CS771 Assignment-1, Report

July 17, 2024

Abstract

In the 1st part of the question, our objective is to demonstrate the vulnerability of the Cross Connection Arbiter PUF **(COCO-PUF)** to a single linear model, contrary to Melbo's belief. We achieve this by simplifying the COCO-PUF into two linear models and analyzing their behavior. We proceed systematically, starting from basic principles and gradually advancing towards our goal. We first decompose the COCO-PUF into two linear models, denoted as (u, p) and (v, q), which can accurately predict its Res1 and Res0. By simplifying the COCO-PUF in this manner, we lay the groundwork for our subsequent analysis. Throughout this process, we adhere to the instruction that our mapping function, $\phi(c)$, should depend solely on the challenge vector, c, and universal constants. Next, we proceed step by step, considering necessary transformations and adjustments, to arrive at a linear model with a different dimensionality compared to the individual 32-bit models. Through rigorous mathematical derivation, we define the mapping function $\phi(c)$ and determine the dimensionality of the feature vector, W, required for the linear model. Our approach is guided by the principles outlined in the problem statement, ensuring that our solution remains aligned with the given constraints, objectives and everything learned during the discussion hours and lectures.

In the last part of the assignment, we evaluate the performance of the sklearn.svm.**LinearSVC** model and the sklearn.linear_model.**Logistic-Regression** model on predicting the responses of the Cross Connection PUF (COCO-PUF). We explore the impact of various hyperparameters on training time and test accuracy. Specifically, we vary the loss hyperparameter in LinearSVC (hinge vs squared hinge), along with other hyperparameters such as C, tol, and penalty. We present our findings in terms of training time, test accuracy, and the influence of different hyperparameter settings on the model's performance. Please be aware that the data presented in the tables within the solution for part 3 has been tested on a local machine, and there may be slight variations if run on Google Colab or another platform.

1 Mathematical Derivation of Linear Model of COCO-PUF

Following the approach outlined in our lecture slides, we initially derive the linear model for a single Physical Unclonable Function (PUF) operating with 32-bit challenges. At this stage, our analysis applies to both the working (w) and reference (r) PUFs. We begin by specifying the input to both PUFs as the 32-bit challenges ($c_i' s$). Each bit in the 32-bit binary vector ($\mathbf{c}$) corresponds to a pair of multiplexers, each with four unknown delays (represented as p_i, q_i, r_i, and s_i) through which the incoming signal traverses.

Here, t_i^u and t_i^l represent the times at which the signal departs from the i_{th} MUX pair. Expressing t_i^u in terms of t_{i-1}^u, t_{i-1}^l and c_i, we obtain:

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

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

On Further modification, we found a relation between t_u^i and Δ_i.

$$t_i^u = t_{i-1}^u - (c_i) \cdot (\Delta_{i-1}) + c_i \cdot (s_{i-1} - p_i) + (p_i)$$

Following this relation recursively we found out that

after adding all equation:-

$$t_{32}^u - t_0^u = \sum_{i=1}^{n=32} c_i \left(-\Delta_{i-1} - p_i + s_i\right) + \sum_{i=1}^{32} p_i$$

Thus following the lecture slides we have, $\Delta_i = t_i^u - t_i^l$. Subtracting equations (1), (2) we get, $\Delta_i = \Delta_{i-1} \cdot d_i + \alpha_i \cdot d_i + \beta_i$, where α_i, β_i depend on constants that are indeterminable from the physical/ measurable perspective and thus we call them system constants and are given by:

$$\alpha_i = (p_i - q_i + r_i - s_i)/2 \quad \beta_i = (p_i - q_i - r_i + s_i)/2$$

where p_i, q_i, r_i, s_i are system parameters. Also d_i is governed by the challenge bits/input ($c_i' s$):

$$d_i = (1 - 2 \cdot c_i)$$

$\Delta_{-1} = 0$. Moreover, observing the recursion unfold carefully we can simplify this relation further:

$$c_1 \cdot \Delta_0 = c_1 \cdot (\alpha_0 \cdot d_0 + \beta_0)$$

$$c_2 \cdot \Delta_1 = c_2 \cdot (\alpha_0 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_1 + \beta_1)$$

$$c_3 \cdot \Delta_2 = \alpha_0 \cdot d_2 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1) \cdot d_2 + \beta_2$$

$$\vdots$$

Adding all till Δ gives us the recurrence generalization Δ_{31}, the last output.

$$c_{32} \cdot \Delta_{31} = c_1 \cdot w_0 \cdot x_0 + c_2 \cdot w_1 \cdot x_1 + \cdots + c_{32} \cdot w_{31} \cdot x_{31} + c_{32} \cdot \beta_{31} = \sum_{i=1}^{n=32} c_i \cdot \Delta_{i-1}$$

$$\sum_{i=1}^{n=32} c_i \cdot \Delta_{i-1} = \sum_{i=1}^{n=32} w_i \cdot x_i \cdot \sum_{j=i+1}^{n=32} c_j + \sum_{i=1}^{n=32} c_i \cdot (s_i - p_i)$$

Hence the final expression for the Upper Signal reaching time is

$$t_i^u = \sum_{i=1}^{n=32} w_i \cdot x_i \cdot \sum_{j=i+1}^{n=32} c_j + \sum_{i=1}^{n=32} c_i \cdot (s_i - p_i - \beta_{i-1}) + \sum_{i=1}^{n=32} p_i$$

where **w, x** are 32 dimensional vectors and **W, X** are 33 dimensional, just including the bias term $W_{32} = \beta_{31}$ and $X_{32} = 1$. Each term of **w, x** being:

$$b = \beta_{31} \qquad w_0 = \alpha_0 \qquad w_i = \alpha_i + \beta_{i-1} \qquad x_i = \prod_{j=i}^{31} d_i$$

Final feature Vector for the Linear Model is -

$$\mathbf{X} = [c_1, c_2, \cdots, c_n, x_1 \cdot \sum_{i=2}^{n=32} c_i, x_2 \cdot \sum_{i=3}^{n=32} c_i, \cdots, x_{31} \cdot c_{32}]$$

Remark: The only parameter we need for predicting the output of Δ_{31}, given a new challenge, is the vector **W**, which we extract from pre-existing efficient linear models which use the feature vector **X** for multiple challenge-response pairs as available data for training.

2 Dimensionality of our new feature map

As mentioned above the feature vector obtained contains 63 terms hence the dimensionality of our Map $\boldsymbol{\phi(\mathbf{x})}$ is **63**.

3 Solution to Q3

Now we have obtained the upper signal of the COCO as a Linear Model it can be used to create a classifier model which can correctly predict the Response 1 and Response 0 generated by COCO PUF.

So now, After analyzing and deducing **w, x, b** for the **upper signal time** t_i^u the COCO-PUF problem, let the Upper signal reaching time be defined as linear model for the PUF0 and PUF1 as analyzed above, be with the parameters (u, p) and (v, q).

$$t_{31}^{u1} = \mathbf{u}^T \cdot \mathbf{x} + \mathbf{p} \qquad t_{31}^{u0} = \mathbf{v}^T \cdot \mathbf{x} + \mathbf{q}$$

where **x** is the common feature vector composed of the 32 bits of a challenge and **u, v** are made of system parameters for both the PUF's respectively.

As for the question the response is based on the difference between upper time signal and Lower time signal of both the PUFs :

$$D = (t_{31}^{l1} \text{ - } t_{31}^{l0}) \qquad Response(0) = \begin{cases} 1, & \text{if } D \leq 0 \\ 0, & \text{if } D > 0 \end{cases}$$

$$D = (t_{31}^{u1} \text{ - } t_{31}^{u0}) \qquad Response(1) = \begin{cases} 1, & \text{if } D \leq 0 \\ 0, & \text{if } D > 0 \end{cases}$$

Further modification leads to the final response being very similar to that of a normal PUF linear model response. Thus the final response we receive for the difference between the upper signal reaching time for PUF1 and PUF0, applying the sign function, is:

$$\gamma(Resp1) = \frac{1 + sign[(t_{31}^{u1} - t_{31}^{u0})]}{2}$$

3.1 Simplifying $t_{31}^{u1} - t_{31}^{u0}$

As we know from earlier discussion:

$$t_{31}^{u1} = u^T \cdot X + p = (u_0, u_1, u_2, \cdots, u_{31}, p)^T \cdot (x_0, x_1, x_2, \cdots, x_{31}, 1)$$
$$t_{31}^{u0} = v^T \cdot X + q = (v_0, v_1, v_2, \cdots, v_{31}, q)^T \cdot (x_0, x_1, x_2, \cdots, x_{62}, 1)$$
$$t_{31}^{u1} - t_{31}^{u0} = (u - v)^T \cdot x + (p - q)$$
$$= (u_0 - v_0, u_1 - v_1, u_2 - v_2, \cdots, u_{61} - v_{61}, p - q)^T \cdot (x_0, x_1, x_2, \cdots, x_{61}, 1)$$

So, we can rename the system variables $u_i - v_i$ as z_i's and $p - q$ as y and write this difference as a new 63 dimensional linear model as follows:

$$t_{31}^{u1} - t_{31}^{u0} = \mathbf{z}^T \cdot \mathbf{X} + \mathbf{y} = (z_0, z_1, z_2, \cdots, z_{61}, y)^T \cdot (x_0, x_1, x_2, \cdots, x_{61}, 1)$$

Now,

$$\gamma = \frac{1 + sign[(\mathbf{Z}^T \cdot \mathbf{X})]}{2} \quad \textbf{where} \quad \mathbf{Z}^T \cdot \mathbf{X} = \mathbf{z}^T \cdot \mathbf{x} + \mathbf{y}, Z_{63} = y, X_{63} = 1$$

where z_i's depend upon the corresponding working and reference PUF system parameters or in laymen terms the internal delays in each MUX used. Also y is a constant bias term that comes from the 2 bias terms used to model the linear model for both the PUF's independently and **x** is the challenge binary vector.

Now the issue we face ahead of us is how to make a linear model even after we are using the square of the linear model made above.

3.2 Feature Transformation and Model Augmentation

As Proved above for the given 32-bit challenge we can convert it into 63 dimensional feature Map ϕ :

$$t_i^u = \sum_{i=1}^{n=32} w_i \cdot x_i \cdot \sum_{j=i+1}^{n=32} c_j + \sum_{i=1}^{n=32} c_i \cdot (s_i - p_i - \beta_{i-1}) + \sum_{i=1}^{n=32} p_i$$

Considering the terms involving the given input challenge bits c_i in the expression to be as features or the Mapping Function of our model. The terms involving the delay constant and other parameters of the PUF to be included in the model **W** and bias term **B** .
We can thus change the feature vector **x** to something inspired from the above equation and generate a linear model of different dimensions than the original one inside the bracket in the equation:

$$(W^T \cdot X)$$

Similar arguments can be given for the Response 0 as well in place of upper signal time we can use lower signal time and can derive the similar results that have been derived for Response 1.
The feature map will also remain same for the lower signal.

3.3 Q4 ans

Thus the **new dimensionality** of the newly generated feature vector will be:

$$31 + 32 = \mathbf{63\ features}$$

4 Performance Analysis for Models

4.1 Performance Comparison of LinearSVC with Different Loss Functions

Loss Function	Total Features	Model Train Time (s)	Map Time (s)	Acc0	Acc1
Hinge	63	2.886	.0161	0.9824	0.9972
Squared Hinge	63	1.784	0.0103	0.9824	0.9982

Analysis:
From the table, we observe that the model trained with the squared hinge loss achieves a slightly higher test accuracy compared to when trained with the hinge loss. However, the difference in accuracy is insignificant. Interestingly, the model trained with the squared hinge loss takes slightly longer to train compared to the model trained with the hinge loss. This indicates that the squared hinge loss may lead to a more complex optimization problem. Nonetheless, the difference in training time between the two models is relatively small.

Possible reasons for the observed differences:

- The squared hinge loss penalizes outliers more strongly than the hinge loss, potentially leading to improved generalization performance.
- The optimization problem associated with the squared hinge loss may require more iterations to converge, leading to increased training time.
- The dataset characteristics and the specific challenges posed by the COCO-PUF may favor one loss function over the other, influencing the model's performance.

4.2 Performance Comparison based on the value of C

Analysis for LinearSVC with different C values:

C Value	Total Features	Model Train Time (s)	Map Time (s)	Acc0	Acc1
Low	63	3.055	0.0158	0.9824	0.9972
Medium	63	3.1115	0.0156	0.9824	0.9984
High	63	3.1536	0.01592	0.9824	0.9984

Analysis for LogisticRegression with different C values:

C Value	Total Features	Model Train Time (s)	Map Time (s)	Acc0	Acc1
Low	63	3.579	0.0264	0.983	0.9959
Medium	63	3.73	0.0272	0.9833	0.9985
High	63	3.2836	0.0175	0.9832	0.9985

From the analysis, we observe the following: For LinearSVC:

- The model trained with a medium C value achieves the highest test accuracy among the three configurations.

For LogisticRegression:

- The model trained with a high C value achieves the highest test accuracy, followed closely by the model with a medium C value.

Possible reasons for the observed differences:

- Lower C values result in a simpler decision boundary, which may not capture the complexities of the dataset, leading to lower accuracy.
- Higher C values result in a more complex decision boundary, which may lead to better performance on the training set but could cause overfitting.
- The balance between bias and variance changes with different C values, affecting the model's performance on the test set.

5 Effect of Tolerance Parameter on Model Performance

Model	Tolerance	Total Features	Model Train Time (s)	Acc0	Acc1
LinearSVC	Low	63	2.9871	0.9823	0.9970
LinearSVC	Medium	63	2.9848	0.9824	0.9972
LinearSVC	High	63	2.704	0.9823	0.9962
LogisticRegression	Low	63	7.18	0.9832	0.9961
LogisticRegression	Medium	63	3.250	0.983	0.9959
LogisticRegression	High	63	0.889	0.9582	0.9847

From the table, we observe the following:

- For LinearSVC, varying the tolerance parameter leads to significant differences in training time, with the low tolerance resulting in the longest training time and the high tolerance resulting in the shortest training time.
- However, the test accuracy of the LinearSVC model decreases as the tolerance increases, indicating that higher tolerance values may lead to poorer generalization performance.
- For LogisticRegression, the differences in training time across different tolerance values are less pronounced compared to LinearSVC.

6 Effect of Penalty on Model Performance

Model	Penalty	Total Features	Model Train Time (s)	Acc0	Acc1
LinearSVC	l1	63	39.5	0.9824	0.9975
LinearSVC	l2	63	3.2	0.9824	0.9972
LogisticRegression	l1	63	36.59	0.983	0.9977
LogisticRegression	l2	63	3.354	0.983	0.9959

These results suggest that the choice of tolerance parameter can have a significant impact on both the training time and the test accuracy of the models. Lower tolerance values may lead to longer training times but potentially better performance, while higher tolerance values may result in faster training times but poorer generalization performance. It is crucial to strike a balance between training time and model performance when selecting the tolerance parameter.

Concluding Part 3:
From the above data analysis, it seems that LogisticRegression with a high C value performs the best in terms of test accuracy (0.9985) and an approximate model train time of 3.28 seconds and map time of 0.0175 seconds. This model strikes a good balance between accuracy and computational efficiency.

The Team

Name	Email	Roll Number
Akash Kumar Gupta	akashkg22@iitk.ac.in	220095
Dhruv Bansal	dhrubb22@iitk.ac.in	220359
Raunak Raj	raunakr22@iitk.ac.in	220876
Dhruv Mittal	mdhruv22@iitk.ac.in	220364
Ayush Shrivastava	ayushsri22@iitk.ac.in	220272
Ajay Singh	ajaysi22@iitk.ac.in	220090