CS771: Assignment 1

Aayushman Gupta (210021) **Divya Gera (210352)**

Mridul Pandey (210634) **Rudransh Goel (210880)** **Siddharth Garg (211031)**

Sourav Sharma (211055)

Question 1-By giving a detailed mathematical derivation (as given in the lecture slides), show how a CAR-PUF can be broken by a single linear model. Give derivations for a map $\phi : \{0,1\}^{32} \rightarrow \mathbb{R}^D$ mapping 32 -bit $0/1$-valued challenge vectors to D-dimensional feature vectors (for some $D > 0$) so that for any CAR-PUF, there exists a D-dimensional linear model $\mathbf{W} \in \mathbb{R}^D$ and a bias term $b \in \mathbb{R}$ such that for all $\mathrm{CRPs}(\mathbf{c}, r)$ with $\mathbf{c} \in \{0,1\}^{32}, r \in \{0,1\}$, we have

$$\frac{1 + \text{sign}\left(\mathbf{W}^\top \phi(\mathbf{c}) + b\right)}{2} = r$$

Answer 1-

Defining notations for a single arbiter PUF (similar to the ones used in the lecture):
Let $\mathbf{c_i}$ represent the i^{th} challenge bit and $\mathbf{c_i} \in \{0,1\}$
Let $\mathbf{t_i^u}$ represent the time at which the upper signal leaves the i^{th} mux and $\mathbf{t_i^l}$ represent the time at which the lower signal leaves the i^{th} mux
Let p_i, q_i, r_i and s_i represent the various delays of the mux as shown in the figure below:

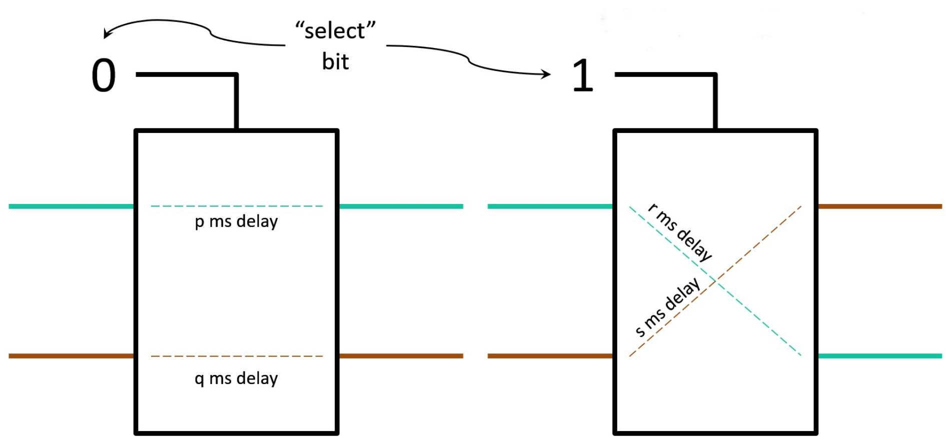


Figure 1: A simple multiplexer

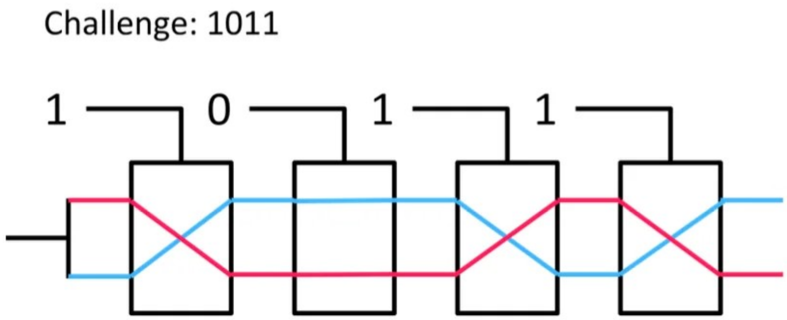


Figure 2: PUF with 4 multiplexers

For i =1:

$$t_1^u = (1 - c_1) \cdot (t_0^u + p_1) + c_1 \cdot (t_0^l + s_1)$$
$$t_1^l = (1 - c_1) \cdot (t_0^l + q_1) + c1 \cdot (t_0^u + r_1)$$

For i^{th} MUX:

$$t_i^u = (1 - c_i) \cdot (t_{i-1}^u + p_i) + c_i \cdot (t_{i-1}^l + s_i)$$
$$t_i^l = (1 - c_i) \cdot (t_{i-1}^l + q_i) + c_i \cdot (t_{i-1}^u + r_i)$$

Let $\Delta_i \stackrel{\text{def}}{=} t_i^u - t_i^l$ denote the lag, if $\Delta_{31} < 0$, the upper signal reaches first, else, the lower signal reaches first.
Using the above expressions, we have

$$\begin{aligned}
\Delta_1 &= (1 - c_1) \cdot \left(t_0^u + p_1 - t_0^l - q_1\right) + c_1 \cdot \left(t_0^l + s_1 - t_0^u - r_1\right) \\
&= (1 - c_1) \cdot (\Delta_0 + p_1 - q_1) + c_1 \cdot (-\Delta_0 + s_1 - r_1) \\
&= (1 - 2c_1) \cdot \Delta_0 + (q_1 - p_1 + s_1 - r_1) \cdot c_1 + (p_1 - q_1)
\end{aligned}$$

To make notation simpler, let $d_i \stackrel{\text{def}}{=} (1 - 2c_i)$, then

$$\Delta_1 = \Delta_0 \cdot d_1 + \alpha_1 \cdot d_1 + \beta_1$$

where

$$\alpha_1 = (p_1 - q_1 + r_1 - s_1)/2$$
$$\beta_1 = (p_1 - q_1 - r_1 + s_1)/2$$

Note that a similar relation holds for any stage: $\Delta_i = d_i \cdot \Delta_{i-1} + \alpha_i \cdot d_i + \beta_i$

$$\alpha_i \stackrel{\text{def}}{=} (p_i - q_i + r_i - s_i)/2$$
$$\beta_i \stackrel{\text{def}}{=} (p_i - q_i - r_i + s_i)/2$$

We take $\Delta_{-1} = 0$ (absorb initial delays into p_0, q_0, r_0, s_0). Further solving recursively, we get:

$$\Delta_0 = \alpha_0 \cdot d_0 + \beta_0 \text{ (since } \Delta_{-1} = 0)$$
$$\Delta_1 = \Delta_0 \cdot d_1 + \alpha_1 \cdot d_1 + \beta_1$$

Plugging the value of Δ_0:

$$\Delta_1 = \alpha_0 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_1 + \beta_1$$
$$\Delta_2 = \alpha_0 \cdot d_2 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1) \cdot d_2 + \beta_2$$

Therefore, combining all, we get:

$$\begin{aligned}
\Delta_{31} &= w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_{31} \cdot x_{31} + \beta_{31} \\
&= \mathbf{W}^\top \mathbf{x} + b
\end{aligned}$$

where,

$$\begin{aligned}
&x_i = d_i \cdot d_{i+1} \cdot \ldots \cdot d_{31} \\
&w_0 = \alpha_0 \\
&w_i = \alpha_i + \beta_{i-1} \text{ (for } i > 0)
\end{aligned}$$

If $\Delta_{31} < 0$, upper signal wins and response is 0. If $\Delta_{31} > 0$, lower signal wins and response is 1. Thus, response is simply $\frac{\text{sign}\left(\mathbf{W}^\top \mathbf{x} + b\right) + 1}{2}$

CAR-PUF

Response for a single PUF is given by:

$$\frac{1 + \text{sign}\left(\mathbf{W}^\top \mathbf{x} + b\right)}{2}$$

where $\mathbf{x}$ and $\mathbf{W}$ are given by:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{31} \end{bmatrix} \quad \text{and} \quad \mathbf{W} \quad = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{31} \end{bmatrix}$$

where

$$\begin{aligned}
&x_i = d_{31} \cdot d_{30} \cdot \ldots \cdot d_i \\
&d_i = 1 - 2c_i
\end{aligned}$$

Here, d_i is defined in terms of c_i as shown in the previous subsection.
Let Δ_w represent the working PUF and Δ_r represent the reference PUF, then we have

$$\begin{aligned}
\Delta_w &= \mathbf{U}^T \mathbf{x} + p \\
\Delta_r &= \mathbf{V}^T \mathbf{x} + q
\end{aligned}$$

where $\mathbf{U}$ and $\mathbf{V}$ are respective feature vectors and p and q are respective bias terms for working and reference PUF.
By definition of **CAR-PUF**, response to the challenge (r) is zero when $|\Delta_w - \Delta_r| \leq \tau$. Squaring both sides, we have,

$$\begin{aligned}
&|\Delta_w - \Delta_r|^2 \leq \tau^2 \\
\Rightarrow &|\Delta_w - \Delta_r|^2 - \tau^2 \leq 0
\end{aligned}$$

Therefore, we get the value of the response as,

$$r = \frac{1 + \text{sign}(|\Delta_w - \Delta_r|^2 - \tau^2)}{2}$$

Further simplifying, and substituting the values of Δ_w and Δ_r:

$$\begin{aligned}
|\Delta_w - \Delta_r|^2 - \tau^2 &= \Delta_w^2 + \Delta_r^2 - 2\Delta_w\Delta_r - \tau^2 \\
&= (\mathbf{U}^T\mathbf{x} + p)^2 + (\mathbf{V}^T\mathbf{x} + q)^2 - 2(\mathbf{U}^T\mathbf{x} + p) \cdot (\mathbf{V}^T\mathbf{x} + q) - \tau^2 \\
&= (\mathbf{U}^T\mathbf{x})^2 + (\mathbf{V}^T\mathbf{x})^2 + 2(p - q) \cdot (\mathbf{U}^T - \mathbf{V}^T) \cdot \mathbf{x} - 2(\mathbf{U}^T\mathbf{x}) \cdot (\mathbf{V}^T\mathbf{x}) + (p - q)^2 - \tau^2
\end{aligned}$$

Expanding these terms:

$$\mathbf{U}^T\mathbf{x} = u_0 \cdot x_0 + u_1 \cdot x_1 + u_2 \cdot x_2 + \cdots = \sum_{i=0}^{31} u_i x_i$$

$$(\mathbf{U}^T\mathbf{x})^2 = (\sum_{i=0}^{31} u_i x_i)^2 = \sum_{i=0}^{31} u_i^2 x_i^2 + \sum_{i \neq j; i,j=0}^{31} u_i u_j x_i x_j \{x_i^2 = 1\}$$

$$(\mathbf{U}^T\mathbf{x})^2 + (\mathbf{V}^T\mathbf{x})^2 = \sum_{i=0}^{31} (u_i^2 + v_i^2) + \sum_{i \neq j; i,j=0}^{31} (u_i u_j + v_i v_j) x_i x_j$$

$$2(p-q) \cdot (\mathbf{U}^T - \mathbf{V}^T) \cdot \mathbf{x} = 2(p-q) \sum_{i=0}^{31} (u_i - v_i) x_i$$

$$(\mathbf{U}^T\mathbf{x}) \cdot (\mathbf{V}^T\mathbf{x}) = (\sum_{i=0}^{31} u_i x_i)(\sum_{j=0}^{31} v_j x_j) = \sum_{i=0}^{31} u_i v_i + \sum_{i \neq j; i,j=0}^{31} u_i v_j x_i x_j$$

Plugging in the above expressions, we get,

$$\mathbf{W}^\top \mathbf{x} + b = \sum_{i \neq j; i,j=0}^{31} (u_i u_j + v_i v_j + u_i v_j) x_i x_j + 2(p-q) \sum_{i=0}^{31} (u_i - v_i) x_i + \sum_{i=0}^{31} (u_i^2 + v_i^2 + u_i v_i) + (p-q)^2 - \tau^2$$
$$= w_{0,1} \cdot x_0 \cdot x_1 + w_{0,2} \cdot x_0 \cdot x_2 + w_{0,3} \cdot x_0 \cdot x_3 + \cdots + w_{30,31} \cdot x_{30} \cdot x_{31} + [w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_{31} \cdot x_{31}] + b$$

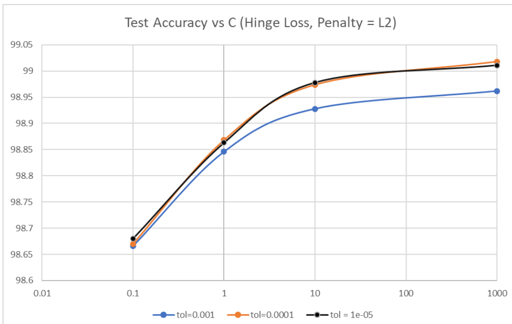
where:

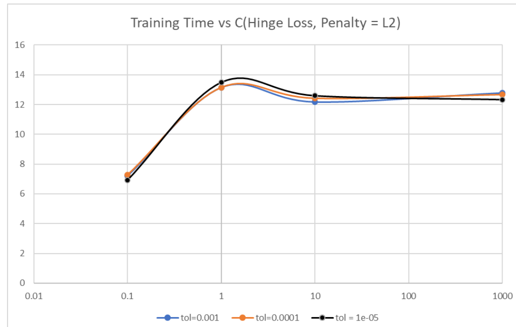
$$w_{i,j} = 2(u_i u_j + v_i v_j + u_i v_j)$$
$$w_i = 2(p-q)(u_i - v_i)$$
$$b = \sum_{i=0}^{31} u_i^2 + v_i^2 + u_i v_i + (p-q)^2 - \tau^2$$
$$\mathbf{W}^\top = [w_{0,1} \quad w_{0,2} \quad w_{0,3} \quad \cdots \quad w_{1,2} \quad w_{1,3} \quad \cdots \quad w_{30,31} \quad w_0 \quad w_1 \quad \cdots \quad w_{31}]$$
$$\mathbf{x}^\top = [x_0 x_1 \quad x_0 x_2 \quad \cdots \quad x_0 x_{31} \quad \cdots \quad x_{30} x_{31} \quad x_0 \quad x_1 \quad \cdots \quad x_{31}]$$

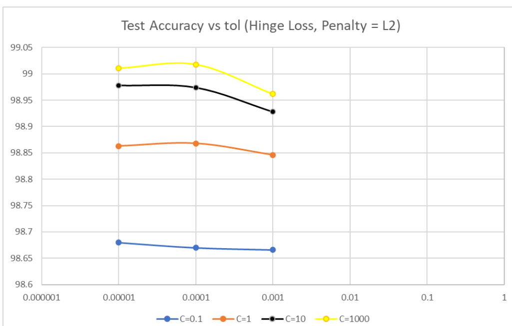
The vector, $\mathbf{x}$, contains terms involving x_i's where each x_i is itself a function of d_i, which is a function of c_i.
W is a 528 ($\binom{32}{2} + 32$) dimensional vector.

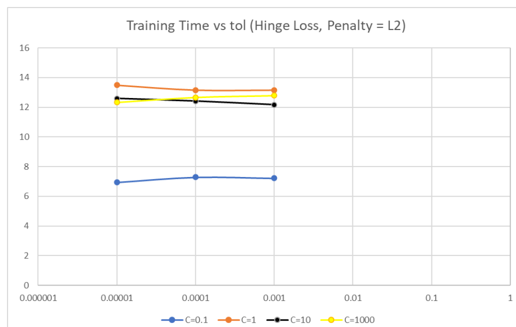
Question 3- Report outcomes of experiments with both the sklearn.svm.LinearSVC and sklearn.linear model.LogisticRegression methods when used to learn the linear model. In particular, report how various hyperparameters affected training time and test accuracy using tables and/or charts.

Answer 3-
We have conducted experiments for all 4 parts to study trends of training time and testing accuracy with choice of loss hyperparameter in LinearSVC, value of C in LinearSVC and LogisticRegression, changing tol in LinearSVC and LogisticRegression and changing penalty (regularization) hyperparameter in LinearSVC and LogisticRegression.









C	**Test Accuracy (%)**	**Training Time (secs)**
0.1	98.666	7.216
1	98.846	13.150
10	98.928	12.172
1000	98.962	12.785

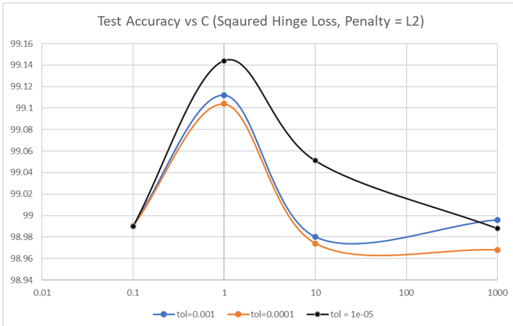
(a) tol = 0.001, penalty = l2, Loss hyperparameter = ‘hinge’

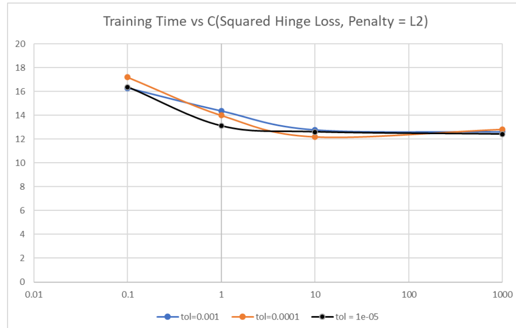
C	**Test Accuracy (%)**	**Training Time (secs)**
0.1	98.67	7.283
1	98.868	13.150
10	98.974	12.418
1000	99.018	12.666

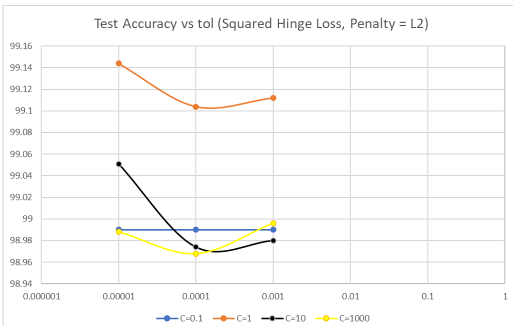
(b) tol = 0.0001, penalty = l2, Loss hyperparameter = ‘hinge’

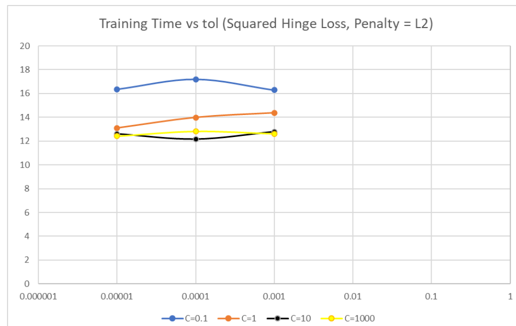
C	**Test Accuracy (%)**	**Training Time (secs)**
0.1	98.68	6.934
1	98.863	13.495
10	98.978	12.595
1000	99.011	12.332

(c) tol = 1e-05, penalty = l2, Loss hyperparameter = ‘hinge’









C	Test Accuracy (%)	Training Time (secs)
0.1	98.99	16.290
1	99.112	14.367
10	98.98	12.782
1000	98.996	12.617

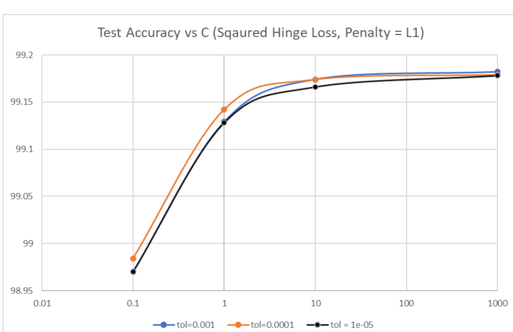
(a) tol = 0.001, penalty = l2, Loss hyperparameter = 'squared hinge'

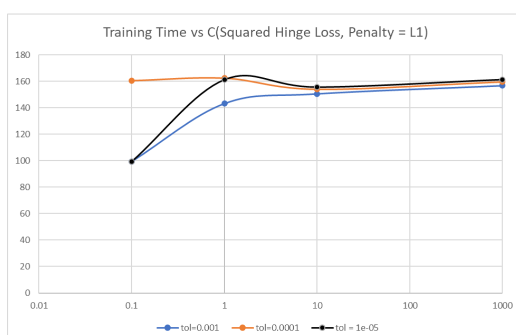
C	Test Accuracy (%)	Training Time (secs)
0.1	98.99	17.194
1	99.104	13.984
10	98.974	12.180
1000	98.968	12.812

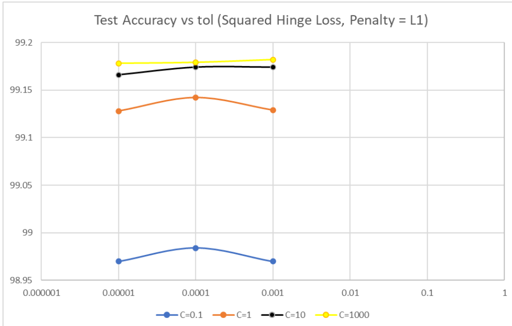
(b) tol = 0.0001, penalty = l2, Loss hyperparameter = 'squared hinge'

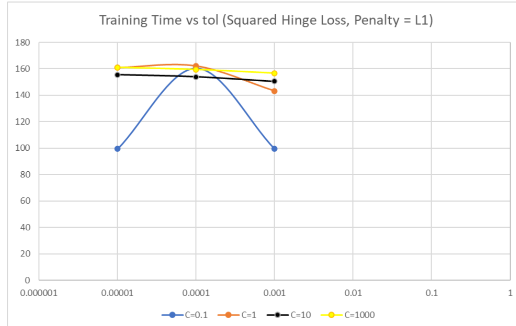
C	Test Accuracy (%)	Training Time (secs)
0.1	98.99	16.362
1	99.144	13.105
10	99.051	12.606
1000	98.988	12.417

(c) tol = 1e-05, penalty = l2, Loss hyperparameter = 'squared hinge'









C	Test Accuracy (%)	Training Time (secs)
0.1	98.97	99.407
1	99.129	143.168
10	99.174	150.488
1000	99.182	156.654

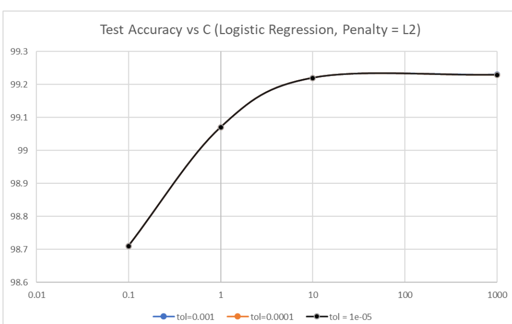
(a) tol = 0.001, penalty = l1, Loss hyperparameter = 'squared hinge'

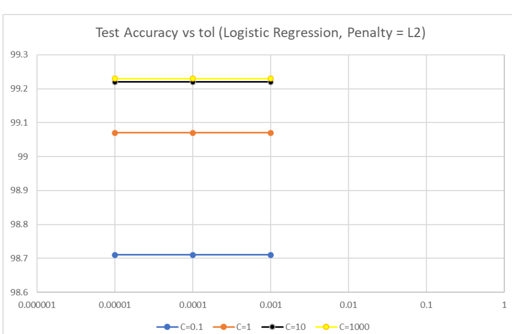
C	Test Accuracy (%)	Training Time (secs)
0.1	98.984	160.465
1	99.142	162.317
10	99.174	153.940
1000	99.179	159.52

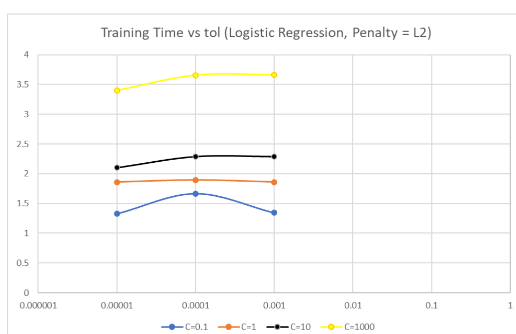
(b) tol = 0.0001, penalty = l1, Loss hyperparameter = 'squared hinge'

C	Test Accuracy (%)	Training Time (secs)
0.1	98.97	99.407
1	99.128	161.083
10	99.166	155.423
1000	99.178	161.1

(c) tol = 1e-05, penalty = l1, Loss hyperparameter = 'squared hinge'







C	Test Accuracy (%)	Training Time (secs)
0.1	98.71	1.343
1	99.07	1.862
10	99.22	2.286
1000	99.230	3.661

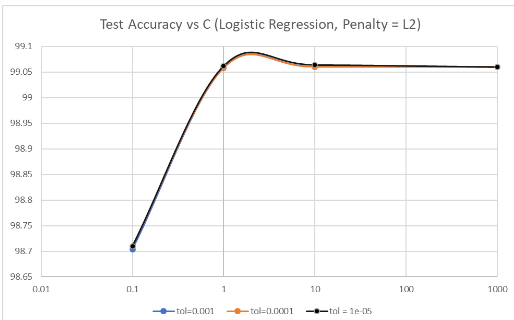
(a) tol = 0.001, penalty = l2, Logistic Regression

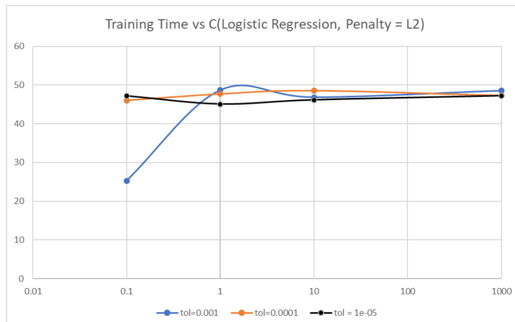
C	Test Accuracy (%)	Training Time (secs)
0.1	98.71	1.664
1	99.07	1.893
10	99.22	2.286
1000	99.229	3.655

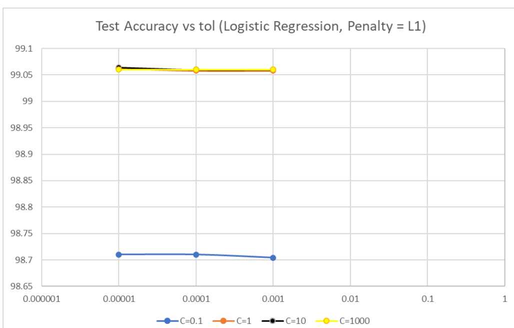
(b) tol = 0.0001, penalty = l2, Logistic Regression

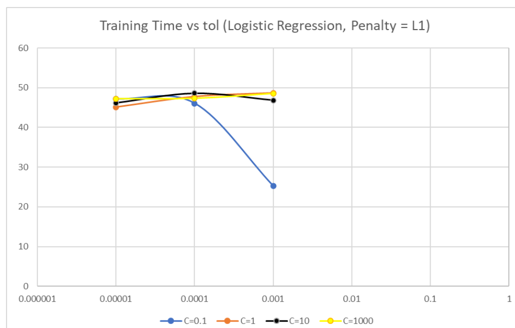
C	Test Accuracy (%)	Training Time (secs)
0.1	98.71	1.328
1	99.07	1.861
10	99.22	2.102
1000	99.229	3.399

(c) tol = 1e-05, penalty = l2, Logistic Regression









C	Test Accuracy (%)	Training Time (secs)
0.1	98.704	25.261
1	99.058	48.661
10	99.06	46.837
1000	99.06	48.526

(a) tol = 0.001, penalty = l1, Logistic Regression

C	Test Accuracy (%)	Training Time (secs)
0.1	98.71	46.048
1	99.058	47.734
10	99.06	48.601
1000	99.06	47.302

(b) tol = 0.0001, penalty = l1, Logistic Regression

C	Test Accuracy (%)	Training Time (secs)
0.1	98.71	47.178
1	99.062	45.086
10	99.064	46.138
1000	99.06	47.205

(c) tol = 1e-05, penalty = l2, Logistic Regression

Note:

- The combination of penalty = L1 and loss hyperparameter = ‘hinge’ is not allowed in the sklearn.linear_model.
- The solver used with penalty = L1 is ‘saga’.

From the experimental results obtained, the best accuracy is obtained with C = 1000, tol = 0.001, penalty = L2 with LogisticRegression. The testing accuracy = 99.23% (and the corresponding training time is 3.661 sec).

References

- https://scikit-learn.org/stable/modules/linear_model.html
- CS771 Lecture Slides 2023-24 Sem-II offering
- https://en.wikipedia.org/wiki/Khatri%E2%80%93Rao_product#Column-wise_Kronecker_product