Question 1

1. (15 points) Consider a dataset with three data points in $\mathbb{R}^2$:

$$X = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ -2 & 0 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \end{bmatrix}$$

Manually solve the optimal hyperplane optimization problem (below) to get the optimal hyperplane $(b^*, \mathbf{w}^*)$ and its margin.

Optimization Problem:

$$\min_{b,w} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{subject to: } y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \qquad (n = 1, \ldots, N)$$

We rewrite X and y as a system of inequalities

- $-b \geq 1$
- $+w_2 - b \geq 1$
- $-2w_1 + b \geq 1$

$\Rightarrow -w_1 + \frac{1}{2}w_2 \geq 1$

$\Rightarrow -w_1 \geq 1 \quad = w_1 \leq -1$

$\Rightarrow w_2 \geq 0$

Using our minimization condition

$$\min_{b,w} \frac{1}{2}w^Tw = \min_{b,w} \frac{1}{2}(w_1\ w_2)\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \min_{b,w} \frac{1}{2}(w_1^2 + w_2^2)$$

By inspection, our minimizing values would be

$w_1 = -1$

$w_2 = 0$

$b = -1$

The margin is simply:

$$\frac{1}{\|\vec{w}\|} = \frac{1}{1} = \boxed{1}$$

$$\Rightarrow \boxed{w = \begin{pmatrix} -1 \\ 0 \end{pmatrix},\ b = -1}$$

Question 2

2. (15 points) Consider the following proof concerning the matrix Q from the linear hard-margin SVM algorithm.

Proof: Let $\mathbf{u} = [x_0, \mathbf{x}_u^T]^T \in \mathbb{R}^{d+1}$ and $\mathbf{x}_u \in \mathbb{R}^d$. Since we have $Q = \begin{bmatrix} 0 & 0_d^T \\ 0_d & I_d \end{bmatrix}$, then

$$\mathbf{u}^T Q \mathbf{u} = [x_0, \mathbf{x}_u^T] \begin{bmatrix} 0 & 0_d^T \\ 0_d & I_d \end{bmatrix} \begin{bmatrix} x_0 \\ \mathbf{x}_u \end{bmatrix} = \mathbf{x}_u^T \mathbf{x}_u \geq 0$$

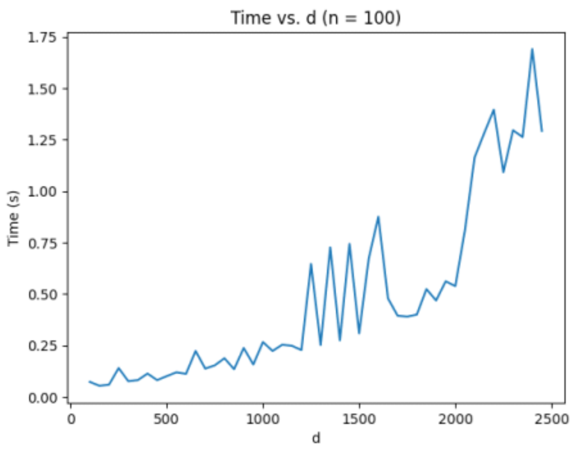
This is the case for arbitrary $\mathbf{u}$.

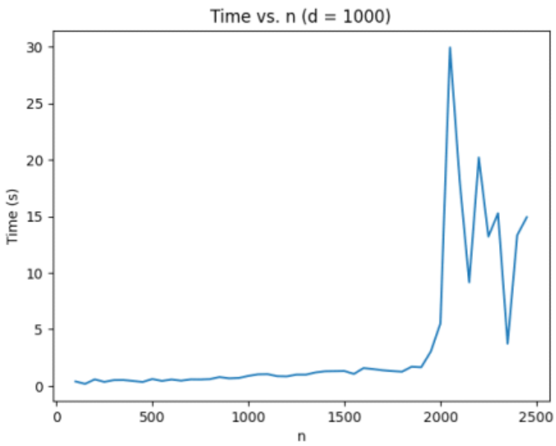
(a) What property of matrix Q does this prove?
(b) What does this property of Q mean for the standard QP problem?
(c) What is the usefulness of (b) in terms of finding a solution for the QP problem?

a) By definition it makes Q a positive semi-definite matrix
b) It makes the QP problem a convex problem
c) It helps us find the optimal solution easier being able to utilize properties of convex problems to use certain well known methods. Specifically QP-solvers can solve it in O((N+d)^3)

Question 3

Part b:

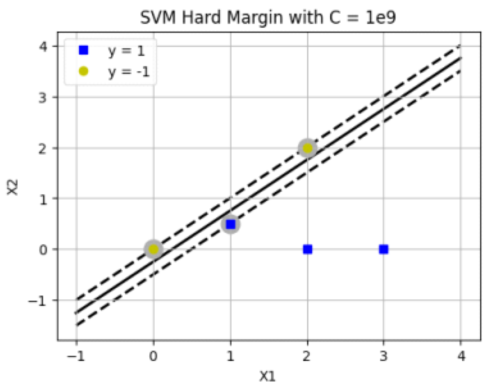


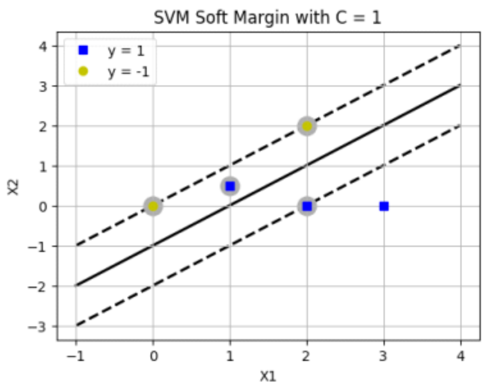


Our algorithm is not guaranteed to be efficient given any size of X. As we go to higher numbers of features or datapoints we get an algorithm that grows more than linearly. The specific big-oh notation is not obvious from the tests we have done above and would include more analysis into the calculation itself.

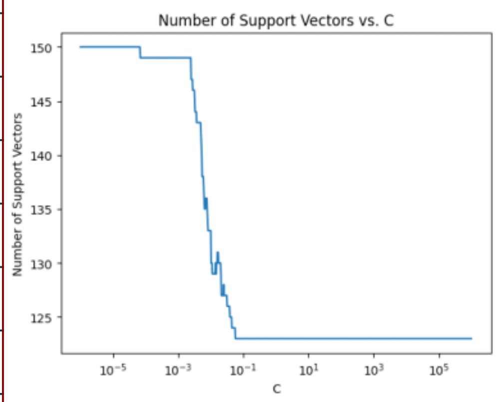
Question 4

Part a: In this part I add a point to the test data before that would be on the wrong side of the margin. Here I train both hard and soft margin SVM to see the difference. The results is that the hard margin changes a lot to make it still perfectly separable while the soft margin keeps the same margin as before and just allows the point to be misclassified.





Part b:



Here is a plot of the number of support vectors versus C. Notice that the x-axis is in log scale. We see behavior as we would expect where at very low C we get the maximum number of support vectors and as we increase C it decreases

Here I experiment with different kernels and extremes of C values for each

```
Linear kernel, C = 1, Training Accuracy =  1.0
Linear kernel, C = 1, Testing Accuracy =  0.8698245614035087
Linear kernel, C = 1e-6, Training Accuracy =  0.16
Linear kernel, C = 1e-6, Testing Accuracy =  0.09684210526315790
Linear kernel, C = 1e6, Training Accuracy =  1.0
Linear kernel, C = 1e6, Testing Accuracy =  0.8698245614035087
Polynomial kernel, C = 1, Training Accuracy =  0.9866666666666667
Polynomial kernel, C = 1, Testing Accuracy =  0.8080701754385965
Polynomial kernel, C = 1e-6, Training Accuracy =  0.16
Polynomial kernel, C = 1e-6, Testing Accuracy =  0.0968421052631579
Polynomial kernel, C = 1e6, Training Accuracy =  1.0
Polynomial kernel, C = 1e6, Testing Accuracy =  0.8059649122807018
RBF kernel, C = 1, Training Accuracy =  0.9933333333333333
RBF kernel, C = 1, Testing Accuracy =  0.871578947368421
RBF kernel, C = 1e-6, Training Accuracy =  0.16
RBF kernel, C = 1e-6, Testing Accuracy =  0.0968421052631579
RBF kernel, C = 1e6, Training Accuracy =  1.0
RBF kernel, C = 1e6, Testing Accuracy =  0.8887719298245614
```

Our results are interesting. For linear, polynomial, and RBF we get bad results for a low C values and better for higher. RBF with a high C value performed best on the testing set.