmatrix} Q_{\boldsymbol{BB}} & y_{\boldsymbol{B}} \\ y_{\boldsymbol{B}}{}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{d}^* \\ b^* \end{bmatrix} = \begin{bmatrix} -\nabla f\left(\boldsymbol{\alpha}^k\right)_{\boldsymbol{B}} \\ 0 \end{bmatrix} \quad (LS2)$$

With (LS1), we get $Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right) = Q\boldsymbol{\alpha}^k - e + b^* y = \nabla f\left(\boldsymbol{\alpha}^k\right) + b^* y$, as k is large enough.

Hence,

$$-YQ\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right) = -Y\nabla f\left(\boldsymbol{\alpha}^k\right) - b^* \boldsymbol{e}$$

where $Y \equiv \operatorname{diag}(y)$. Therefore, a maximal violating pair us obtained by the following formulation, since b^* does not affect the order of sequence:

$$arg\max_i \left(-y_i Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)_i\right) = arg\max_i \left(-y_i \nabla f\left(\boldsymbol{\alpha}^k\right)_i\right) arg\min_i \left(-y_i Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)_i\right) = arg\min_i \left(-y_i \nabla f\left(\boldsymbol{\alpha}^k\right)_i\right)$$

In order to deduce the local convergence rate, we also have to introduce two lemmas, which has been proved by Lin [4].

Lemma 1

If $v_1 \geq ... \geq v_l$, then $\sum_{i=1}^{l} v_i^2 \geq \frac{(v_1 - v_l)^2}{2}$

Lemma 2 If Q is invertible, then for any $\boldsymbol{x}$ such that (1) $\boldsymbol{e}^T \boldsymbol{x} = 0$; (2) $v \equiv Q\boldsymbol{x}$, $v \equiv Q\boldsymbol{x}$ $\max\limits_i \left((Q\boldsymbol{x})_i\right) = v^1 > v^l = \min\limits_i \left((Q\boldsymbol{x})_i\right)$, and $v^1 v^l \geq 0$, we have

$$\min\left(\left|v^1\right|, \left|v^l\right|\right) \leq \left(1 - \frac{\boldsymbol{e}^T Q^{-1} \boldsymbol{e}}{\sum_{i,j}\left|Q_{ij}^{-1}\right|}\right) \max\left(\left|v^1\right|, \left|v^l\right|\right)$$

On the basis of the assumptions and lemmas above, we talk about the difference between $\boldsymbol{\alpha}^{k+1}$ and $\boldsymbol{\alpha}^k$:

$$\begin{aligned}
&\left(\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^*\right)^T Q\left(\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^*\right) - \left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)^T Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right) \quad \text{(diff)} \\
&= 2\left(\boldsymbol{d}^k\right)^T \left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)_B + \left(\boldsymbol{d}^k\right)^T Q_{BB} \boldsymbol{d}^k \\
&= \left(\boldsymbol{d}^k\right)^T \left(2\left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)_B - \nabla f\left(\boldsymbol{\alpha}^k\right)_B - b^k y_B\right) \\
&= \left(\boldsymbol{d}^k\right)^T \left(\left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)_B + \left(b^* - b^k\right) y_B\right) \\
&= \left(\boldsymbol{d}^k\right)^T \left(\left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)_B + \left(b^k - b^*\right) y_B\right) \\
&= -\left[-\left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)_B + \left(b^* - b^k\right) y_B\right]^T Q_{BB}^{-1} \left[-\left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)_B + \left(b^* - b^k\right) y_B\right]
\end{aligned}$$

we define that

$$\hat{Q} \equiv Y_B Q_{BB}^{-1} Y_B \quad \text{and} \quad v \equiv -Y\left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)$$

where $Y_B \equiv \operatorname{diag}(y_B)$,then $v_B \equiv -Y_B\left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)_{\mathrm{B}}$

So, the (diff) formulation can be rewritten as

$$-\left[v_B + \left(b^* - b^k\right) e_B\right]^T \hat{Q}\left[v_B + \left(b^* - b^k\right) e_B\right]$$

We define

$$v^1 \equiv \max_i (v_i) = \max_{i \in B} (v_i), v^l \equiv \min_i (v_i) = \min_{i \in B} (v_i)$$

and denote min(eig(.)) and max(eig(.)) to be the minimal and maximal eigenvalues of a matrix respectively. Then,

$$\begin{aligned}
&-\left[v_B + \left(b^* - b^k\right) e_B\right]^T \hat{Q} \left[v_B + \left(b^* - b^k\right) e_B\right] \\
&\geq \min\left(eig\left(\hat{Q}\right)\right) \left[v_B + \left(b^* - b^k\right) e_B\right]^T \left[v_B + \left(b^* - b^k\right) e_B\right] \\
&\underset{lemma1}{\geq} \min\left(eig\left(\hat{Q}\right)\right) \frac{(v_1 - v_l)^2}{2} \\
&\underset{lemma2}{\geq} \frac{\min\left(eig\left(\hat{Q}\right)\right)}{2} \left(\frac{y^T Q^{-1} y}{\sum_{i,j} \left|Q_{ij}^{-1}\right|}\right)^2 \max\left(\left|v^1\right|, \left|v^l\right|\right)^2 \qquad (IEQ1) \\
&\geq \frac{\min\left(eig\left(\hat{Q}\right)\right)}{2l} \left(\frac{y^T Q^{-1} y}{\sum_{i,j} \left|Q_{ij}^{-1}\right|}\right)^2 \left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)^T Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right) \\
&\geq \frac{\min\left(eig\left(\hat{Q}\right)\right)}{2l \max\left(eig\left(Q^{-1}\right)\right)} \left(\frac{y^T Q^{-1} y}{\sum_{i,j} \left|Q_{ij}^{-1}\right|}\right)^2 \left(Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)\right)^T Q^{-1} Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right) \\
&\geq \frac{\min\left(eig\left(\hat{Q}\right)\right)}{2l \max\left(eig\left(Q^{-1}\right)\right)} \left(\frac{y^T Q^{-1} y}{\sum_{i,j} \left|Q_{ij}^{-1}\right|}\right)^2 \left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)^T Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)
\end{aligned}$$

Therefore, a constant $c \equiv 1 - \min_B \left(\frac{\min\left(eig\left(\hat{Q}\right)\right)}{2l \max\left(eig\left(Q^{-1}\right)\right)} \left(\frac{y^T Q^{-1} y}{\sum_{i,j} \left|Q_{ij}^{-1}\right|}\right)^2\right), 0 < c < 1$ exists to make sure when k is large,

$$\left(\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^*\right)^T Q\left(\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^*\right) \leq c\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)^T Q\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right)$$

holds, which show that the algorithm is with linear convergence.

3.3 Analysis on Parameters

In this part, we perform experiments on different regularisation parameter $C = 0.001, 0.1, 0.5, 5, 10$ and kernel parameter $\mu = 1, 5, 10, 15$ and then we compare the convergence and performance on them.

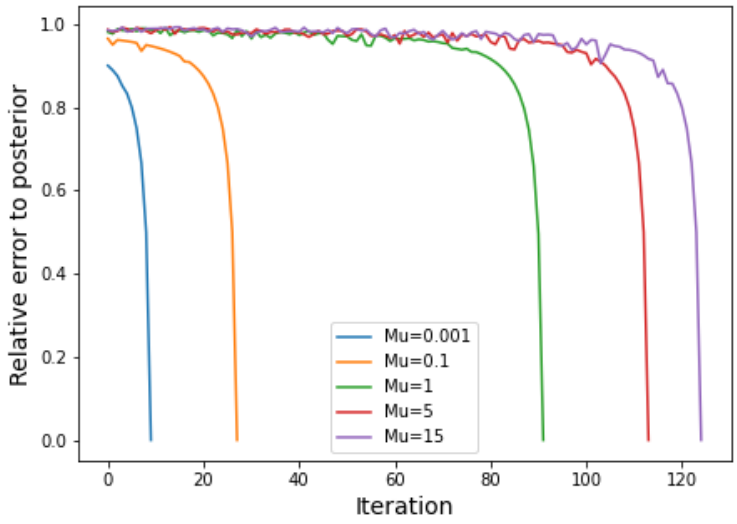

Figure 4: Convergence plot for multiple μ

From Figure 4, we can find that iteration speed to stationary point is slower with μ increasing, this is because the larger μ brings much more computation complexity when we create a kernel matrix with large parameter. However, from table 1 we find that when $\mu = 0.001$, the training and test error are around 0.5, which means the

Table 1: Training and test error for multiple μ

μ	Train error(%)	Test error(%)
0.001	0.5100	0.5267
0.1	0.0000	0.0000
1	0.0000	0.0000
5	0.0000	0.0000
15	0.0000	0.0067

SVM classification does not work here. This is attributes to the Kernel Matrix K is almost not S.P.D under this case, not ensuring the optimality problem has the unique global minimiser. When $\mu = 15$, test error begins to increase, hence the overfitting occurs because of too complicated mapping brought by RBF kernel with a high parameter.

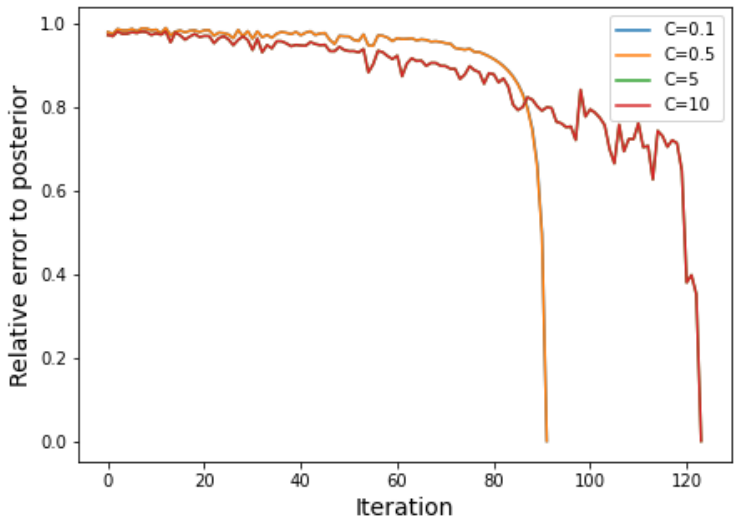

Figure 5: Convergence plot for multiple C

Form the graph above, we find that different parameters has the same iteration situation: when C=0.1 or 0.5,iteration stops at 92th iteration; when C=5 or 10, iteration stops at 134th iteration. This is because parameter C just serves as a range bound when we select maximal violating pairs. So the situation where the feasible $\boldsymbol{\alpha}$ are same might occur when when we choose different C, leading to the same iteration situation.

3.4 Analysis on Complexity

3.4.1 Time Complexity

When we implement the SMO-Light algorithm, we mainly use two for-loops to find the maximal violating pair. In the worst case, we have to travel all m points of data set in each loop. Hence the time complexity for SVM-Light is $O\left(m^2\right)$ [1].

3.4.2 Space Complexity

In SMO-Light algorithm, the working set's size q is 2. So the highest memory requirements are depends on how many rows of Q are stored for each iteration. In this case, $O(qm)$ floating numbers are needed to stored for updating parameter, computing gradients, solving sub-problems phases. Also, Q_{BB} needs $O(q^2)$ space and $\boldsymbol{\alpha}^k$ needs $O(m)$ space. So the total space complexity per iteration should be $O(qm + q^2 + m)$ [1].

3.5 Analysis on Classification Performance

In order to verify SVM-Light's validity, we select $\mu = 1$ and $C = 0.5$ as the parameters, which are the paramenter pair with the best convergence performance. And then we measure the training and test error based on this and visulise the classification result.

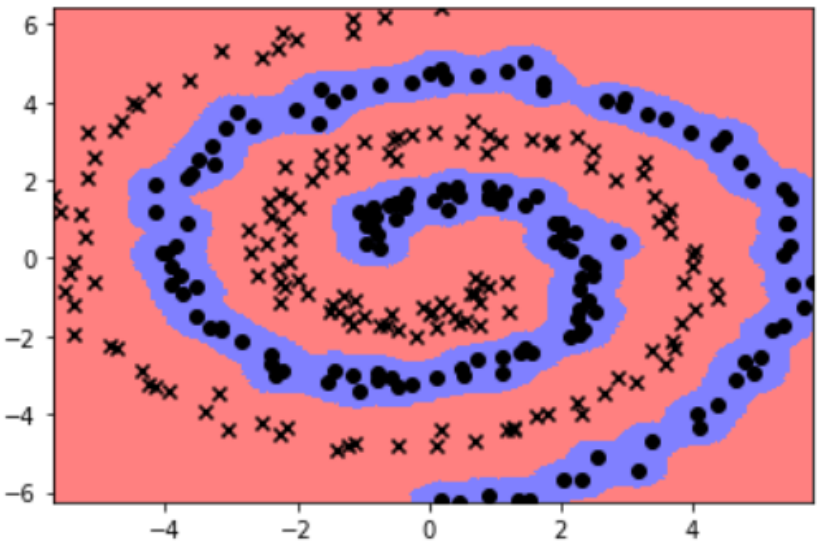

Figure 6: Classification result

The training error and test error are measured as $\frac{number\ of\ correct\ prediction}{data\ set\ size}$, which are both 0 in this case. This means the SVM-Light Algorithm works very well and classify all test data points correctly.

4 Conclusion

Based on experiments and theories, we can conclude that the SVM-Light is with a global asymptotic convergence and possess a faster convergence than SMO-Simplified. The local convergence could be linear for SVM-Light and we had better to choose μ between 0.1 and 5 to ensure a faster convergence and avoid non-S.P.D kernel matrix or overfitting happen. Finally, SVM-Light performs very well on spiral data set, dividing all test data correctly.

References

[1]

[2] A.Ng. Cs229 lecture notes. In *CS229 lecture notes*, page 1(1):1–3. 2020.

[3] Hang Li. Svm. In *Statistical learning methods*. Tsinghua University Press, 2012.

[4] Chih-Jen Lin. Linear convergence of a decomposition method for support vector machines. 2001.

[5] Chih-Jen Lin. Asymptotic convergence of an smo algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13(1):248–250, 2002.