Robotics 811 - Homework 4

Xiang Zhi Tan

December 17, 2015

1 Q1

1(a)

Fristly, we notice that the equation is a separative form of differentiation and can be separate into

$$\frac{dy}{dx} = \frac{2}{x^2}(1-y)$$
$$(1-y)dy = \frac{2}{x^2}dx$$

By applying integration to both side, we get

$$\int (1-y)dy = \int \frac{2}{x^2}dx$$
$$y - \frac{1}{2}y^2 + c = -2x^{-1} + c$$

By observing the equation, we notice it has the form of a quadratic equation, we then rearrange it and put it into the form of the quadratic formula.

$$y - \frac{1}{2}y^2 + c = -2x^{-1} + c$$
$$-\frac{1}{2}y^2 + y + (2x^{-1} - c) = 0$$
$$y^2 + (-2y) + (-4x^{-1} + c) = 0$$
$$y = \frac{2 \pm \sqrt{4 - 4(-4x^{-1} + c)}}{2}$$
$$y = 1 \pm \frac{\sqrt{4 + 16x^{-1} - 4c}}{2}$$
$$y = 1 \pm \frac{\sqrt{4}\sqrt{1 + 4x^{-1} - c}}{2}$$
$$y = 1 \pm \sqrt{1 + 4x^{-1} - c}$$
$$y = 1 \pm \sqrt{4x^{-1} - c}$$

Because we know the initial condition of $y(1) = -1$, we can plug it back into the previous equation and solve for c

$$-1 = 1 \pm \sqrt{\frac{4}{1} - c}$$

Because the square root will be positive, the only way to have a negative on the lefthand side would be that the $\pm$ is a negative sign.

$$-1 = 1 - \sqrt{4 - c}$$
$$2 = \sqrt{4 - c}$$
$$c = 0$$

Now, we know that the analytic solution would be the following:

$$y(x) = 1 - 2\sqrt{x^{-1}}$$

1(b)

We implemented the Euler Method in the matlab script, **q1b.m**. Below is the result of the Method at each x with a step size of 0.05, x and the real y in table form.

n	x	y	real y
0	1.000000	-1.000000	-1.000000
1	0.950000	-1.050000	-1.051957
2	0.900000	-1.104050	-1.108185
3	0.850000	-1.162726	-1.169305
4	0.800000	-1.226723	-1.236068
5	0.750000	-1.296894	-1.309401
6	0.700000	-1.374293	-1.390457
7	0.650000	-1.460248	-1.480695
8	0.600000	-1.556452	-1.581989
9	0.550000	-1.665109	-1.696799
10	0.500000	-1.789149	-1.828427
11	0.450000	-1.932562	-1.981424
12	0.400000	-2.100956	-2.162278
13	0.350000	-2.302507	-2.380617
14	0.300000	-2.549691	-2.651484
15	0.250000	-2.862707	-3.000000
16	0.200000	-3.276924	-3.472136
17	0.150000	-3.861457	-4.163978
18	0.100000	-4.775677	-5.324555
19	0.050000	-6.507076	-7.944272
20	0.000000	-11.835382	-Inf

By observing the values, we notice this method is not too accurate and there are deviation in the end values.

1(c)

We implemented the fourth-order Runge-Kutta method in the matlab script, **q1c.m**. Below is the result of the Method at each x with a step size of 0.05, x and the real y in table form.

n	x	y	real y
0	1.000000	-1.000000	-1.000000
1	0.950000	-1.051957	-1.051957
2	0.900000	-1.108185	-1.108185
3	0.850000	-1.169305	-1.169305
4	0.800000	-1.236068	-1.236068
5	0.750000	-1.309401	-1.309401
6	0.700000	-1.390457	-1.390457
7	0.650000	-1.480695	-1.480695
8	0.600000	-1.581990	-1.581989
9	0.550000	-1.696800	-1.696799
10	0.500000	-1.828429	-1.828427
11	0.450000	-1.981426	-1.981424
12	0.400000	-2.162282	-2.162278
13	0.350000	-2.380624	-2.380617
14	0.300000	-2.651498	-2.651484
15	0.250000	-3.000033	-3.000000
16	0.200000	-3.472225	-3.472136
17	0.150000	-4.164292	-4.163978
18	0.100000	-5.326343	-5.324555
19	0.050000	-7.973392	-7.944272
20	0.000000	-183129202253624666683263156224.000000	-Inf

By observing the result, we observe that the values don't deviate until $n = 8$. Compared to Euler methods, fourth-order Runge-Kutta method is more accurate.

1(d)

We implemented the fourth-order Adams-Bashforth in the script, **q1d.m**. Below is the result of the Method at each x with a step size of 0.05, x and the real y in table form. Compared to the analytic solution, the results deviates faster as x approaches 0.

0	1.000000	-1.000000	-1.000000
1	0.950000	-1.051952	-1.051957
2	0.900000	-1.108174	-1.108185
3	0.850000	-1.169285	-1.169305
4	0.800000	-1.236038	-1.236068
5	0.750000	-1.309357	-1.309401
6	0.700000	-1.390393	-1.390457
7	0.650000	-1.480601	-1.480695
8	0.600000	-1.581854	-1.581989
9	0.550000	-1.696602	-1.696799
10	0.500000	-1.828133	-1.828427
11	0.450000	-1.980974	-1.981424
12	0.400000	-2.161562	-2.162278
13	0.350000	-2.379424	-2.380617
14	0.300000	-2.649367	-2.651484
15	0.250000	-2.995918	-3.000000
16	0.200000	-3.463310	-3.472136
17	0.150000	-4.141388	-4.163978
18	0.100000	-5.248560	-5.324555
19	0.050000	-7.504348	-7.944272
20	0.000000	-15.471548	-Inf

By observing the result, we see that the fourth-order Adams-Bashforth is more accurate than Euler's method but less accurate then fourth-order Runge-Kutta method.

2 Q2

2(a)

Firstly, we plotted the function in matlab using the *contour* function on the range, $y = [-4, 2]$ and $x = [-2, 4]$. Following is the result of the contour plot

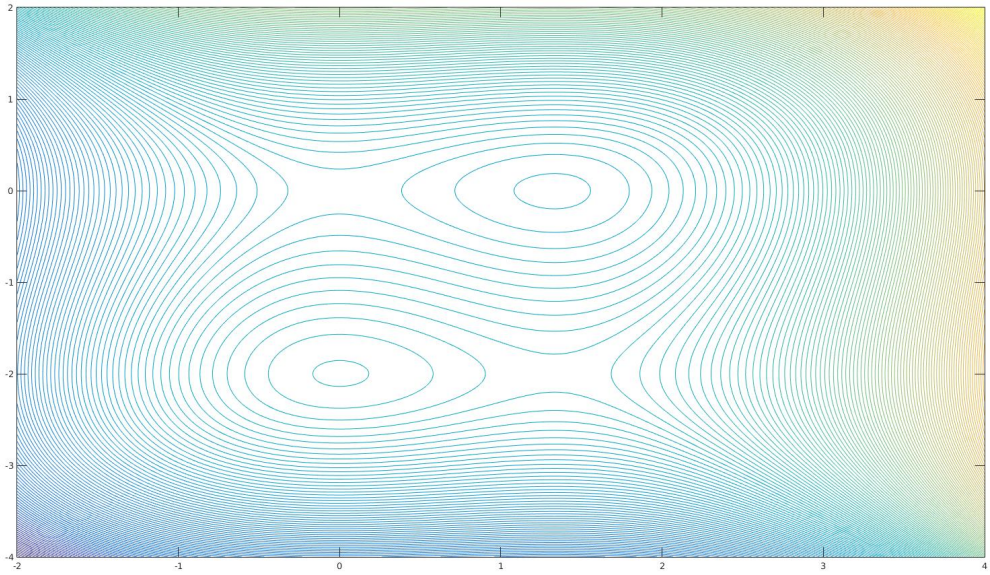

Figure 1: plot of the function $f(x, y) = x^3 + y^3 - 2x^2 + 3y^2 - 8$

By examining the sketch created, we notice there are four critical points, $(0, -2), (0, 0), (\frac{4}{3}, -2), (\frac{4}{3}, 0)$. We then measure the gradients(in the domain space) of the nearby points to determine the type of critical point. Following are the finding for each point:

1. Point$(0, -2)$
   This is a local maxima. By measuring the gradient at 8 points around the point with a step size of 0.1 relative to this point. we found them all to have a position gradient which means they are moving upwards towards $(0, -2)$ which means this is a local maximum.

2. Point$(0, 0)$
   This is a saddle points. When measuring the gradient at 8 points around the point with a step size of 0.1 relative to this point, we measured a positive gradient at the two points of $(0.1, 0)$ and $(-0.1, 0)$ and the remaining 6 points to have a negative gradient. This fits the definition of a saddle point.

3. Point$(\frac{4}{3}, 0)$
   This is the local minima. When measuring the gradient at the 8 points around the point with the step size of 0.1 relative to this point, all gradients are negative, this means that the direction of all these points are towards this point, meaning that this point must serve as the local minima.

4. Point$(\frac{4}{3}, -2)$
   This is a saddle points. When measuring the gradient, we found a negative gradient at points

$(1.23333, -2)$ and $(1.43333, -2)$ and positive gradient at all remaining points. This shows that this must be a saddle point.

2(b)

The gradient(partial derivative) of x and y can be found with the following equation:

$$\frac{\partial}{\partial x} = x(3x - 4)$$
$$\frac{\partial}{\partial y} = y(3y + 6)$$

Using the steepest descent algorithm, we first calculate the gradient at point $(1, -1)$. Which gives us the gradient at x, $\frac{\partial}{\partial x} = -1$ and gradient at y, $\frac{\partial}{\partial y} = -3$. Since the gradient are both non zero, we try to minimize the function $g(t) = f(x + t\vec{u})$ with u being the gradient. Minimizing the function, we found $t = \frac{1}{3}$. We update the points using the following algorithm $x^{n+1} = x^n - t\vec{u}$. After updating the points, we found the points to be $(\frac{4}{3}, 0)$ which is one of the critical points(also the local minima). The gradient at that point is also 0, stopping the algorithm. Therefore, we need only one step to converge to the local minimum, if we start steepest descent at point $(1, -1)$.

3 Q3

3(a)

First, we know that eigenvectors of Q(real symmetric positive definite matrix) are orthogonal(proof of this could be easily found online or derived). This means that given two eigenvectors, v_1 and v_2 come from two distinct eigenvalues, λ_1 and λ_2, their inner product would be 0. By grinding through the equations, we could derive the Q-orthogonal definition.

$$\begin{aligned}
&< v_1, v_2 > = 0 && \\
&< \lambda_1 v_1, v2 > = 0 && \lambda \text{ is a scalar and will not change the result of the inner product} \\
&< Qv_1, v_2 > = 0 && \\
&(Qv_1)^T v_2 = 0 && \\
&v_1^T Q^T v_2 = 0 && Q^T = Q \\
&v_1^T Q v_2 = 0 && \text{the definition of Q-orthogonal}
\end{aligned}$$

3(b)

The solution in the previous section would be sufficient as it is based on the fact that eigenvectors of Q are orthogonal to each other, which is now explicitly given.

4 Q4

I discussed with Rick Goldstein on parts of the problem

4(a)

From the algorithm in the notes on page 17, we know that:

$$d_k = -g_k + B_{k-1} d_{k-1} \tag{1}$$

As we want to show $d_k^T Q d_k = -d_k^T Q g_k$, we will derive the right-hand side from the left using equation 1.

$$\begin{aligned} d_k^T Q d_k &= d_k^T Q(-g_k + B_{k-1} d_{k-1}) && \text{,inserted 1} \\ &= -d_k^T Q g_k + B_{k-1} d_k^T Q d_{k-1} && \text{, } d_k \text{ and } d_{k-1} \text{ are Q-orthogonal, } d_k^T Q d_{k-1} = 0 \\ &= -d_k^T Q g_k \end{aligned} \tag{2}$$

This shows that $d_k^T Q d_k = -d_k^T Q g_k$. Now we will show that we can obtain x_{k+1} with only Q to find g_k and Qg_k. First is obtaining d_k.

$$\begin{aligned} d_k &= -g_k + B_{k-1} d_{k-1} \\ &= -g_k + \frac{g_k^T Q d_{k-1}}{d_{k-1}^T Q d_{k-1}} d_{k-1} \\ &= -g_k - \frac{d_{k-1}^T Q g_k}{d_{k-1}^T Q g_{k-1}} d_{k-1} && \text{, rearrange and apply 2} \end{aligned} \tag{3}$$

Obtaining d_k now only depends on d_{k-1}, which is obtain in the previous iterative or given if it's the first, Qg_{k-1} and g_k, which is what we want.
Now we will see how to obtain α_k

$$\begin{aligned} \alpha_k &= -\frac{g_k^T d_k}{d_k^T Q d_k} \\ \alpha_k &= \frac{g_k^T d_k}{d_k^T Q g_k} && \text{, apply 2} \end{aligned} \tag{4}$$

Again, obtaining α_k would only need variables that we already calculated. To get $x_k + 1$ would just be $x_{k+1} = x_k + \alpha_k d_k$. Which both have been shown to be able to be derived using only Q to evaluate g_k and Qg_k.

4(b)

We will derive the desire solution using the fact that $y_k = x_k - g_k$ and $p_k = \Delta f(y_k)$.

$$\begin{aligned} p_k &= \Delta f(y_k) \\ &= Q(y_k) + b && \text{wrote the gradient in terms of equation} \\ &= Q(x_k - g_k) + b \\ &= Qx_k - Qg_k + b \\ &= Qx_k + b - Qg_k \\ &= g_k - Qg_k && \text{, } Qx_k + b \text{ is the gradient at the point } x_k \text{, therefore can be written as } g_k \\ Qg_k &= g_k - p_k \end{aligned} \tag{5}$$

4(c)

First, we will modified some terms in the original algorithm in the notes on page 17 using the equations 2 and 5.
For equation 2(a) in the notes which has already been modified in 4

$$\begin{aligned} \alpha_k &= \frac{g_k^T d_k}{d_k^T Q g_k} \\ \alpha_k &= \frac{g_k^T d_k}{d_k^T (g_k - p_k)} && \text{,applied 5} \end{aligned}$$

For equation 2(c) in the notes.

$$\begin{aligned} B_k &= \frac{g_{k+1}^T Q d_k}{d_k^T Q g_k} \\ &= \frac{d_k^T Q g_{k+1}}{d_k^T Q g_k} \\ &= \frac{d_k(g_{k+1} - p_{k+1})}{d_k^T(g_k - p_k)} \quad \text{,applied 5} \end{aligned} \tag{6}$$

The final general algorithm will be similar to the one in the notes on page 17.

Result: Minimum of the equation
Let $d_0 = -g_0$;
for $k = 0, ..., n-1$ **do**
  $\alpha_k = \frac{g_k^T d_k}{d_k^T(g_k - p_k)}$;
  $x_{k+1} = x_k + \alpha_k d_k$;
  $B_k = \frac{d_k(g_{k+1} - p_{k+1})}{d_k^T(g_k - p_k)}$;
  $d_{k+1} = -g_{k+1} + B_k d_k$
end
Return X_n;

Algorithm 1: Conjugate Gradient Method

Note, the gradient of each point can be obtain by sampling the surrounding points

5 Q5

The order of the direction searched does not matter. This is because Q is a symmetric positive definite matrix and this makes the $d_0, d_1, ..., d_k$ vectors Q-orthogonal and linearly independent(proof on page 14 of notes).This mean each individual d are independent and does not depend on prior directions. As k increases in the search, it will slowly search a larger spam subspace and in the end find the minimum of the whole function space at $k = n - 1$. The order doesn't matter as any permutation of the search will eventually give the same result. Furthermore, we will only need to search each direction onces. This is because of the Expanding subspace theorem(in notes page 20) that states that for all x in the sequence x_k generated by the following rule

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k d_k \\ \text{where } \alpha_k &= -\frac{g_k^T d_k}{d_k^T Q d_k} \\ g_k &= Q X_k + b \end{aligned}$$

The generated x_{k+1} will be the minimum of the line $x_{k+1} + \alpha_k d_k$. This means that at one search along the line, it will find the minimum in that space. Furthermore, the Expanding subspace theorem also minimizes on the affine variety, $x_0 + B_{k+1}$. The $x_k + 1$ we find is the minimum of the span of all the previous direction$(d_0, d_1, ...d_k)$ as well, means that there are no other point in that subspace that could be smaller than the found $x_k + 1$. Therefore, we will only need to search the direction once for each direction.

6 Q6

As we are finding the maximum area with a given parameter,p, we know that the objective function must be

$$f(x, y) = xy$$

with the constraint of

$$2y + 2x - p = 0$$

First we construct the Lagrangian

$$L(x, y, \alpha) = xy + \alpha(2y + 2x - p)$$

We then compute the gradient at set it to 0

$$\Delta L(x, y, \alpha) = \begin{pmatrix} y + 2\alpha \\ x + 2\alpha \\ 2x + 2y - p \end{pmatrix} = \vec{0}$$

We are now given 3 equations

$$y = 2\alpha \tag{7}$$

$$x = 2\alpha \tag{8}$$

$$\begin{aligned} 2x &= -2y + p \\ x &= \frac{-2y + p}{2} \end{aligned} \tag{9}$$

Insert 9 into 8, we get

$$2\alpha = \frac{-2y + p}{2} \tag{10}$$

Then, insert 10 into 7, we can solve y

$$\begin{aligned} y &= \frac{-2y + p}{2} \\ 2y &= -2y + p \\ y &= \frac{p}{4} \end{aligned} \tag{11}$$

We then insert 11 back into 8 and we solve x

$$x = \frac{p}{4} \tag{12}$$

We have now found the critical points in the equation, where $x = \frac{p}{4}$ and $y = \frac{p}{4}$. Notice that $x = y$. To show that we are achieving the maximum, we need to verify the second order sufficiency where we need to show the following matrix

$$L(x^*) = \Delta^2 f(x*) + \alpha^T \Delta^2 h(x^*) \tag{13}$$

is negative semi-definate on m which is $m = y|\Delta h(x^*)y = 0$. First we solve the partial derivatives for $f(x)$ and $h(x)$

$$\begin{aligned} \frac{\partial}{\partial^2 x} &= 0 \\ \frac{\partial}{\partial^2 y} &= 0 \\ \frac{\partial}{\partial xy} &= 1 \\ \frac{\partial}{\partial yx} &= 1 \\ \Delta^2 h &= 0 \end{aligned}$$

We have solved $L(x^*)$

$$L(x^*) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

We then check where the matrix is positive semi-definite or negative semi-definite by choosing a vector from the subspace M and apply $y^T L y$.

$$\begin{aligned} \Delta h(x) &= [22] \\ [22]y &= 0 \\ \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} &= \begin{pmatrix} -1 \\ 1 \end{pmatrix} \end{aligned} \tag{14}$$

$$\begin{pmatrix} -1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = -2 \tag{15}$$

This shows the matrix is negative semi-definite and therefore, x^* must be the maximum of f. Meaning $x = \frac{p}{4}$ and $y = \frac{p}{4}$ will give the largest area given the parameter, p

7 Q7

7(a)

We can first rearrange the constraints 2 and 3 into the following form by moving elements on the right handside to the left and multiplying constraints 2 by -1.

$$\begin{aligned} -w^T x_i - b + 1 - \xi_i \leq 0 \\ \text{if } y_i = 1 w^T x_i + b + 1 - \xi_i \leq 0 \\ \text{if } y_i = -1 \end{aligned}$$

Through observing the inequalities, we notice that the only difference is the element $w^T x_i$ which has a different sign that could be change by the values of y_i. This allows us to combine both inequalities into the following constraint.

$$-y_i(w^T x_i + b) + 1 - \xi_i \leq = 0$$

7(b)

The Lagrangian for our optimization problem is as following:

$$L([w\ 0\ \xi]^T, \alpha, \beta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{l} \xi_i + \sum_{i=1}^{l} \alpha_i g_i([w\ b\ \xi]^T) - \sum_{i=1}^{l} \beta_i \xi_i$$

7(c)

Following is the steps to minimize the Lagrangian which is ther primal form of the SVM. Here's the original Lagrangian.

$$L([w\ 0\ \xi]^T, \alpha, \beta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{l} \xi_i + \sum_{i=1}^{l} \alpha_i(-y_i(w^T x_i + b) + 1 - \xi_i) - \sum_{i=1}^{l} \beta_i \xi_i \tag{16}$$

First, we find the partial of w,b and ξ_i. Following are the partials follow by setting them to 0.
Finding partial of w

$$\begin{aligned} \frac{\partial}{\partial w} &= \frac{2}{2} w - \sum_{i=1}^{l} (\alpha_i y_i x_i) = 0 \\ w &= \sum_{i=1}^{l} (\alpha_i y_i x_i) \end{aligned} \tag{17}$$

Finding the partial of b.

$$\begin{aligned} \frac{\partial}{\partial b} &= -\sum_{i=1}^{l} (\alpha_i y_i) = 0 \\ \sum_{i=1}^{l} (\alpha_i y_i) &= 0 \end{aligned} \tag{18}$$

Finding the partial of xi_i.

$$\begin{aligned}\frac{\partial}{\partial \xi_i} &= C - \sum_{i=1}^{l}(\alpha_i) - \sum_{i=1}^{l}(\beta_i) = 0 \\ C &= \sum_{i=1}^{l}(\alpha_i) + \sum_{i=1}^{l}(\beta_i)\end{aligned} \tag{19}$$

Now we insert equations 17, 19 in equation 16.

$$\begin{aligned}L &= \frac{1}{2}(\sum_{i=1}^{l}(\alpha_i y_i x_i))^2 + (\sum_{i=1}^{l}(\alpha_i) + \sum_{i=1}^{l}(\beta_i))\sum_{i=1}^{l}\xi_i + \sum_{i=1}^{l}\alpha_i(-y_i((\sum_{j=1}^{l}(\alpha_j y_j x_j))x_i + b) + 1 - \xi_i) - \sum_{i=1}^{l}\beta_i\xi_i \\ L &= \frac{1}{2}(\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i\alpha_j y_i y_j x_j^T x_i)) + \sum_{i=1}^{l}(\alpha_i\xi_i) + \sum_{i=1}^{l}(\beta_i\xi_i) + \sum_{i=1}^{l}(\alpha_i\alpha_j y_i y_j x_j^T x_i)) \\ &\quad - b\sum_{i=1}^{l}(\alpha_i y_i) + \sum_{i=1}^{l}(\alpha_i) - \sum_{i=1}^{l}(\alpha_i\xi_i) - \sum_{i=1}^{l}\beta_i\xi_i \\ L &= \sum_{i=1}^{l}(\alpha_i) - \frac{1}{2}(\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i\alpha_j y_i y_j x_j^T x_i)) - b\sum_{i=1}^{l}(\alpha_i y_i)\end{aligned} \tag{20}$$

By applying equation 18 in the previous equation, we can derive the final form

$$L^*(\alpha) = \sum_{i=1}^{l}(\alpha_i) - \frac{1}{2}(\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i\alpha_j y_i y_j x_j^T x_i)) \tag{21}$$

7(d)

As we want to write the equation in terms of H and f such that $L^*(\alpha) = \frac{1}{2}\alpha^T H\alpha + f^T\alpha$. By looking at equation 21, we see a similar structure between them. For the first part $\frac{1}{2}\alpha^T H\alpha$

$$\begin{aligned}\frac{1}{2}\alpha^T H\alpha &= \frac{1}{2}(\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i\alpha_j y_i y_j x_j^T x_i)) \\ H &= (\sum_{i=1}^{l}\sum_{j=1}^{l}(y_i y_j x_j^T x_i))\end{aligned}$$

For second part $f^T\alpha$

$$\begin{aligned}f^T\alpha &= \sum_{i=1}^{l}(\alpha_i) \\ f^T &= [1, 1, ...1]\end{aligned}$$

7(e)(i)

We implemented the SVM in the matlab file **svmSolver.m** and the solver can be tested and visualized by the files **q7e_1.m** for the first part and **q7e_2.m** for the second part of the question. Following are graphs of using the SVM solver on the test data and the decision boundary and margins for C=1,0.1,0.01,0.001

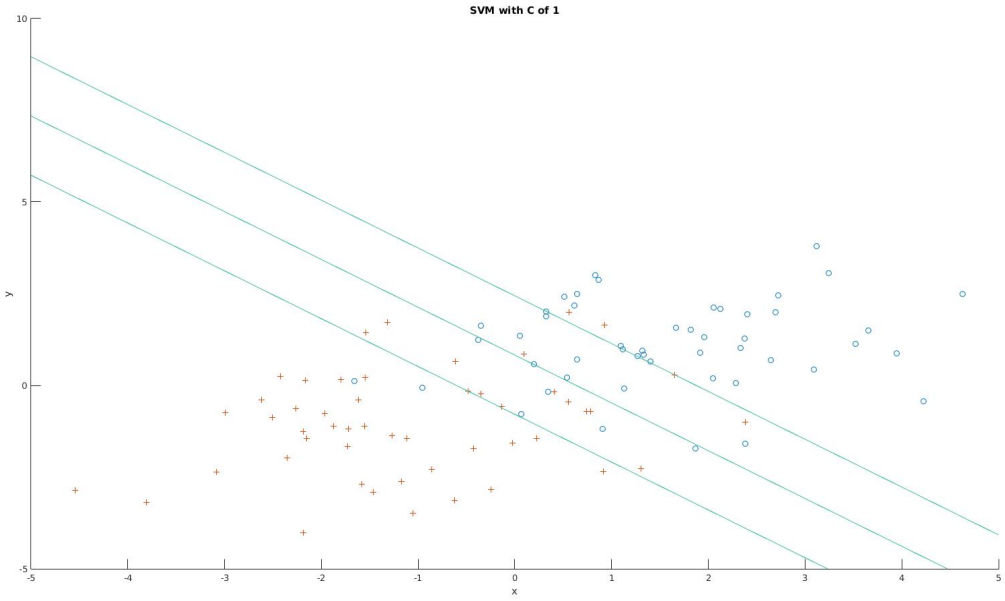

Figure 2: Plot of decision boundary and margins with C=1

The margins are really small and cannot be observed in the graph.

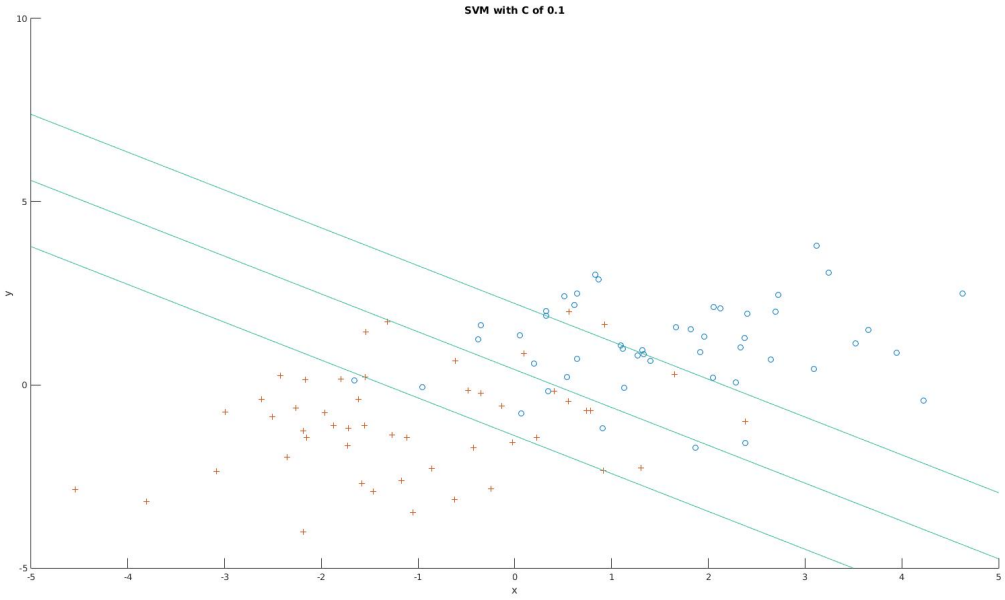

Figure 3: Plot of decision boundary and margins with C=0.1

The margins are really small and cannot be observed in the graph.

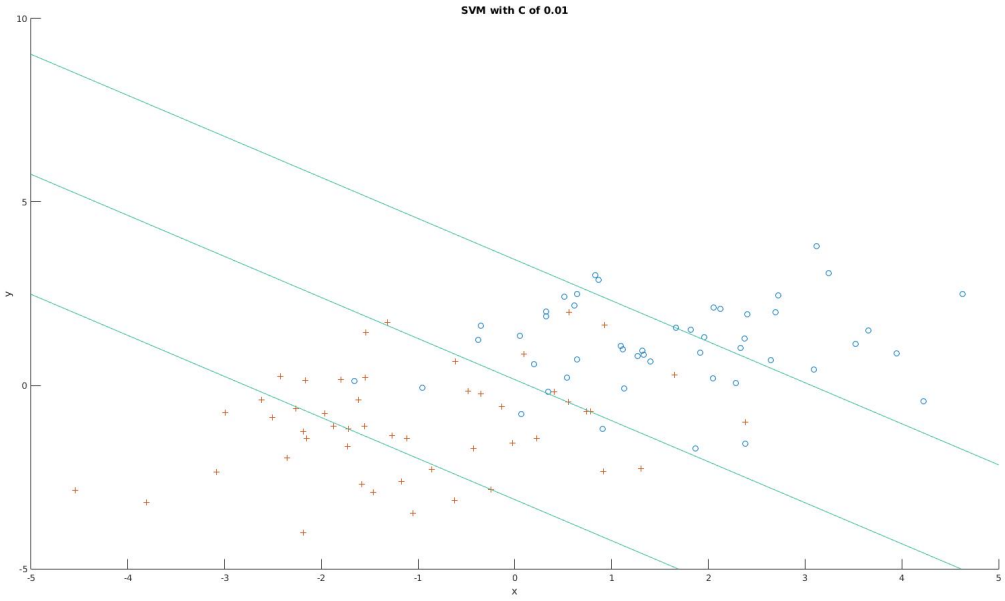

Figure 4: Plot of decision boundary and margins with C=0.01

The margins are small but you can still see the slight increase in width compare to the previous two figures.

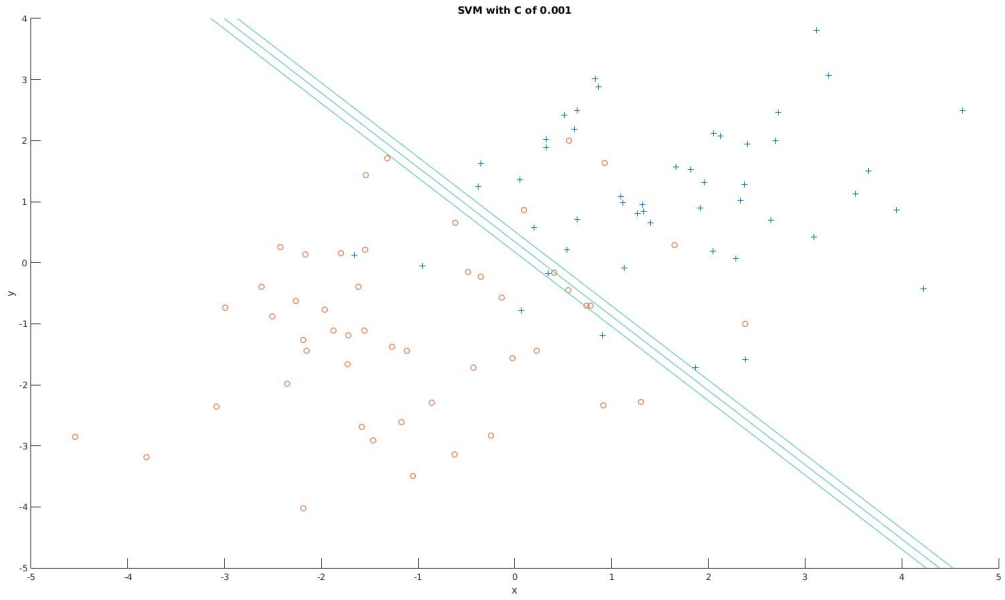

Figure 5: Plot of decision boundary and margins with C=0.001

You can clearly see decision boundary's and it's margin in the previous figure.

7(e)(ii)

As observed in the previous few figures, we see that the decision margins increase as C decreases. Therefore, as C approaches 0, the sampling errors are negated.

$$\lim C \to 0 min \frac{1}{2}||w||^2 + C\sum_{i=1}^{l}\xi_i = min\frac{1}{2}||w||^2 \tag{22}$$

Therefore, SVM will now ignore the sampling error and might choose points that have significant error as support vectors and give the wrong result

7(e)(iii)

This part of the question is implemented in the script,**q7e_2.m**. Following is the table showing the accuracy of my algorithm on a 10-fold cross validation run.

k	correct P	wrongly label	P detection %	correct N	wrongly label	N detection %	total accuracy
1	49	0	0.326667	104	101	1	0.602362
2	70	0	0.526316	120	63	1	0.750988
3	68	0	0.539683	128	58	1	0.771654
4	63	0	0.463235	117	73	1	0.711462
5	52	0	0.382353	118	84	1	0.669291
6	75	0	0.539568	114	64	1	0.747036
7	70	0	0.514706	117	66	1	0.739130
8	75	0	0.547445	116	62	1	0.754941
9	65	0	0.481481	118	70	1	0.723320
10	78	0	0.609375	126	50	1	0.803150

One interesting observation is that the SVM over classifies the negative(the hand writing of zero). It has 100% accuracy on classifying that type of character but it also falsely classified large number(range from $60\% - 30\%$) of ones as zeros

7(e)(iv)

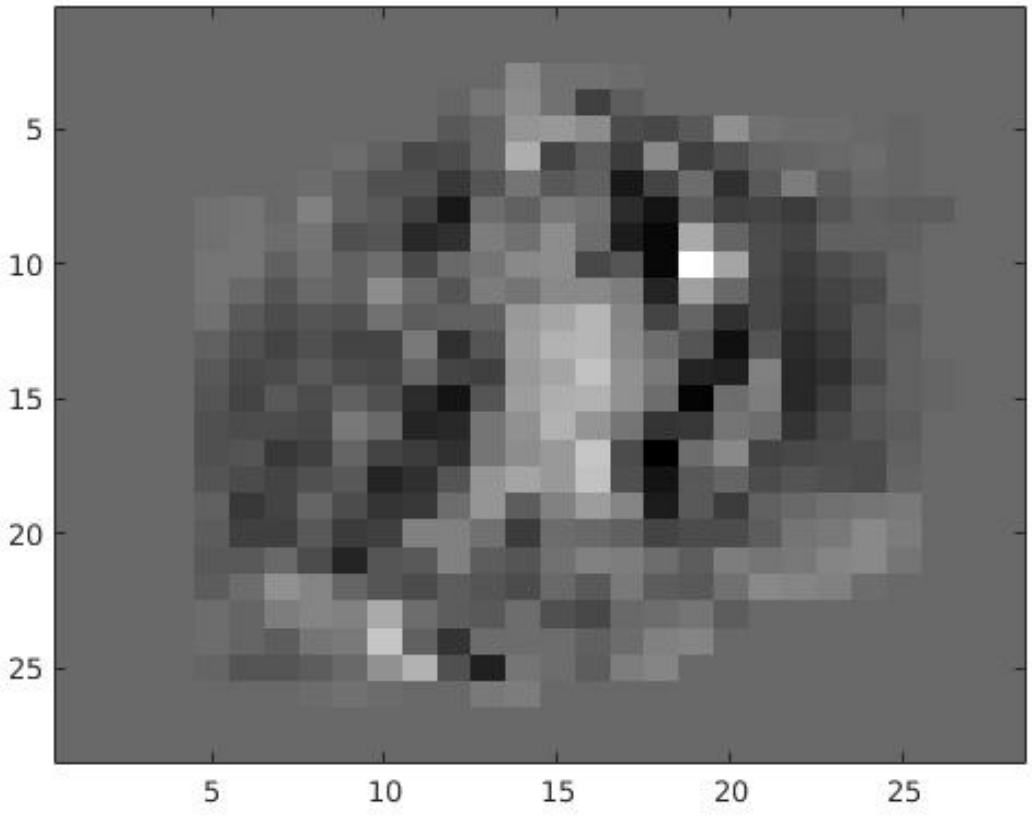

Figure 6: plot of weight function

The weight functions shows us what the algorithm learns. It assigns positive number when it detects features in the middle and assigns negative values to features around in the circle. One possibility that hand writing of zeros has a higher accuracy than hand writing of ones is that most of the zeros' feature space does not overlap with ones and it also has a larger area to be compared with. Where as ones, it relies on the middle and also the edge which it shares with zero(you can see the slight discoloration on those are compare to the side of the zero). The selection of the feature space is really important and the larger size of zero in the feature space has skewed the result towards it.