The purpose of this project is to implement the hard margin support vector machine (version $h2$).

Recall that we would like to solve the following optimization problem:

**Hard margin SVM** ($SVM_{h2}$):

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 \\
\text{subject to} & \quad w^\top u_i - b \geq 1 \quad i = 1, \ldots, p \\
& \quad -w^\top v_j + b \geq 1 \quad j = 1, \ldots, q,
\end{align*}$$

where $\{u_1, \ldots, u_p\}$ is a set of $p$ blue points and $\{v_1, \ldots, v_q\}$ is a set of $q$ red points in $\mathbb{R}^n$ (here, $n = 2$). We assume that these two sets of points are separable. Figure 1 shows examples of points and separating planes, while figure 2 shows an example of the separating plane with the margins.

![Figure 1: Examples of points and separating planes.](image)

The problem is to find a separating hyperplane $H_{w,b}$ of equation $w^\top x - b = 0$ which maximizes the smallest distance $\delta$ from these data points, called the *margin*.

The margin is $\delta = 1/\|w\|$. The margin hyperplanes are the hyperplanes $H_{w,b+1}$ (the blue hyperplane) of equation $w^\top x - b - 1 = 0$ and $H_{w,b-1}$ (the red hyperplane) of equation
\[w^T x - b + 1 = 0.\] In order to solve the above problem, we solve the dual program. The dual program is the following program:

**Dual of the Hard margin SVM (SVM\textsubscript{h2}):**

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \left( \lambda^\top \mu^\top \right) X^\top X \left( \begin{array}{c} \lambda \\ \mu \end{array} \right) - \left( \lambda^\top \mu^\top \right) 1_{p+q} \\
\text{subject to} & \quad \sum_{i=1}^{p} \lambda_i - \sum_{j=1}^{q} \mu_j = 0 \\
& \quad \lambda \geq 0, \quad \mu \geq 0,
\end{align*}
\]

where \(X\) is the \(n \times (p + q)\) matrix given by

\[X = \begin{pmatrix} -u_1 & \cdots & -u_p & v_1 & \cdots & v_q \end{pmatrix},\]

a matrix whose columns are the \(u_i\) and the \(v_j\).

Then \(w\) is determined as follows:

\[w = -X \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \sum_{i=1}^{p} \lambda_i u_i - \sum_{j=1}^{q} \mu_j v_j.\]

To solve the dual using ADMM we need to determine the matrices \(P, q, A\) and \(c\). We have

\[P = X^\top X, \quad q = -1_{p+q},\]

and since the only constraint is

\[\sum_{i=1}^{p} \lambda_i - \sum_{j=1}^{q} \mu_j = 0,\]
the matrix $A$ is the $1 \times (p + q)$ row vector

$$A = \begin{pmatrix} 1^T_p & -1^T_q \end{pmatrix},$$

and the right-hand side $c$ is the scalar

$c = 0$.

Obviously the matrix $A$ has rank 1.

We obtain $b$ using any $i_0$ such that $\lambda_{i_0} > 0$ and any $j_0$ such that $\mu_{j_0} > 0$. Since the corresponding constraints are active, we have

$$w^T u_{i_0} - b = 1, \quad -w^T v_{j_0} + b = 1,$$

so we obtain

$$b = w^T(u_{i_0} + v_{j_0})/2.$$

For improved numerical stability, we can average over the sets of indices defined as $I_{\lambda > 0} = \{i \in \{1, \ldots, p\} \mid \lambda_i > 0\}$ and $I_{\mu > 0} = \{j \in \{1, \ldots, q\} \mid \mu_j > 0\}$. We obtain

$$b = w^T \left( \left( \sum_{i \in I_{\lambda > 0}} u_i \right)/|I_{\lambda > 0}| + \left( \sum_{j \in I_{\mu > 0}} v_j \right)/|I_{\mu > 0}| \right) / 2.$$

The number $|I_{\lambda > 0}|$ of strictly positive $\lambda_i$ is denoted in the Matlab program by numsvl1 and the number $|I_{\mu > 0}|$ of strictly positive $\mu_j$ is denoted by numsvm1.

(50 points) Write a Matlab program that implements the above method and solve for $w$ and $b$. The Euclidean norm of $w$ is denoted by $n w$. The ADMM optimizer is already implemented in SVMhard2 in your project zip file.

The functions buildhardSVM2, qsolve1, showdata, showSVMs2 and makeline (which is needed for solveSVM2) are given in the file folder Matlabcode4.

The parameter $\rho$ is used by the function qsolve1 that solve the optimization program using ADMM. We suggest using $\rho = 10$.

A difficulty that arises is to tune the tolerance parameters needed to deal with floating-point arithmetic. We suggested some tolerance parameters for you, but you are welcome to experiment.

Run your program on the following sets of data points:

$$v1 = [1 2 3 1 1 3 -1 -3;-1 0 -2 -0.5 -4 -3 -3 -3];$$
$$u1 = [-1 -1 0 1 -3 -4 0.5 3 0.5;0 1 2 3 0 -2 2 2.5 2.5];$$
$$u2 = 10.1*randn(2,20)+15;$$
$$v2 = -10.1*randn(2,20)-15;$$
If you want to suppress printing $\lambda$ and $\mu$, call the function using

$[\_, \_, w, b] = \text{SVMhard2}(\rho, u, v)$

What do you observe when you run hard SVM on the data set $(u_3, v_3)$? Write your answer in the report. Also try running these programs with several different random seeds and report the results. Try varying $\rho$ - what behavior do you notice? Are there choices of $\rho$ where the program breaks?