HOMEWORK#4

Name: Yiming Tong
GTID: 903541553

Problem 1: SVM.

1. Explain why can we set the margin c = 1 in SVM formulation?

   The original optimization problem is:

$$\max_{w,b} \frac{2c}{\|w\|}$$
$$s.t.\ y^i\left(w^\top x^i + b\right) \geq c, \forall i$$

   We can see that the magnitude of c merely scales w and b, and does not change the relative goodness of different classifiers. So, we set the margin c=1 to get a cleaner problem.

2. What does this imply in terms of how to relate data to w?
   Based on the formula, we can see that the optimal w is a linear combination of a small number of data points whose a_i is not equal to 0. This"sparse" representation can be viewed as data compression

3. Explain why only the data points on the "margin" will contribute to the sum above, i.e., playing a role in defining w.

   Apply Lagrangian multiplier to solve following optimization problem:

$$\min_{w,b} \frac{1}{2} w^\top w$$
$$s.t.\ 1 - y^i\left(w^\top x^i + b\right) \leq 0, \forall i$$

   The lagrangian function :

$$L(w, b, \alpha) = \frac{1}{2} w^\top w + \sum_{i=1}^{m} \alpha_i \left(1 - y^i\left(w^\top x^i + b\right)\right)$$

   Taking derivative and set to zero:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} \alpha_i y^i x^i = 0$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y^i = 0$$

So,

$$w = \sum a_i \, y^i x^i$$

According to KTT conditions:

$$a_i \geq 0$$

$$a_i g_i(w) = 0$$

That is:

$$a_i \left(1 - y^i \left(w^T x^i + b\right)\right) = 0$$

So, for data points with $1 - y^i\left(w^T x^i + b\right) > 0$, a_i must be equal to zero, which indicates that it won't influence w. And only data points with $1 - y^i\left(w^T x^i + b\right) = 0$ will have positive a_i and can influence w. That means only data points with $w^T x^i + b = \pm 1$, playing a role in defining w. And according to definition, data points with $w^T x^i + b = \pm$ is on the"margin".

Problem 2: Neural networks

1. Consider a neural networks for a binary classification using sigmoid function for each unit. If the network has no hidden layer, explain why the model is equivalent to logistic regression.

As for this network without hidden layer, there will be only two steps between input and output. The first step is to combine all the inputs linearly using weight vector. The second step is to transform the result of the first step using sigmoid function. Assuming w is the weight vector, x is the input. Then the output will be:

$$: \frac{1}{1 + \exp(-w^\top x)}$$

Obviously, the model is equivalent to logistic regression.

2. Also find the gradient of l with respect to α and β.

Using chain rule of derivatives:
Given that:

$$l(w,\alpha,\beta) := \sum_{i=1}^{n} \left(y^i - \sigma\left(w^\top z^i\right)\right)^2$$
$$\text{where } z_1^i = \sigma\left(\alpha^\top x^i\right), z_2^i = \sigma(\beta^\top x^i)$$

Let $u^i = w^T z^i$

$v^i = a^T x^i$

$\tau^i = \beta^T x^i$

Gradient:

$$\frac{\partial l(w,\alpha,\beta)}{\partial \alpha} = \frac{\partial l(w,\alpha,\beta)}{\partial z_1^i}\frac{\partial z_1^i}{\partial \alpha}$$

$$= \sum_i 2(y^i - \sigma(u^i))\sigma(u^i)(1-\sigma(u^i))w_1\sigma(v^i)(1-\sigma(v^i))x^i$$

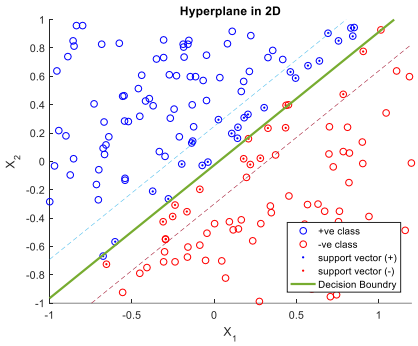
$$\frac{\partial l(w,\alpha,\beta)}{\partial \alpha} = \frac{\partial l(w,\alpha,\beta)}{\partial z_2^i}\frac{\partial z_2^i}{\partial \beta}$$

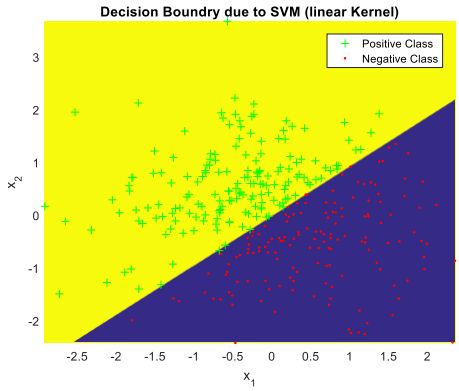
$$= \sum_i 2(y^i - \sigma(u^i))\sigma(u^i)(1-\sigma(u^i))w_2\sigma(\tau^i)(1-\sigma(\tau^i))x^i$$

Problem 3:

```
w1=[(alp_old.*Y).*X(:,1),(alp_old.*Y).*X(:,2)];
```

w is a n*2 vector in this problem.





```
W

W =

   -7.5016    7.6700

Optimal Objective function value 0.5*W^2

ans =

   57.5517

Elapsed time is 36.345308 seconds.
```

Sub Gradient descent method for the primary problem with constraints:

General constraints [edit]

The subgradient method can be extended to solve the inequality constrained problem

minimize $f_0(x)$ subject to
$f_i(x) \leq 0, \quad i = 1, \ldots, m$

where f_i are convex. The algorithm takes the same form as the unconstrained case

$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$

where $\alpha_k > 0$ is a step size, and $g^{(k)}$ is a subgradient of the objective or one of the constraint functions at x. Take

$$g^{(k)} = \begin{cases} \partial f_0(x) & \text{if } f_i(x) \leq 0 \ \forall i = 1 \ldots m \\ \partial f_j(x) & \text{for some } j \text{ such that } f_j(x) > 0 \end{cases}$$

where ∂f denotes the subdifferential of f. If the current point is feasible, the algorithm uses an objective subgradient; if the current point is infeasible, the algorithm chooses a subgradient of any violated constraint.

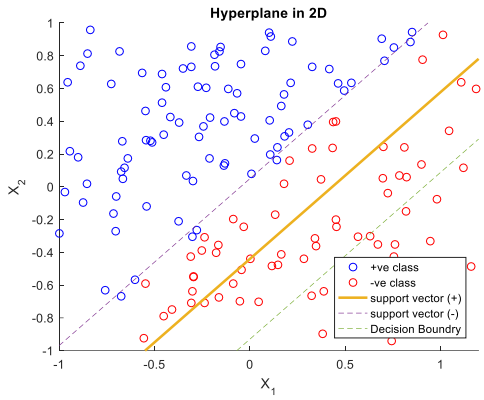
1) Sub Gradient descent for primary problem:

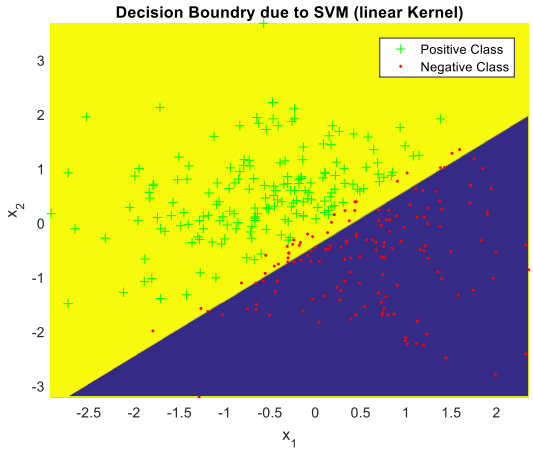
$w_{(k+1)} = w_{(k)} - c * g(k)$
$g(k)$ depends on the $f_0(x)$ and $f_i(x)$ for all $i=1,\cdots m$

From the above algorithm we know, we need to do gradient descending to meet all constraints, then do the optimization.
The core codes are here:

```
while norm1>tol

   total_iterations=total_iterations+1;
   w1 = w2;
   ttt = 0;
   for i=1:N
      if (1-w1*X(i,:)'-b)*Y(i)>0
         w2 = w1-(-s*X(i,:)*Y(i));
         b = b-(-s*Y(i));
         ttt = 1;
         break;
      end
   end
   if ttt==0
      w2 = w1-2*s*w1;
   end
   norm1=norm(w2-w1);
   display(norm1);
   if total_iterations==10000
      break;
   end
```

If all the constraints are met(ttt=0), then we can do the objective function's iteration.
The result is here:






```
W

W =

  -4.9959    4.8553

Optimal Objective function value 0.5*W^2

ans =

   24.2663

Elapsed time is 16.093495 seconds.
```

2) Stochastic Sub Gradient descent for primary problem:

2 Stochastic subgradient method

The stochastic subgradient method is essentially the subgradient method, but using noisy subgradients and a more limited set of step size rules. In this context, the slow convergence of subgradient methods helps us, since the many steps help 'average out' the statistical errors in the subgradient evaluations.

We'll consider the simplest case, unconstrained minimization of a convex function $f : \mathbf{R}^n \to \mathbf{R}$. The stochastic subgradient method uses the standard update

$$x^{(k+1)} = x^{(k)} - \alpha_k \tilde{g}^{(k)},$$

where $x^{(k)}$ is the kth iterate, $\alpha_k > 0$ is the kth step size, and $\tilde{g}^{(k)}$ is a noisy subgradient of f at $x^{(k)}$,

$$\mathbf{E}(\tilde{g}^{(k)}|x^{(k)}) = g^{(k)} \in \partial f(x^{(k)}).$$

Even more so than with the ordinary subgradient method, we can have $f(x^{(k)})$ increase during the algorithm, so we keep track of the best point found so far, and the associated function value

$$f_{\text{best}}^{(k)} = \min\{f(x^{(1)}), \ldots, f(x^{(k)})\}.$$

The sequences $x^{(k)}$, $\tilde{g}^{(k)}$, and $f_{\text{best}}^{(k)}$ are, of course, stochastic processes.

For this question, each time we need to calculate the g(k),in a stochastic way.

What I do to implement the stochastic question is to introduce a Gaussian random number which give each w a noise(to avoid local optimization):

```
for i=1:N
   if (1-w1*X(i,:)'-b)*Y(i)>0
      w2 = w1-(-s*X(i,:)*Y(i));
      b = b-(-s*Y(i));
      w2 = (1+normrnd(0,0.1))*w2;
      b = (1+normrnd(0,0.1))*b;
      ttt = 1;
      break;
   end
end
if ttt==0
   w2 = w1-2*s*w1;
   w2 = (1+normrnd(0,0.1))*w2;
end
```

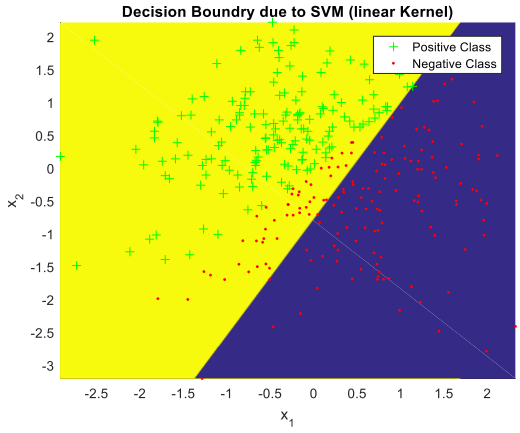
The code here is exactly the same codes with sub gradient descent with a disturbance in a N(0,0.1). Which is unbiased.

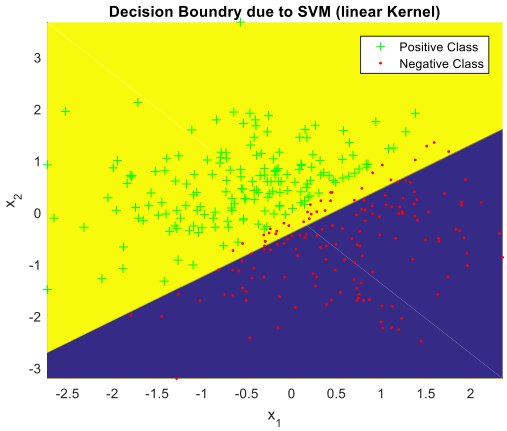
The optimization result here is far better than the sub gradient descent with respect to the Optimal objective function value:

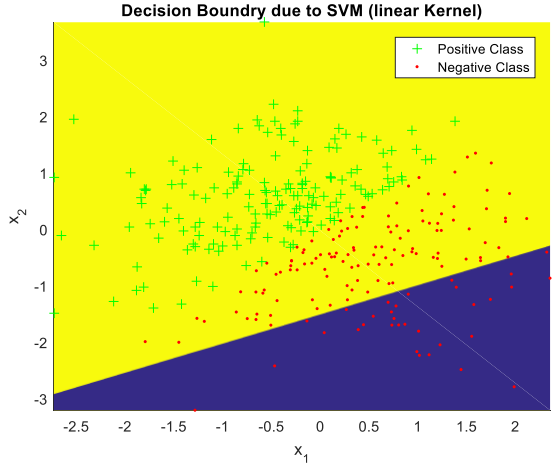
```
W

W =

  -2.1413    1.7576

Optimal Objective function value 0.5*W^2

ans =

   3.8371

Elapsed time is 13.479349 seconds.
```

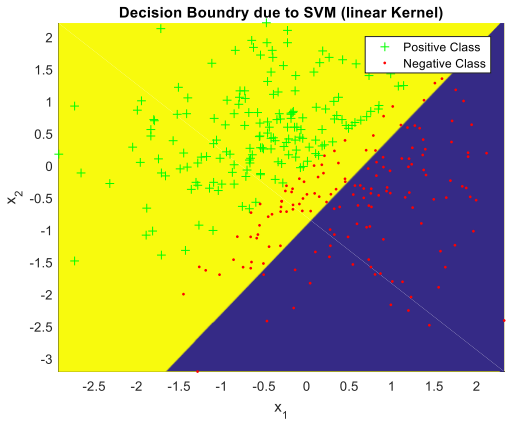
However, it's not stable with respect to accuracy.

I tried four times for the stochastic methods:




Decision Boundry due to SVM (linear Kernel)
Positive Class
Negative Class
3
2
1
0
-1
-2
-3
x2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x1


Decision Boundry due to SVM (linear Kernel)
Positive Class
Negative Class
3
2
1
0
-1
-2
-3
x2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x1


Decision Boundry due to SVM (linear Kernel)
Positive Class
Negative Class
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
x2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
x1

Summary:
Because the tolerance is the same, so these three methods are under the same condition.

With respect to running time:
Stochastic sub gradient<sub gradient descent < the dual ascent optimization method.

With respect to Optimization value:
Stochastic sub gradient<sub gradient descent < the dual ascent optimization method too.

But Stochastic sub gradient is not stable.