Name: Wenguang Lin Programme: MSc Robotics and Computation Student Number: 15034455

COMP0120 Numerical Optimisation

Project

1 Introduction

The purpose of the project is to design a Support Vector Machine (SVMs) for solution of classification problems. This report mainly talks about the whole structure of SVM including its methodology, three different optimization methods (Quadratic Penalty, Augmented Lagrangian and InteriorPoint Barrier) with four different Kernels (linear, polynomial, RBF and Sigmoid). I also talk about how to deal with multi-class classification problem by using the method called one against all which is widely used in machine learning field. Thea data set I used is Fisher Iris data containing three classes, four features and 150 samples. I have manually splited 80% of the data as the traning set and 20% of the data as the testing set.

2 Methodology

2.1 Decision rule

I want to optimize the margin of the classifier by making those regions (the region having +1, and the region having -1) as far as possible.
a function is defined as:

$$f(x) = w * x^{'} + b \tag{1}$$

$$\begin{aligned} f(x) > 1 \; in \; region \; +1 \\ f(x) < 1 \; in \; region \; -1 \end{aligned} \tag{2}$$

It will passes through zeros in center.

2.2 Computing the margin width

$$x^{+} = x^{-} + r * w \tag{3}$$

where, x^{-} is the points subjective to $f(x^{-}) = -1$, x^{+} is the closest point with $f(x^{+}) = +1$, r is a scalar multiple and w is the vector which is perpendicular to the boundary.
Points on the margin need to satisfy:

$$\begin{aligned} w \cdot x^{-} + b = -1 \\ w \cdot x^{+} + b = -1 \end{aligned} \tag{4}$$

Inserting the equation 2.2 into the equation 2.2

$$\begin{aligned} w \cdot (x + rw) + b = +1 \\ \rightarrow r||w||^2 + w \cdot x + b = +1 \\ \rightarrow r||w||^2 - 1 = +1 \\ \rightarrow r = \frac{2}{||w||^2} \end{aligned} \tag{5}$$

$$Margin = ||x^{+} - x^{-}|| = ||rw|| = \frac{2}{||w||^2}||w|| = \frac{2}{\sqrt{w^T w}} \tag{6}$$

Our objective is to separate the margin as far as possible. thus the optimal w can be obtained by using maximum margin classifier:

$$w^{*} = \underset{w}{\operatorname{argmax}} \frac{2}{\sqrt{w^T w}} \tag{7}$$

Changing the maximization problem to a minimization problem and the primal problem can be defined as :

$$\begin{aligned} w^* = \underset{w}{\operatorname{argmin}} \sum_j w_j^2 \\ subjectiveto \\ y^i = +1 \rightarrow w \cdot x^i + b \geq +1 \\ y^i = -1 \rightarrow w \cdot x^i + b \leq +1 \end{aligned} \tag{8}$$

The above is the example of a quadratic cost function with a linear constraints. I need to minimize quadratic function's parameter. The constraint can be rewritten as:

$$y^i(w \cdot x^i + b) \geq +1 \tag{9}$$

where y^i can be +1 or -1

2.3 Lagrangian optimization

I can make the hard satisfying constraint easier to satisfy by introducing Lagrange multiplier $\alpha(oneperconstraint)$. I can use Lagrangian multipliers to enforce inequality constraints. The Lagrangian equation can be expressed as:

$$\begin{aligned} L = \frac{1}{2}\sum_j w_j^2 + \sum \alpha_i(1 - y^i(w.x^i + b)) \\ w^* = \underset{w}{\operatorname{argmin}} \underset{\alpha \geq 0}{\operatorname{argmax}} \frac{1}{2}\sum_j w_j^2 + \sum \alpha_i(1 - y^i(w.x^i + b)) \end{aligned} \tag{10}$$

Differentiating the Lagrangian equation and stationary condition with respect to w can be shown as:

$$\begin{aligned} \frac{\partial L}{\partial w} = w - \sum \alpha_{ii} x_i = 0 \rightarrow w^* = \sum_i \alpha_i y_i x_i \\ \frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \rightarrow \sum \alpha_i y_i = 0 \end{aligned} \tag{11}$$

Inserting the equation 2.3 into the equation 2.3, the new Lagrangian equation can be shown as:

$$\begin{aligned} L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ w^* = \underset{\alpha \geq 0}{\operatorname{argmax}} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j \end{aligned} \tag{12}$$

2.4 data are not linearly separable

In order to deal with data which is not fully linearly separable, I am going to relax the constraints for the equation 2.2 slightly to allow for misclassified points.Soft margin is used here to introduce slack variables for violated constraints.

$$\begin{aligned} w^* = \underset{w,\epsilon}{\operatorname{argmin}} \sum_j w_j^2 + R\sum_i \epsilon^i \\ subjective\ to \\ y^i(w^T x^i + b) \geq +1 - \epsilon^i\ (violate\ margin\ by\ \epsilon) \\ \epsilon \geq 0 \end{aligned} \tag{13}$$

where, ϵ is a slack variable ≥ 0, $R\sum_i \epsilon^i$ is the penalty term.
if R is chosen very large, we will pay more attention to make sure no data violate the margin. If R is chosen very small, we will maximize the margin for most of data, but allowing some of data to violate a bit. The cost R is assigned to distance from margin. We can always initialize a solution w and ϵ and B to some value which satisfy the

constraint, even if it does not minimize the objective function.
The optimal ϵ^* as a function of w, can be shown analytically as:

$$J_i = max[0, 1 - y^i(w^i + b)] \tag{14}$$

inserting back to the equation 13,

$$w^* = \underset{w}{\text{argmin}} \sum_j w_j^2 + R \sum_i J_i(y^i, w \cdot x^i + b) \tag{15}$$

where, $\sum_i J_i(y^i, w \cdot x^i + b)$ is a hinge loss as surrogate loss As you can see in the figure 1, the hinge loss is equal to

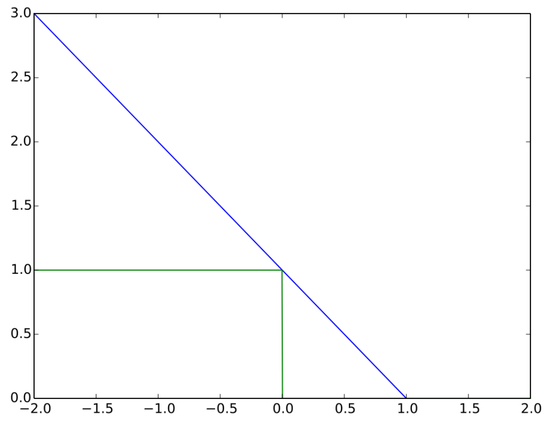

Figure 1: Hinge loss(blue) vs zero one loss (green)

0 when $w + b$ is greater than 1. As $w + b$ is smaller than 1, the hinge loss is increasing linearly.

In this soft margin SVM, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it. Reformatting as a Lagrangian, which as mentioned before we need to minimize the respect to w, b and ϵ and maximize with respect to α (where $\alpha_i \geq 0$, $\mu \geq 0$): the Lagrangian Equation for the data which are not fully linearly separable can be shown as:

$$L_p \equiv \frac{1}{2}||w||^2 + R\sum_{i=1}^{L}\epsilon_i - \sum_{i=1}^{L}\alpha_i[y_i(x_i \cdot w + b) - 1 + \epsilon_i] - \sum_{i=1}^{L}\mu_i\epsilon_i \tag{16}$$

Differentiating with respect to w, b and ϵ_i and the stationary point are

$$\begin{aligned} \frac{\partial L}{\partial w} = w - \sum \alpha_{ii} x_i = 0 \rightarrow w^* = \sum_i \alpha_i y_i x_i \\ \frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \rightarrow \sum \alpha_i y_i = 0 \\ \frac{p}{\partial \epsilon_i} = 0 \rightarrow R = \alpha_i + \mu_i \end{aligned} \tag{17}$$

Now, inserting these derivatives to the Lagrangian Equation,the soft margin dual form can be shown as:

$$w^* = \underset{0 \leq \alpha \leq R}{\text{argmax}} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{18}$$

The maximization actually depends on the dot product $x_i \cdot x_j$, therefore, I will introduce different types of kernels to deal with this dot product. The new equation with kernel is shown as:

$$w^* = \underset{0 \leq \alpha \leq R}{\text{argmax}} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{19}$$

The equation can be further reformatted into a quadratic programming form:

$$\underset{\alpha}{\text{argmax}}[\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\alpha^T H\alpha]$$
$$s.t.\ 0 \le \alpha_i \le R \tag{20}$$
$$\sum_{i=1}^{L}\alpha_i y_i = 0$$

where, $H_{ij} = y_i y_j x_i \cdot x_j = y_i y_j k(x_i, x_j)$

The optimization of the quadratic programming (20) can be done by using a Quadratic Penalty Method or a Augmented Lagrangian Method.
The set of Support Vectors S can be determined by finding the indices such that $(0 < \alpha_i < R)$ R is set as 2 in this case. The constant b can be calculated as:

$$b = \frac{1}{N_s}\sum_{s\in S}(y_s - \sum_{m\in S}\alpha_m y_m x_m \cdot x_s) \tag{21}$$

$$= \frac{1}{N_s}\sum_{s\in S}(y_s - \sum_{m\in S}\alpha_m y_m k(x_m, x_s))$$

where, N_s is the number of support vectors

2.5 Optimization

I have four different optimization methods used for optimization the quadratic function 20. They are Quadratic Penalty Method, Augmented Lagrangian Method, InteriorPoint Barrier and quadratic programming package (which is already installed in MATLAB) are used as the optimization method for the equation 20. I used the quadratic programming package as a reference to see how well the optimization I wrote can performance. The f(x) here is the quadratic programming function 20. The initial guess for the alpha is 0.2*ones(N,1). The equality constraint is $H(x) = Aeq * x$ where, Aeq is the transpose of the testing set (Y'), x is the training set. The inequality constraints is: $0 < \alpha < cost$, where, cost is a vector with a value of 2.

- A line search method with a Newton direction and backtracking line search are used to solve the unconstrained problem at each step for all implemented methods
- And I used $||x_k - x_{k-1}|| < \epsilon$ as a stopping criterion for all implemented methods.

The Quadratic Penalty Method:
$\mu_o = 1$ is chosen for the initial penalty weight. The tolerance is 10e-14. The updating parameter t is 2.
Due to there are two inequality constraints $\alpha_i - 0 \ge 0$ and $R - \alpha \ge 0$, the merit function can be shown as:

$$Q(x;\mu) := f(x) + \frac{\mu}{2}\sum_{i=1}^{p}h_i^2(x) + \frac{\mu}{2}\sum_{i=1}^{m}([f_{1,i}(x)]^+)^2 + \frac{\mu}{2}\sum_{i=1}^{m}([f_{2,i}(x)]^+)^2 \tag{22}$$

where h_i is the equality constraint and two $f_{1,i}$ and $f_{2,i}$ represent two inequality constraints and $[f_i]^+ := max\{f_i, 0\}$. This means when the inequality constraint is satisfied, the inequality term will disappear as

Framework: for a sequence $\{\mu_k\}$:$\mu_k \rightarrow$ as k $\rightarrow$, increasingly penalising the constraint compute the (approximate, $||\nabla_x Q(x_k;\mu_k)|| \le \tau_k, tau \rightarrow 0$ sequence $\{x_k\} \rightarrow x^*$ of minimisers $x_k of Q(x;\mu_k)$.

The gradient of the merit function can be expressed as:

$$\nabla_x Q(x_k;\mu_k) = \nabla f(x_k) + \sum_{i=1}^{p}\mu_k h_i(x_k)\nabla h_i(x_k) + \sum_{i=1}^{p}\mu_k[f_{1,i}(x_k)]^+[\nabla f_{1,i}(x_k)]^+ + \sum_{i=1}^{p}\mu_k[f_{2,i}(x_k)]^+[\nabla f_{2,i}(x_k)]^+ \tag{23}$$

The Hessian of the merit function can be expressed as:

$$\begin{aligned}\nabla^2_{xx}Q(x;\mu_k)p_n = -\nabla^2_x f(x) &+ \sum_{i=1}^{p}\mu_k h_i(x)\nabla^2 h_i(x) + \mu_k \nabla h(x)\nabla h(x)^T \\ &+ \sum_{i=1}^{p}\mu_k [f_{1,i}(x)]^+\nabla^2[f_{1,i}(x)]^+ + \mu_k[\nabla f_1(x)]^+[\nabla f_1(x)^T]^+ \\ &+ \sum_{i=1}^{p}\mu_k [f_{2,i}(x)]^+\nabla^2[f_{2,i}(x)]^+ + \mu_k[\nabla f_2(x)]^+[\nabla f_2(x)^T]^+\end{aligned} \tag{24}$$

The Augmented Lagrangian Method:
$\mu = 10$ and $\nu_0 =$ a vector of ones with size (2N+1)is chosen as a fixed penalty weight and initial Lagrange multiplier. The tolerance is 1e-16.
The merit function:

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}(x,\nu,\mu) := f(x) &+ \sum_{i=1}^{p}\nu_i h_i(x) + \frac{\mu}{2}\sum_{i=1}^{p}h_i^2(x) \\ &+ \sum_{i=1}^{p}\nu_i[f_{1,i}(x)]^+ + \frac{\mu}{2}\sum_{i=1}^{p}[f_{1,i}^2(x)]^+ \\ &\sum_{i=1}^{p}\nu_i[f_{2,i}(x)]^+ + \frac{\mu}{2}\sum_{i=1}^{p}[f_{2,i}^2(x)]^+\end{aligned} \tag{25}$$

The gradient of the merit function can be expressed as:

$$\begin{aligned}\nabla_x\mathcal{L}_{\mathcal{A}}(x_k,\nu^k;\mu_k) = \nabla f(x_k) &+ \sum_{i=1}^{p}[\nu_i^k + \mu_k h_i(x_k)]\nabla h_i(x_k) \\ &+ \sum_{i=1}^{p}[\nu_i^k + \mu_k[f_{1,i}(x)]^+]\nabla[f_{1,i}(x)]^+ \\ &+ \sum_{i=1}^{p}[\nu_i^k + \mu_k[f_{2,i}(x)]^+]\nabla[f_{2,i}(x)]^+\end{aligned} \tag{26}$$

The hessian of the merit function can be expressed as:

$$\begin{aligned}\nabla^2_{xx}\mathcal{L}_{\mathcal{A}}(x_k,\nu^k;\mu_k) = \nabla^2 f(x_k) &+ \sum_{i=1}^{p}[\nu^*\nabla^2 h_i(x_k)] + \mu_k\nabla h_i(x_k)\nabla h_i(x_k)^T \\ &+ \sum_{i=1}^{p}[\nu^*\nabla^2[f_{1,i}(x)]^+] + \mu_k\nabla[f_{1,i}(x)]^+\nabla[f_{1,i}(x)^T]^+ \\ &+ \sum_{i=1}^{p}[\nu^*\nabla^2[f_{2,i}(x)]^+] + \mu_k\nabla[f_{2,i}(x)]^+\nabla[f_{2,i}(x)^T]^+\end{aligned} \tag{27}$$

The InteriorPoint Barrier Method:
$\mu_o = 1$ is chosen for the initial penalty weight. The tolerance is 10e-10. The updating parameter t is 2. The maximum iteration is 1000. The initial guess for alpha is 0.01* ones(N,1).
The merit function:

$$\mathcal{I}(x) = tf(x) + \phi(x) \tag{28}$$

where,

$$\phi(x) = -(\sum_{i=1}^{m}log(-f_{1,i}(x)) + \sum_{i=1}^{m}log(-f_{2,i}(x)))$$

$$\text{dom }\phi = \{x \in \mathcal{R}^n : f_{1,i}(x) < 0 \text{ and } f_{2,i}(x) < 0, i = 1,\ldots,m\}$$

The gradient of the merit function:

$$\nabla_x \mathcal{I}(x_k) = t\nabla f(x_k) + \nabla\phi(x) \tag{29}$$

where,

$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{-f_{1,i}(x)} \nabla f_{1,i}(x) + \sum_{i=1}^{m} \frac{1}{-f_{2,i}(x)} \nabla f_{2,i}(x)$$

The hessian of the merit function:

$$\nabla_{xx}^2 \mathcal{I}(x) = t\nabla_x^2 f(x) + \nabla^2\phi(x) \tag{30}$$

where,

$$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{f_{1,i}(x)^2} \nabla f_{1,i}(x)\nabla f_{1,i}(x)^T + \sum_{i=1}^{m} \frac{1}{-f_{1,i}(x)} \nabla^2 f_{1,i}(x) + \sum_{i=1}^{m} \frac{1}{f_{2,i}(x)^2} \nabla f_{2,i}(x)\nabla f_{2,i}(x)^T + \sum_{i=1}^{m} \frac{1}{-f_{2,i}(x)} \nabla^2 f_{2,i}(x)$$

2.6 Label prediction

The new point point x_new can be classified by evaluating the following equation:

$$y_{predication} = sgn(w \cdot x_{new} + b) \tag{31}$$

2.7 Kernels

I have introduced three different kernels to help with the optimization of the equation 19.

2.7.1 polynomial kernel function

$$k(a,b) = (1 + \sum_j a_j b_j)^d \tag{32}$$

where, d is the polynomial condition and equal to 2

2.7.2 Radial Basis Function

:

$$k(a,b) = exp(\frac{-(a-b)^2}{2\sigma^2}) \tag{33}$$

where, σ used to control over and under fitting
when σ has a large value, all data point looks similar to the kernel. When σ has a small value, only a few points similar to the test point.

2.7.3 Saturating Sigmoid like

$$k(a,b) = tanh(ca^T b + h) \tag{34}$$

where, c and h are both equal to $\frac{1}{\text{number of data}}$

3 Muti-class classifictaion using one-against-all

Support Vector Mahcine (SVM) was designed to solve a binary classification. However, the data I downloaded has three classes which is a multi-class classification problem. The method I am using here to effectively extend for a multi-class classification is called one-against-all. This is also the earliest used implementation for SVM multi-class classification. It construct k SVM models where k is the number of classes (3 in our case). The mth SVM is trained against the other two examples. The mth SVM is considered with positive labels and The other examples are considered with negative labels.

Thus given l training data set $(x_1, y_1), \ldots, (x_l, y_l)$, where $x_i \in R^n, i = 1, \ldots, l$ and $y_i \in \{1, 2, 3\}$ is the class of x. The mth SVM solves the following problem:

$$\begin{aligned} \underset{w^m, b^m, \epsilon^m}{\operatorname{argmin}} \ \frac{1}{2}(w^m)^T w^m + C \sum_{i=1}^{l} \epsilon_i^m \\ (w^m)^T \phi(x_i) + b^m \geq 1 - \epsilon_i^m, if y_i = m, \\ (w^m)^T \phi(x_i) + b^m - 1 + \epsilon_i^m, if y_i \neq m, \\ \epsilon_i^m \geq 0, i = 1, \ldots, l, \end{aligned} \tag{35}$$

where, the training data x_i are mapped to a higher dimensional space by the function ϕ and C is the penalty parameter. What We want to do here is minimize $\frac{1}{2}(w^m)^T w^m$. That means that we would like to maximize $\frac{2}{||w^m||}$, the margin between two groups of data. Because our data are not linear separable, I set a penatlty term $C \sum_{i=1}^{l} \epsilon_i^m$ which is used to reduce the number of training errors. The basic concept behind SVM is to search for a balance between the regularization term $\frac{1}{2}(w^m)^T w^m$ and the training errors.
After solving (1), there are three decision boundaries due to three classes.

$$\begin{aligned} (w^1)^T \phi(x) + b^1 \\ (w^2)^T \phi(x) + b^2 \\ (w^3)^T \phi(x) + b^3 \end{aligned} \tag{36}$$

We say x is in the class which has the largest value of the decision function:

$$class of x \equiv \underset{m=1,\cdot,k}{\operatorname{argmax}}((w^m)^T \phi(x) + b^m) \tag{37}$$

The accuracy of the prediction produced by the multi-class SVM classifier using one against all can be calculated in the following way:

- Each SVM classifier will produce a score. Comparing among all scores generated from different SVM models, the prediction label can be determined by the highest score.

$$predlabel_i = find([scores1_i, scores2_i, scores3_i] == max(score1_i, scores2_i, scores3_i)) \tag{38}$$

- then the accuracy can be calculated as:

$$\frac{\sum_i predlabel_i == ytest_i)}{length(ytest)} * 100 \tag{39}$$

4 Results and Discussions

In this section, I will present the results of three different optimization(Quadratic Penalty, Augmented Lagrangian and InteriorPoint Barrier) with four different kernels (linear, polynomial, RBF, and sigmoid like kernel) respectively.

4.1 Data points

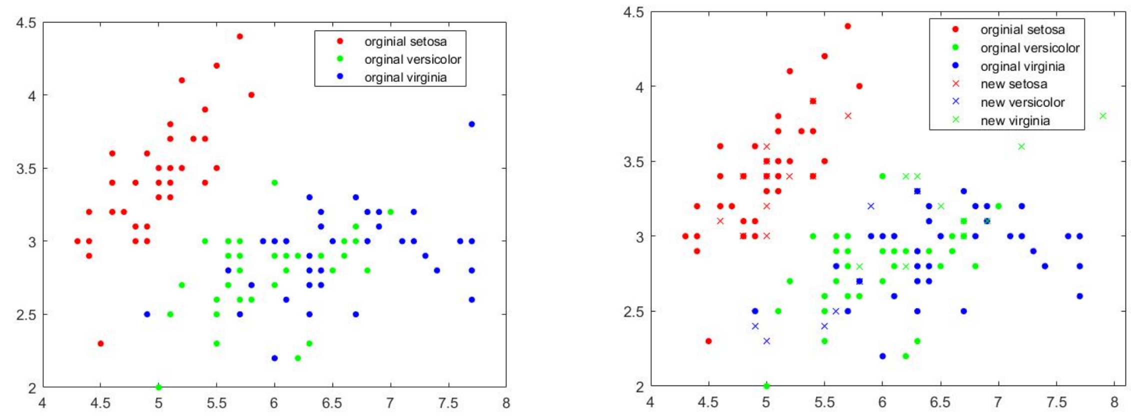

Fig. 2: it shows training data points in the left figure, and training data + new data set in the right figure.

As you can see in the figure 2, there are three classes(setosa, versicolor and virginia). I have divided 80% of the data as the training data and divided the other 20% of data as the testing data.

The following sections will show the rate of convergence quantified by ratio of distances to optimal point $\frac{||x_k - x^*||_2}{||x_{k-1} - x^*||_2}$ against iteration k.

4.2 Quadratic optimization with a linear kernel

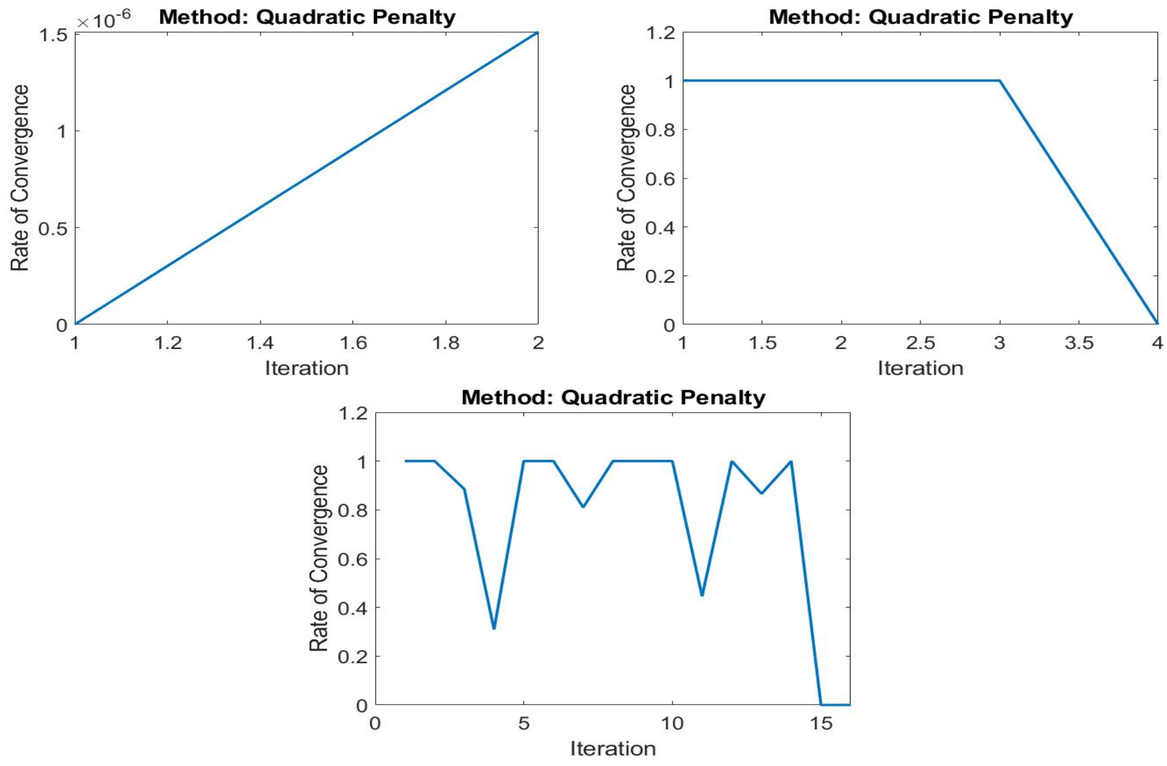

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.3 Quadratic optimization with a polynomial kernel

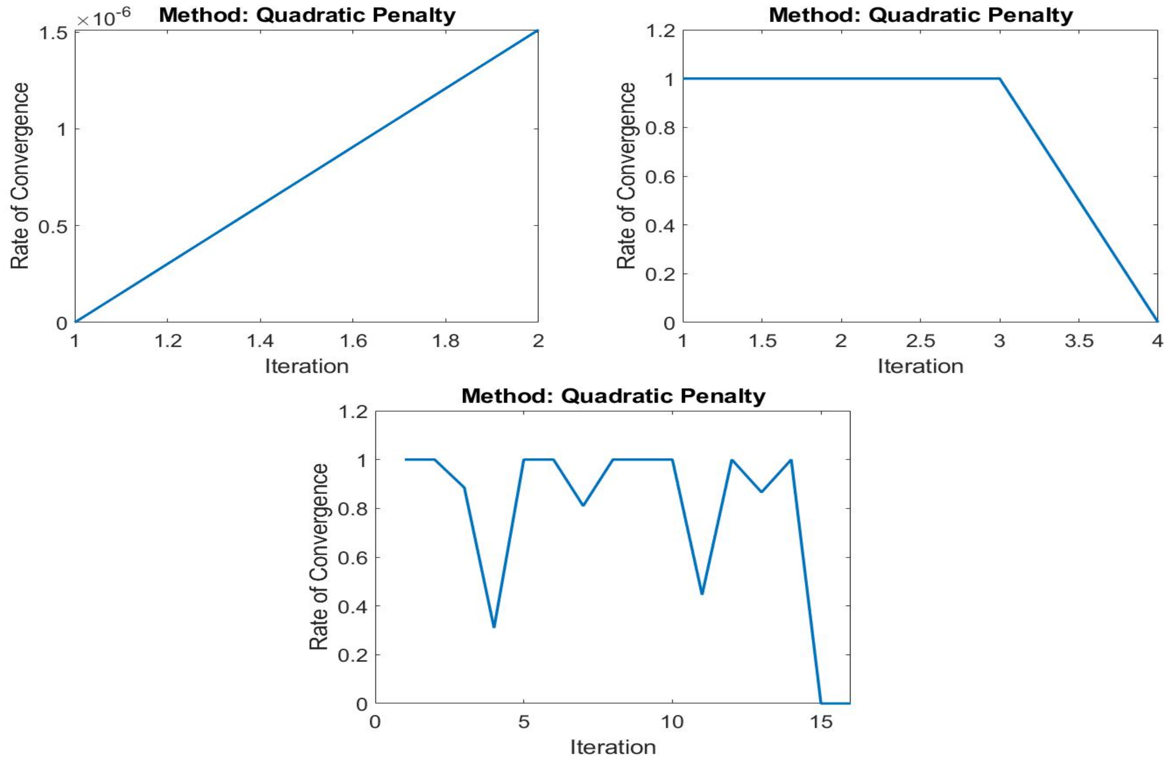

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.4 Quadratic optimization with a RBF kernel

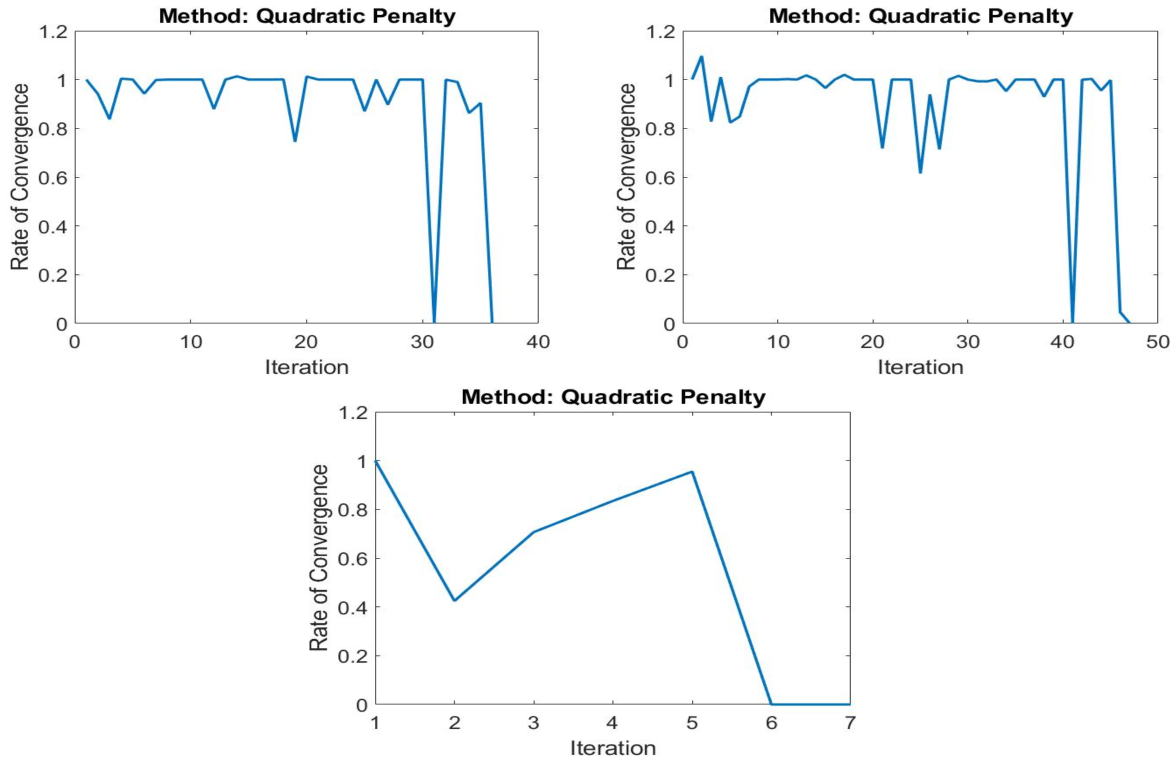

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.5 Quadratic optimization with a Sigmoid kernel

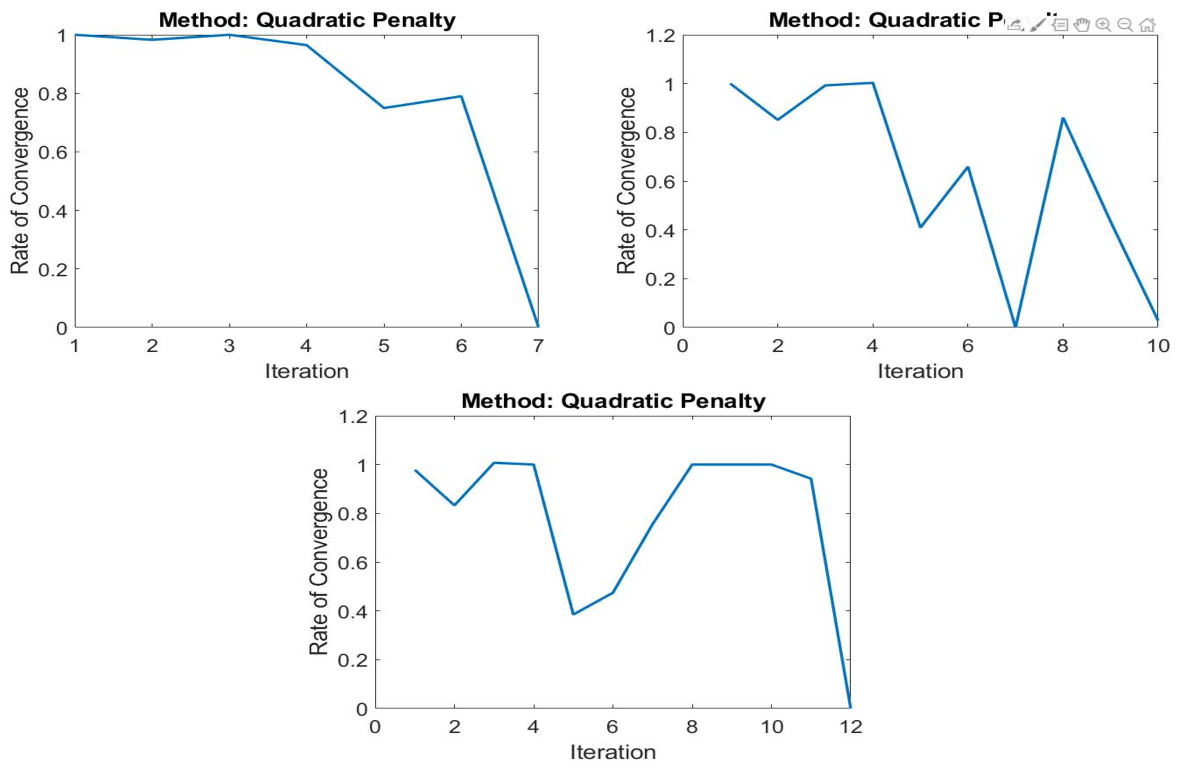

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.6 Augmented Lagrangian with a linear kernel

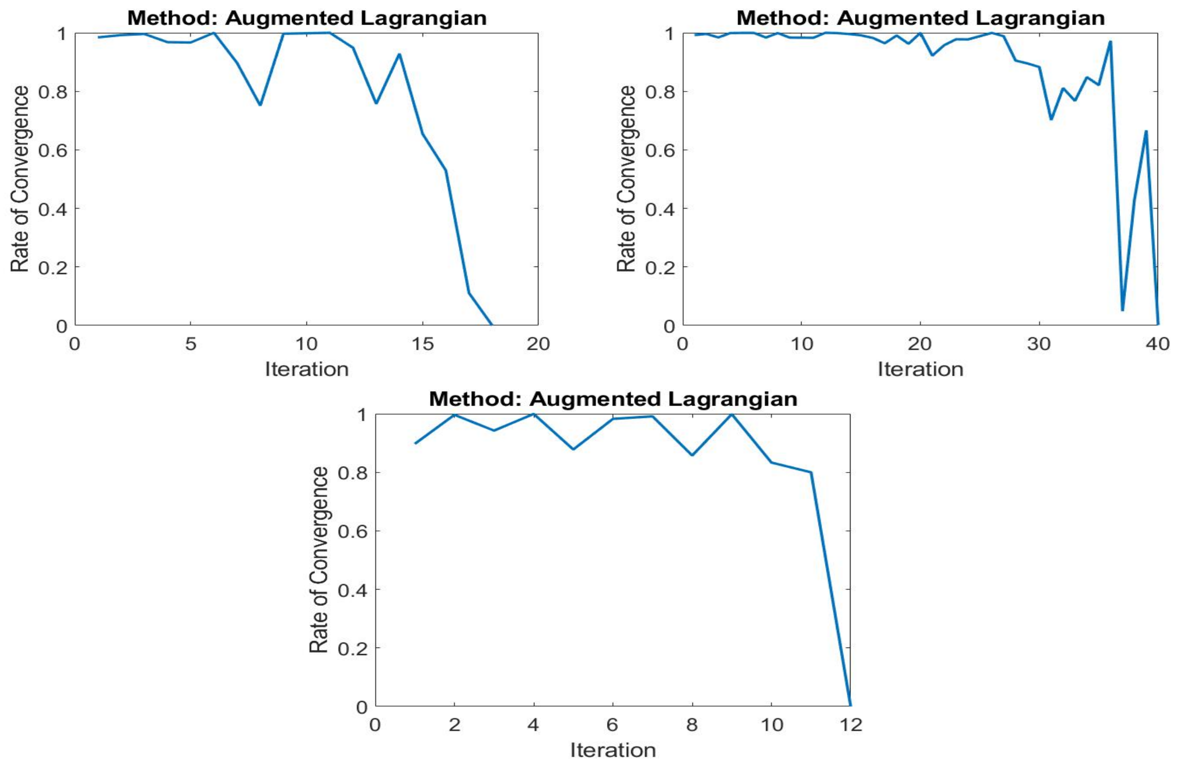

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.7 Augmented Lagrangian with a polynomial kernel

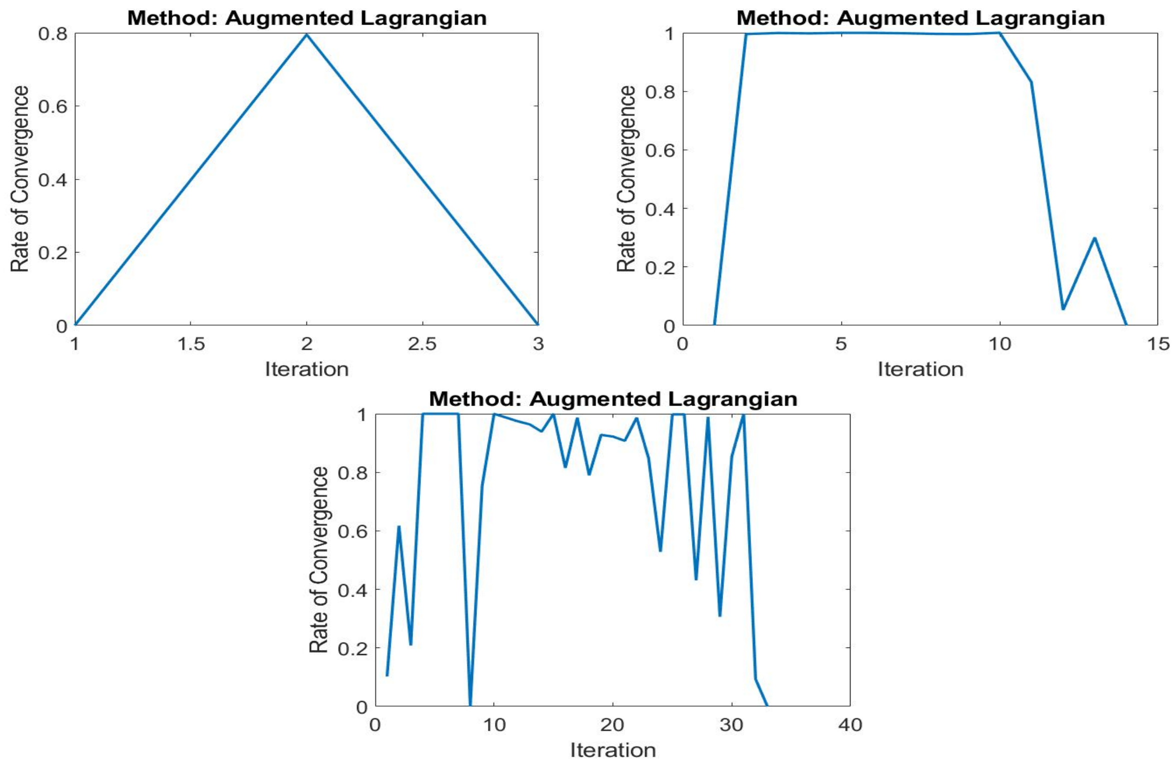

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.8 Augmented Lagrangian with a RBF kernel

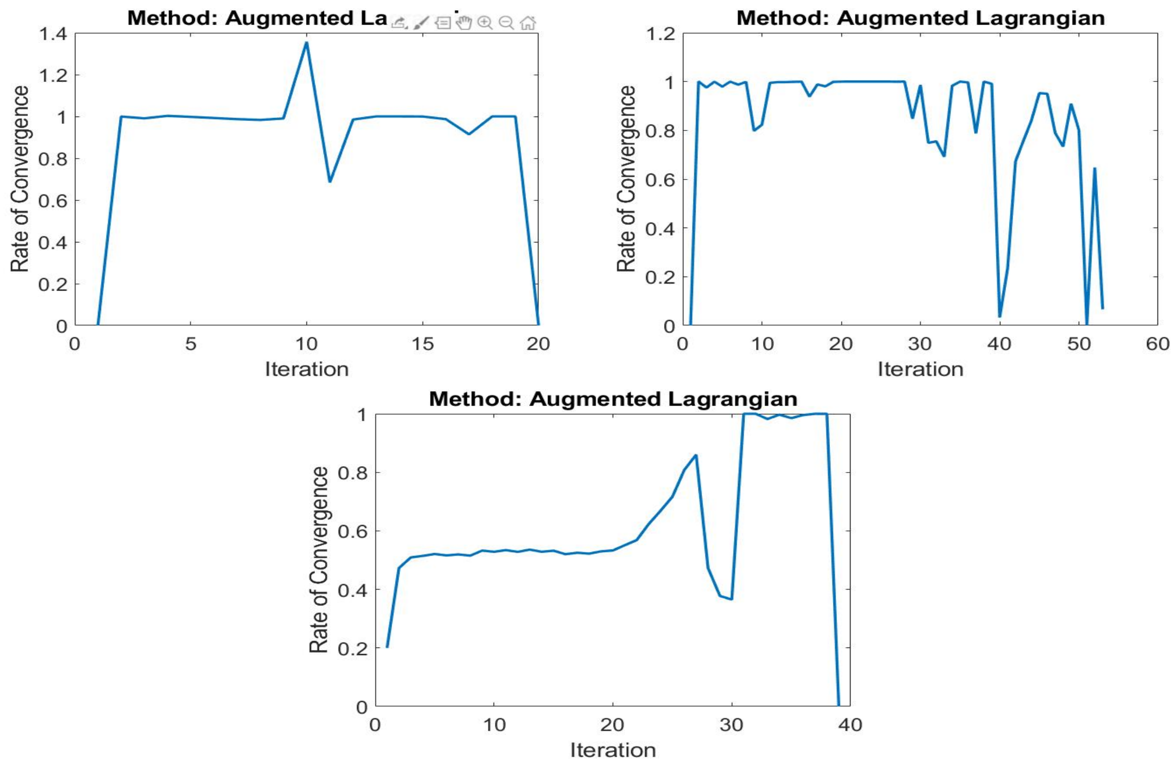

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.9 Augmented Lagrangian with a sigmoid kernel

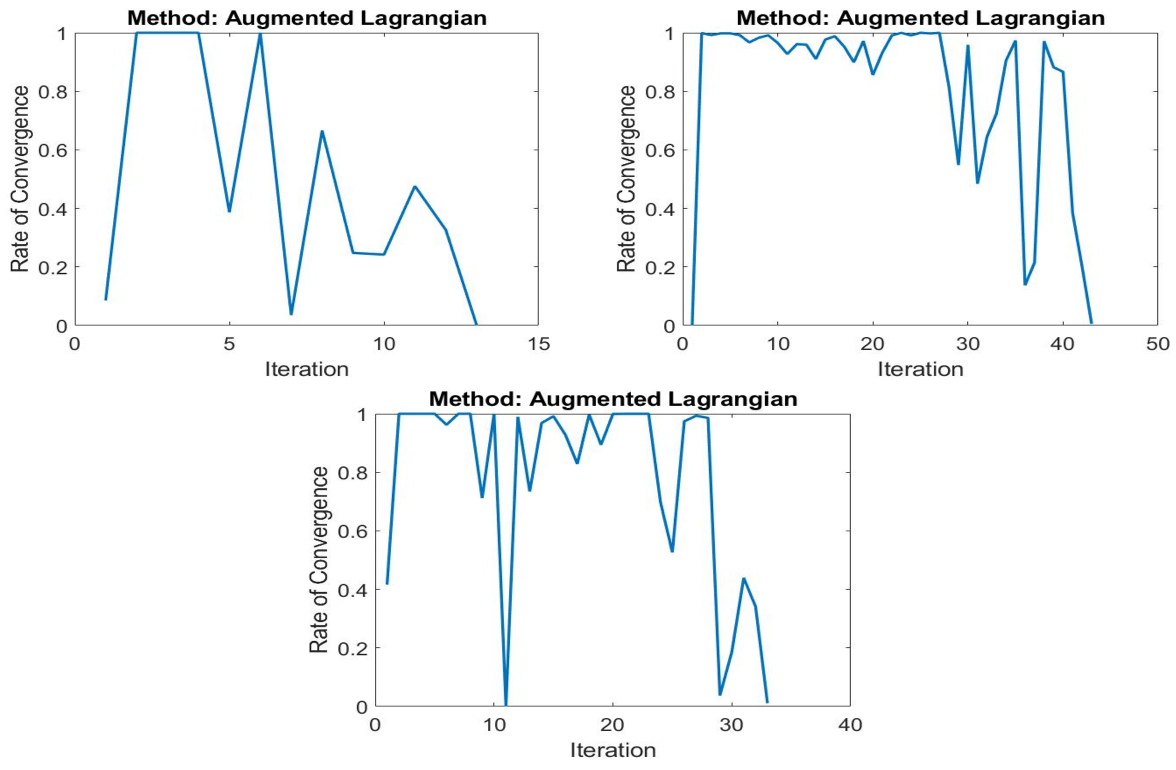

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.10 InteriorPoint Barrier with a linear kernel

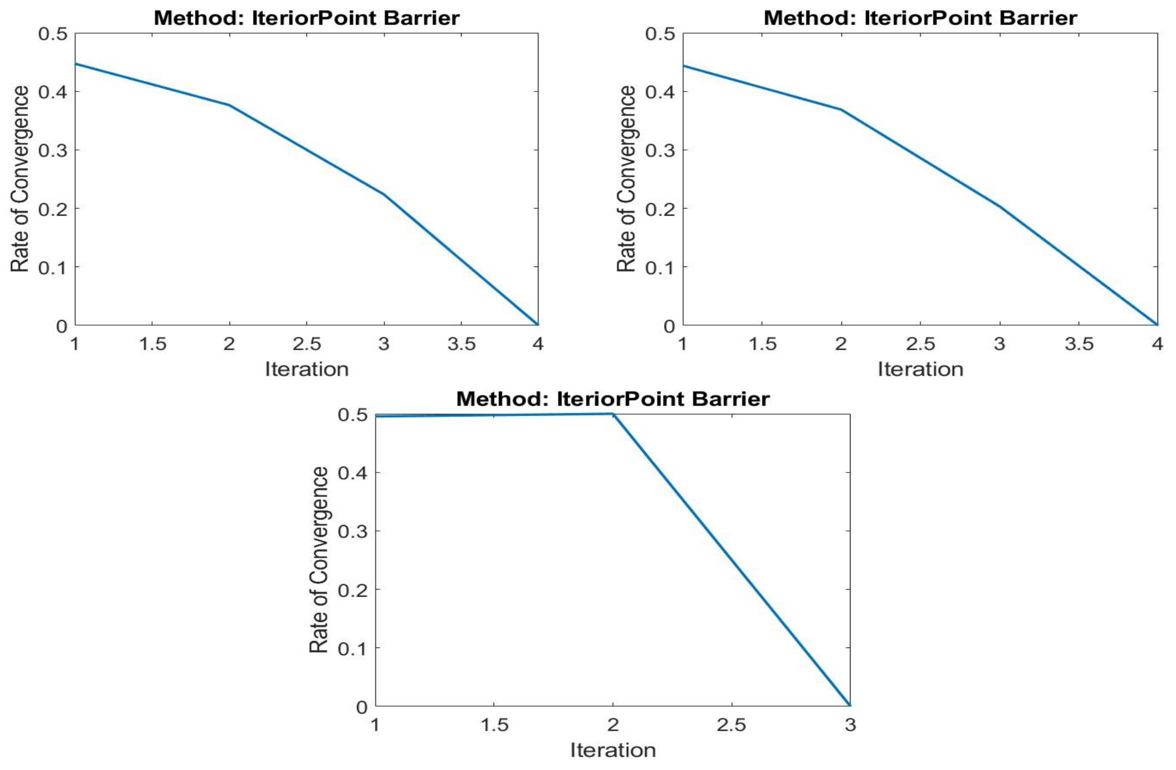

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.11 InteriorPoint Barrier with a polynomial kernel

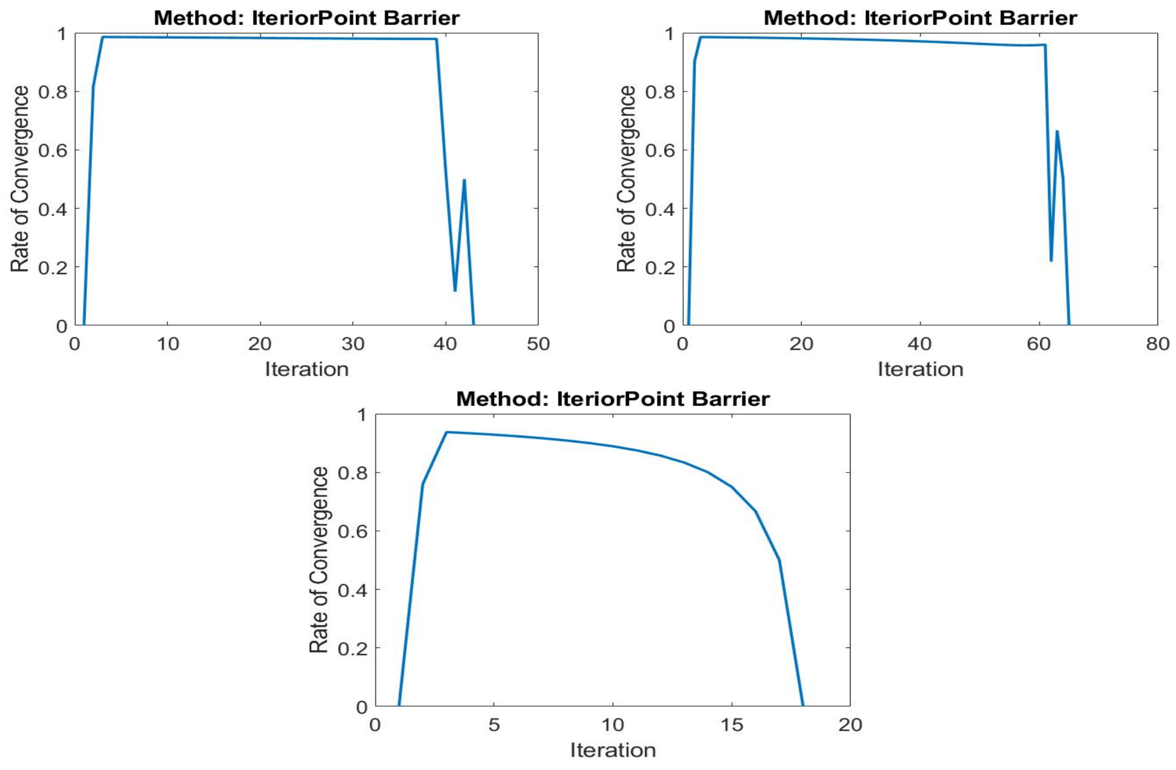

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.12 InteriorPoint Barrier with a linear RBF kernel

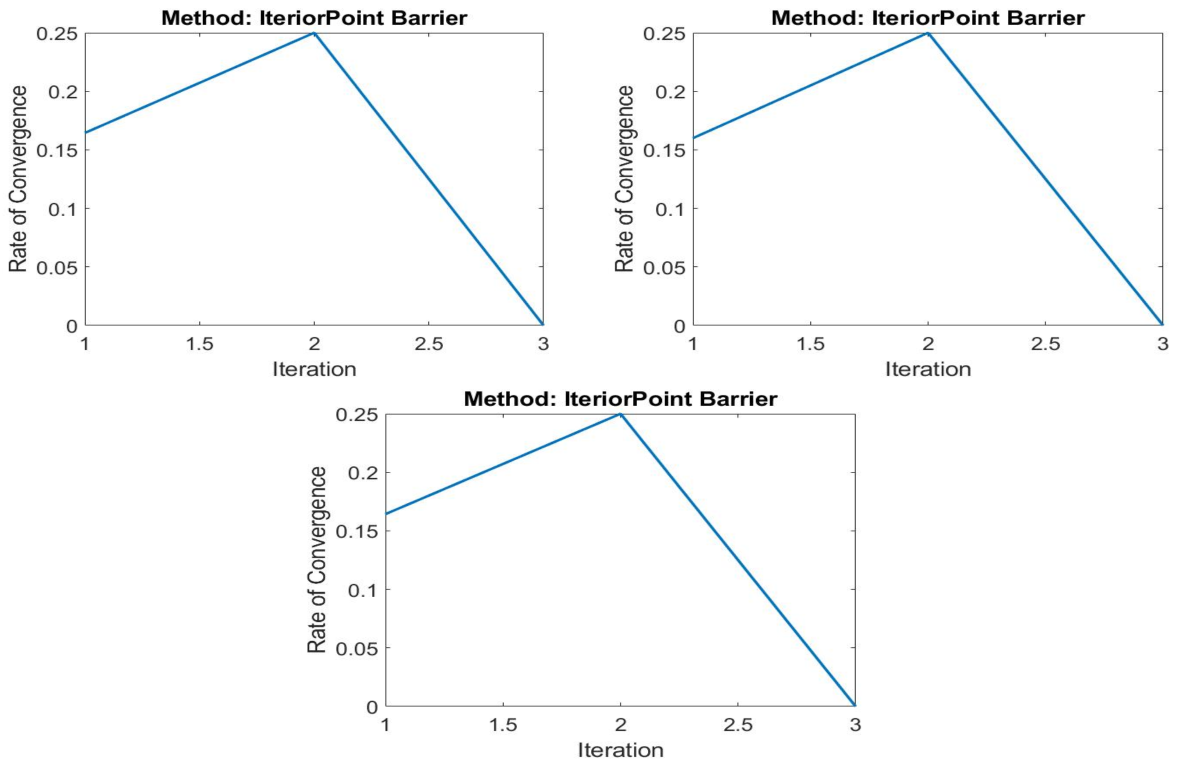

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

4.13 InteriorPoint Barrier with a sigmoid kernel

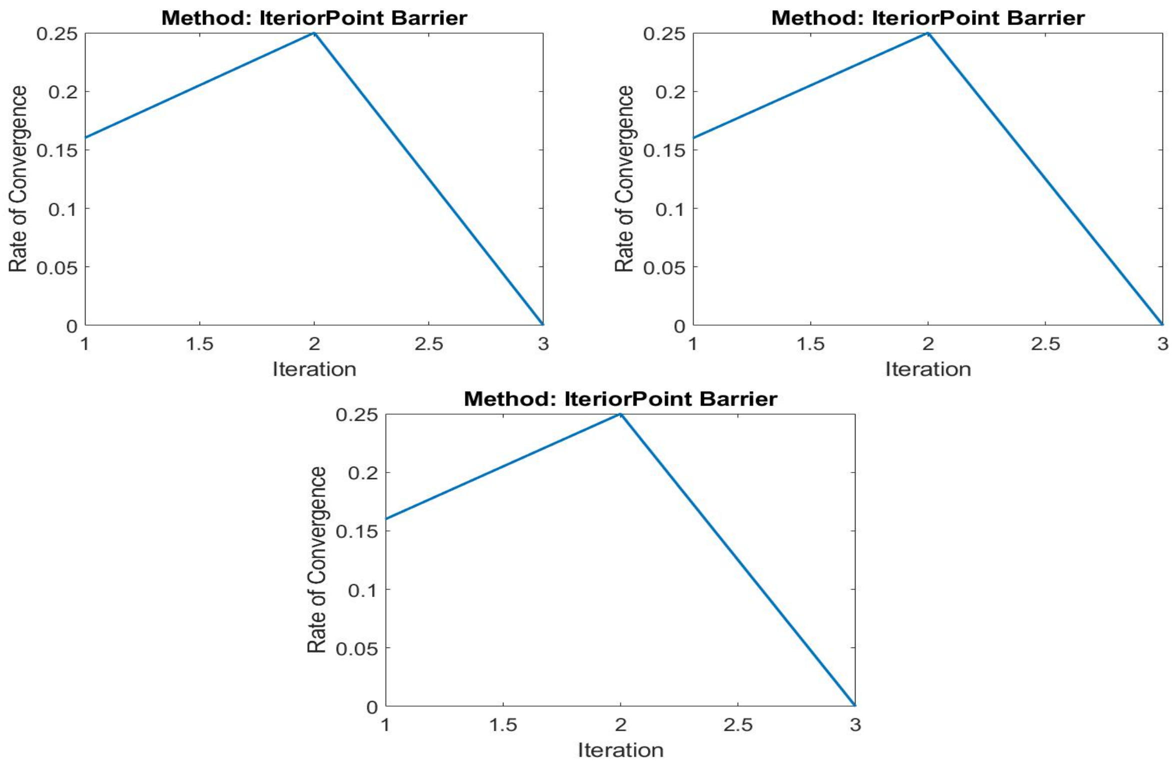

Fig. 2: it shows three different convergence rates graph for 3 different SVM classifiers

The accuracy obtained from different optimization methods with different kernels is shown below: note that the quadratic programming is the installed package. I put its result in the table just for the reference purpose.

Kernel	Quadratic Penalty	Augmented Lagrangian	InteriorPoint Barrier	Quadratic programming
Linear	73.333	73.333	73.333	93.3333
Polynomial	73.333	73.333	40	26.6667
RBF	86.6667	86.6667	76.6667	96.6667
Sigmoid	40	76.6667	26.6667	33.3333

According to the results showed in the above table, the best kernel is RBF due to high accuracy all the time. This also explains why RBF kernel is the most common kernel widely used in SVM. The Augmented Lagrangian method has the best performance throughout all kernels.

Reference

1. Andrew Ng, CS229 Lecture notes , Support vector machines, Part V.
2. Tristan Fletcher, Support Vector Machines Explained
3. Numerical Optimisation Constraint optimisation: Penalty and augmented Lagrangian methods, Lecture 14 (based on Nocedal, Wright)
4. Numerical Optimisation Constraint optimisation: Interior point methods, Lecture 14 (based on Boyd, Vanderberghe)