Assignment 1

Aatman Jain
210018
aatman21@iitk.ac.in
Department of Electrical Eng.

Saumya Gupta
210944
saumyag21@iitk.ac.in
Department of Chemical Eng.

Siddhant Singhai
211028
ssinghai21@iitk.ac.in
Department of Mathematics

Utkarsh Agrawal
211127
utkarsha21@iitk.ac.in
Department of Mathematics

Vekariya Keval
211158
vkeval21@iitk.ac.in
Department of Mathematics

Tejus Khandelwal
211111
tejusk21@iitk.ac.in
Department of Mathematics

Introduction

When a multiplexer is presented with two input signals, it dynamically selects a path based on a control bit, resulting in distinctive output timings dictated by internal delays and signal arrival times. A combination of such MUXes forms an Arbiter PUF. In our proof of demonstrating that a CAR-PUF can be cracked by a linear model, we will start by proving that a standard PUF can be cracked using a linear model. Subsequently, leveraging this insight, we will develop the proof of cracking a CAR-PUF using a linear machine learning model.

1 Cracking a Single Arbiter PUF

1.1 Description of Time Delays and Signals

We assume that when the select bit is 0, the MUX does not swap the paths and continues on the same path. When the select bit is 1, it switches the path. Let the time delay for continuing the same path be (p_i, q_i) for 0 and 1 respectively and for the switched path be (r_i, s_i) for the path from 0 to 1 and 1 to 0 respectively. Let the times for the input of the signals be $t_i^{(u)}$ and $t_i^{(l)}$, the time for the output be $t_{i+1}^{(u)}$ and $t_{i+1}^{(l)}$, the select bit be c_i for the i^{th} multiplexer.

1.2 Prediction of Time Difference for 31st MUX

We also note that our input challenge consists of 31 bits, necessitating the use of 31 multiplexers. The time difference of the 31st multiplexer will be of prime importance in cracking the PUF, and we will be attempting to predict that. We define Δ_i as the difference between the time of arrival at the output of the upper signal and the lower signal for the i^{th} multiplexer. So, $\Delta_i = t_i^{(u)} - t_i^{(l)}$.

1.3 Mathematical Equations Relating Variables

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

$$t_i^{(l)} = (1 - c_i)(t_{i-1}^{(l)} + q_i) + c_i(t_{i-1}^{(u)} + r_i) \tag{2}$$

$$\begin{aligned} t_i^{(u)} - t_i^{(l)} &= (1 - c_i)(t_{i-1}^{(u)} - t_{i-1}^{(l)} + p_i - q_i) \\ &\quad + c_i(t_{i-1}^{(l)} - t_{i-1}^{(u)} + s_i - r_i) \\ &= (1 - 2c_i)(t_{i-1}^{(u)} - t_{i-1}^{(l)}) \\ &\quad + c_i(q_i - p_i + s_i - r_i) + p_i - q_i \end{aligned} \tag{3}$$

1.4 Introduction of New Variables

We introduce new variables α_i, β_i and d_i defined as:

$$\alpha_i = \frac{p_i - q_i + r_i - s_i}{2}, \quad \beta_i = \frac{p_i - q_i - r_i + s_i}{2}, \quad d_i = 1 - 2c_i$$

Substituting Δ_i and Δ_{i-1} in Equation eq 3, we obtain:

$$\Delta_i = \Delta_{i-1} \cdot d_i + \alpha_i \cdot d_i + \beta_i$$

1.5 Finding the 31st Delta

We attempt to obtain the 31st delta (Δ_{31}) by recursively applying the given recurrence formula. Assuming Δ_{-1} is 0, we start with:

$$\Delta_0 = \alpha_0 \cdot d_0 + \beta_0$$

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

Substituting the expression for Δ_0 into the equation for Δ_1, we obtain:

$$\Delta_1 = \alpha_0 \cdot d_1 \cdot d_0 + (\beta_0 + \alpha_1) \cdot d_1 + \beta_1$$

Similarly, for Δ_2, we can use the recurrence formula and obtain:

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

1.6 Observation of a Clear Pattern

We observe a clear pattern in the expression for Δ_i, where Δ_i can be represented as a linear function of a new variable x_j for j ranging from 0 to i. Specifically, we find that:

$$\Delta_i = w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_i \cdot x_i + \beta_i$$

where $x_j = d_j \cdot d_{j+1} \cdot d_{j+2} \ldots d_i$. This linear expression provides a systematic way to compute Δ_i based on the values of x_j and the coefficients w_j and β_i. For us the value of i is 31 (0-based indexing).

$$\Delta_{31} = w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_{31} \cdot x_{31} + \beta_{31}$$

where $w_0 = \alpha_0$, $w_i = \alpha_i + \beta_{i-1}$ and $x_j = d_j \cdot d_{j+1} \cdot d_{j+2} \ldots d_{31}$.

1.7 Conclusive Remarks for Single Arbiter PUF

We have demonstrated that the behavior of a single Arbiter PUF can be modeled as a linear function of $x_j = d_j \cdot d_{j+1} \cdot \ldots \cdot d_{31}$, where d_j is a transformed variable of the selection bit of the j^{th} multiplexer. The coefficients of this linear function, w_j, can be easily predicted using a machine learning model. This implies that the response of a single Arbiter PUF can be effectively cracked by a linear machine learning model.

References : The proof for the linear model used to crack Single Arbiter PUF is taken from CS771 Lecture Slides IITK.

2 Extending the linear model to crack CAR-PUF

2.1 Description of CAR-PUF

A CAR-PUF uses 2 arbiter PUFs– a working PUF and a reference PUF, as well as a secret threshold value τ. Given a challenge, it is fed into both the working PUF and reference PUF and the timings for the upper and lower signals for both PUFs are measured. Let Δ_w and Δ_r be the difference in timings experienced for the two PUFs on the same challenge. The response to this challenge is 0 if $|\Delta_w - \Delta_r| \leq \tau$ and 1 if $|\Delta_w - \Delta_r| > \tau$ where $\tau >$ is the secret threshold value.

2.2 Calculating $|\Delta_w - \Delta_r|$

From section 1.6 we have:
Δ_w = w$_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_{31} \cdot x_{31} + \beta_{w_{31}}$
where $w_0 = \alpha_{w_0}$, $w_i = \alpha_{w_i} + \beta_{w_i}$ and $x_j = d_j \cdot d_{j+1} \cdot d_{j+2} \ldots d_{31}$
Δ_r = r$_0 \cdot x_0 + r_1 \cdot x_1 + r_2 \cdot x_2 + \ldots + r_{31} \cdot x_{31} + \beta_{r_{31}}$
where $r_0 = \alpha_{r_0}$, $r_i = \alpha_{r_i} + \beta_{r_i}$ and $x_j = d_j \cdot d_{j+1} \cdot d_{j+2} \ldots d_{31}$

The value of x_j's would be the same for both the reference and working PUFs as they both have the same value of c_j (and thus d_j) for each j from 0 to 31.
Thus, $\Delta w - \Delta r =$ k$_0 \cdot x_0 + k_1 \cdot x_1 + k_2 \cdot x_2 + \ldots + k_{31} \cdot x_{31} + \beta_{31}$
where $k_i = w_i$ - r_i and $\beta_{31} = \beta_{w_{31}} - \beta_{r_{31}}$

2.3 Getting the linear expression for CAR-PUF

The CAR-PUF gives the value 0 if $|\Delta w - \Delta r| - \tau \leq 0$ and 1 otherwise. Thus our response to the challange would be $\frac{1+\text{sign}(|\Delta w - \Delta r| - \tau)}{2}$

Since we need to evaluate the expression $|\Delta w - \Delta r| \leq \tau$, we can square both the sides to obtain a mathematical equation which we can use to train our model. Squaring both the sides we get:

$\sum_{i=0}^{31}$k$_i^2 \cdot x_i^2$ + β_{31}^2 + $\sum_{i=0}^{30} \sum_{j=i+1}^{31} 2 \cdot k_i \cdot k_j \cdot x_i \cdot x_j$ + $\sum_{i=0}^{31} 2 \cdot w_i \cdot \beta \cdot x_i$ - $\tau^2 \leq 0$
We notice that because $x_j = d_j \cdot d_{j+1} \cdot d_{j+2} \ldots d_{31}$ and d_i = 1 - 2c_i where c_i denotes the bit of the i_{th} MUX, because d_i is either 1 or -1 when c_i is 0 or 1 respectively. This means x_j would be either 1 or -1 depending on parity of -1 in x_j, thus x_j^2 would always be 1, ∀ j ∈ [0, 31]. Using this we can combine $\sum_{i=0}^{31} k_i^2 \cdot x_i^2$ into the constant term alongside β_{31}^2 and τ^2.

Thus our final simplified expression which we will use in our model is :

$\sum_{i=0}^{30} \sum_{j=i+1}^{31} 2\cdot$ k$_i\cdot$ k$_j\cdot$ x$_i\cdot$ x$_j$ + $\sum_{i=0}^{31}$ 2$\cdot$w$_i \cdot \beta \cdot x_i$ + b where b is our bias term and is given by $\sum_{i=0}^{31} k_i^2$ + β_{31}^2 - τ^2

2.4 Final Linear model to crack CAR-PUF

Let $W^T\phi$(c) + b denote our final linear model. Then b = $\sum_{i=0}^{31} k_i^2$ + β_{31}^2 - τ^2
W^T = (a_1, a_2, a_3, ... , a_{528}), ϕ(c) = (x_0 , x_1 , ... , x_{31} , $x_0 \cdot x_1$, $x_0 \cdot x_2$, ... , $x_0 \cdot x_{31}$, $x_1 \cdot x_2$, $x_1 \cdot x_3$, ... , $x_1 \cdot x_{31}$, ... , $x_{30} \cdot x_{31}$)
where x_i = $d_i \cdot d_{i+1} \cdot d_{i+2} \ldots d_{31}$, d_i = 1-2$\cdot c_i$ and a_i are constants. The dimension of ϕ(c) is 528.

3 Outcomes with different models while tuning hyperparameters

3.1 Changing loss hyperparameter in LinearSVC:

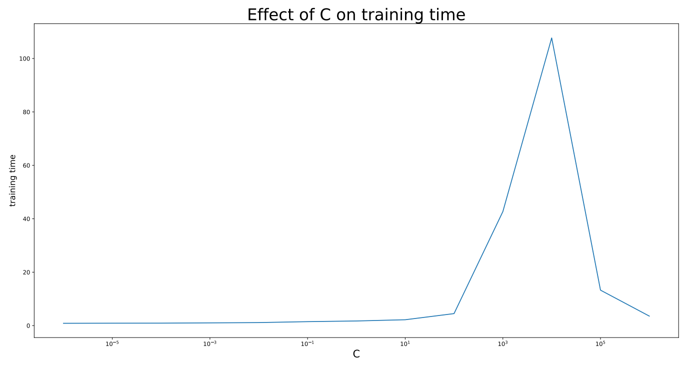
Accuracies on changing the loss function in the LinearSVC model, with default values of hyperparameters and max_iter = 10,000 are as shown:

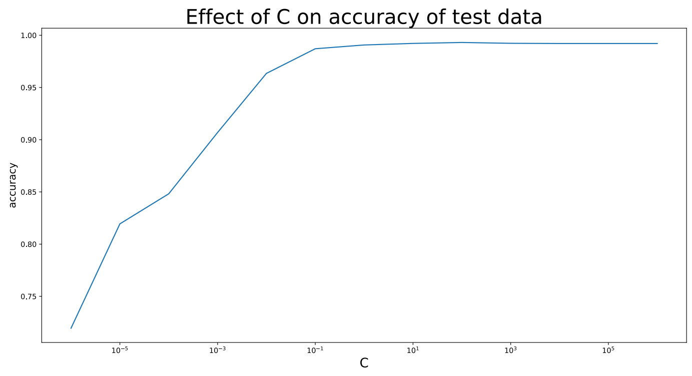
Loss	Training Time(in second)	Test Data
Hinge	14.808194310199998	98.84
Squared Hinge	15.428626121000026	99.1

3.2 Tuning C hyperparameter in LinearSVC and LogisticRegression:

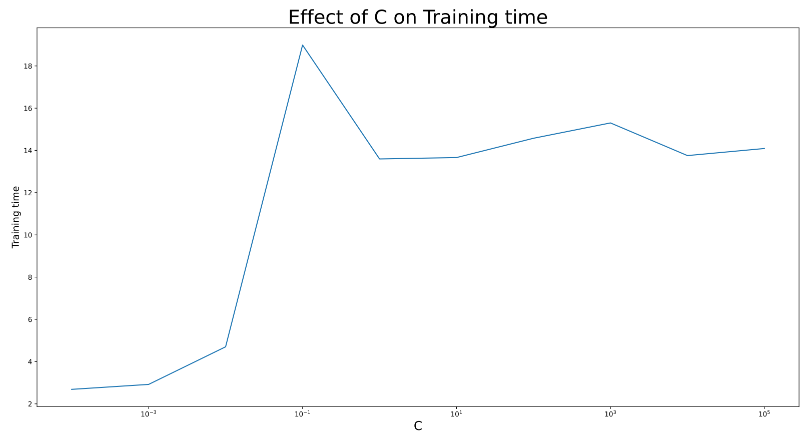
We have plotted graphs for default values of the parameters except for the parameter being compared and we have taken squared hinge loss for LinearSVC.

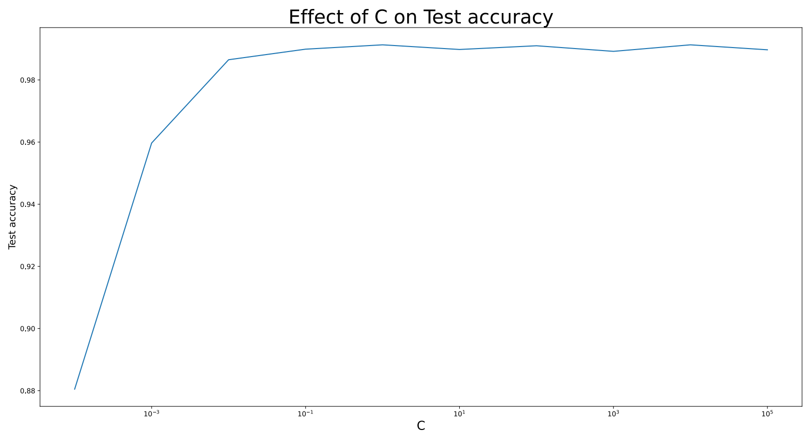
3.2.1 Logistic Regression





3.2.2 LinearSVC

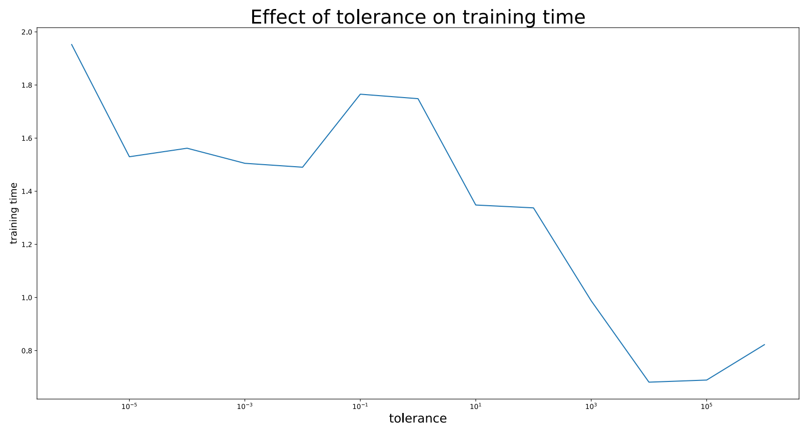


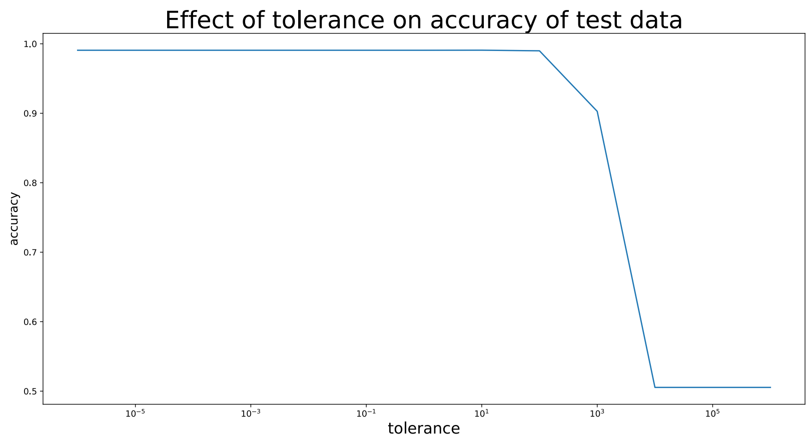


3.3 Tuning tol hyperparameter in LinearSVC and LogisticRegression:

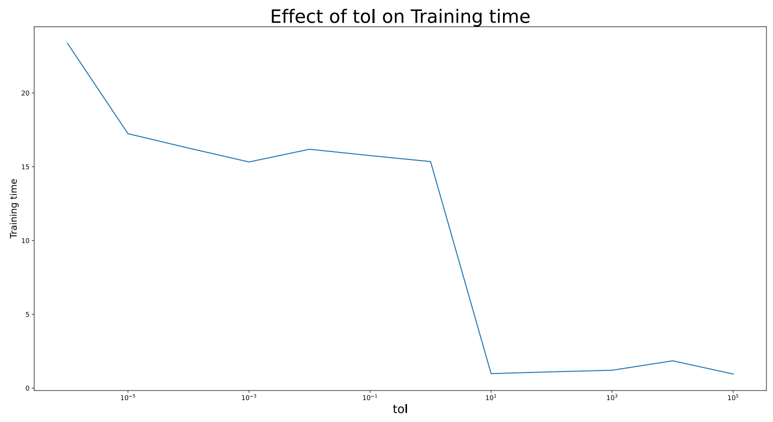
We have plotted graphs for default values of the parameters except for the parameter being compared and we have taken squared hinge loss for LinearSVC.

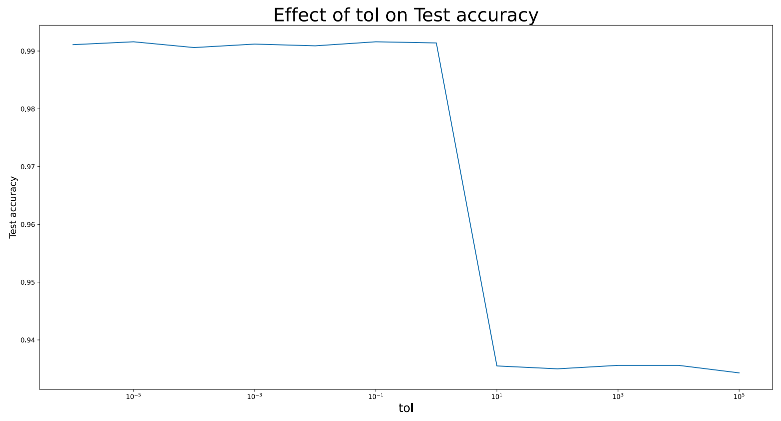
3.3.1 Logistic Regression





3.3.2 LinearSVC





3.4 Changing penalty hyperparameter in LinearSVC and LogisticRegression:

The values are for default values of the parameters except for the parameter being compared and we have taken squared hinge loss for LinearSVC.

Model	**Training Time (in second)**	**Test Data**
LinearSVC	L1: 45.34622061200025 L2: .115758701999766	L1: 98.94 L2: 99.17
Logistic Regression	L1: 138.4171830999985 L2: 5.408789499997511	L1: 98.18 L2: 99.06