The project involves implementing a Support Vector Machine (SVM) classifier that performs training using Sequential Minimal Optimization (SMO).
The SVM classifier will be tested on the following classification problem.

Dataset

"**Banknote authentication dataset**" consists of 1372 images of real and imitated banknotes. Four digital image features represent the image of each money banknote. Two classes (real money and imitation) needs to be identified using these four features. Dataset: https://archive.ics.uci.edu/ml/datasets/banknote+authentication

The algorithm

SMO algorithm will be employed as shown in the pseudocode in figure 1. The formulae required for the calculations are given in figure 2.
The implemented SVM will consist of two programs. SVM_train will take an input file (training data), C, tol, and max_passes as the input, and return the parameters $\alpha_1, \alpha_2, \ldots, \alpha_m, b$ as the output. SVM_test will take an input file (test data) and the parameters trained by SVM_train program as input and return the predicted labels (+1's and −1's) of the training data.

- ANY PROGRAMMING LANGUAGE CAN BE PREFERRED.
- SVM SOFTWARE LIBRARIES OR TOOLBOXES ARE NOT ALLOWED!

Test procedure

The dataset should be randomly divided into training (80% of the data) and test (20% of the data) sets. Make sure that the ratio of positive and negative samples in both datasets are balanced.
After the test is completed, the predicted labels of the test dataset will be evaluated to calculate the correct prediction percentage. The correct prediction ratio will also be reported in the final report.

- The due date of the project is January 10^{th}, 2020.
- The projects can be done individually or in teams of (at most) two people.
- The final report should include the source code, the executable file (if generated), and the correct prediction percentage on the given dataset.

Input:

C: regularization parameter

tol: numerical tolerance

max_passes: max # of times to iterate over α's without changing

$(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$: training data

Output:

$\alpha \in \mathbb{R}^m$: Lagrange multipliers for solution

$b \in \mathbb{R}$: threshold for solution

- Initialize $\alpha_i = 0, \forall i, \quad b = 0$.
- Initialize $passes = 0$.
- **while** $(passes < max\_passes)$
  - $num\_changed\_alphas = 0$.
  - **for** $i = 1, \ldots m$,
    - Calculate $E_i = f(x^{(i)}) - y^{(i)}$ using (1).
    - **if** $((y^{(i)} E_i < -tol \quad \&\& \quad \alpha_i < C) \; || \; (y^{(i)} E_i > tol \quad \&\& \quad \alpha_i > 0))$
      - Select $j \neq i$ randomly.
      - Calculate $E_j = f(x^{(j)}) - y^{(j)}$ using (1).
      - Save old α's: $\alpha_i^{(\text{old})} = \alpha_i, \; \alpha_j^{(\text{old})} = \alpha_j$.
      - Compute L and H by (2) or (3).
      - **if** $(L == H)$
        continue to next i.
      - Compute η by (4) .
      - **if** $(\eta >= 0)$
        continue to next i.
      - Compute and clip new value for α_j using (5) and (6) .
      - **if** $(|\alpha_j - \alpha_j^{(\text{old})}| < 10^{-5})$
        continue to next i.
      - Determine value for α_i using (7) .
      - Compute b_1 and b_2 using (8) and (9) respectively.
      - Compute b by (10).
      - $num\_changed\_alphas := num\_changed\_alphas + 1$.
    - **end if**
  - **end for**
  - **if** $(num\_changed\_alphas == 0)$
    $passes := passes + 1$
  - **else**
    $passes := 0$
- **end while**

Figure 1: SMO algorithm. $x^{(i)}$ denotes the i^{th} input vector, where $y^{(i)}$ denotes the class ($+1$ or -1) of the i^{th} vector.

$$f(x) = \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b \tag{1}$$

$$\text{If } y^{(i)} \neq y^{(j)}, \quad L = \max(0, \alpha_j - \alpha_i), \quad H = \min(C, C + \alpha_j - \alpha_i) \tag{2}$$

$$\text{If } y^{(i)} = y^{(j)}, \quad L = \max(0, \alpha_i + \alpha_j - C), \quad H = \min(C, \alpha_i + \alpha_j) \tag{3}$$

$$\eta = 2\langle x^{(i)}, x^{(j)} \rangle - \langle x^{(i)}, x^{(i)} \rangle - \langle x^{(j)}, x^{(j)} \rangle \tag{4}$$

$$E_k = f(x^{(k)}) - y^{(k)}$$

$$\alpha_j := \alpha_j - \frac{y^{(j)}(E_i - E_j)}{\eta} \tag{5}$$

$$\alpha_j := \begin{cases} H & \text{if } \alpha_j > H \\ \alpha_j & \text{if } L \leq \alpha_j \leq H \\ L & \text{if } \alpha_j < L. \end{cases} \tag{6}$$

$$\alpha_i := \alpha_i + y^{(i)} y^{(j)} (\alpha_j^{(\text{old})} - \alpha_j) \tag{7}$$

$$b_1 = b - E_i - y^{(i)}(\alpha_i - \alpha_i^{(\text{old})})\langle x^{(i)}, x^{(i)} \rangle - y^{(j)}(\alpha_j - \alpha_j^{(\text{old})})\langle x^{(i)}, x^{(j)} \rangle \tag{8}$$

$$b_2 = b - E_j - y^{(i)}(\alpha_i - \alpha_i^{(\text{old})})\langle x^{(i)}, x^{(j)} \rangle - y^{(j)}(\alpha_j - \alpha_j^{(\text{old})})\langle x^{(j)}, x^{(j)} \rangle \tag{9}$$

$$b := \begin{cases} b_1 & \text{if } 0 < \alpha_i < C \\ b_2 & \text{if } 0 < \alpha_j < C \\ (b_1 + b_2)/2 & \text{otherwise} \end{cases} \tag{10}$$

Figure 2: The calculations required for the SMO algorithm. $\langle x, y \rangle$ denotes the dot product ($x^T y$) of the vectors x and y.