CS771: Assignment 1
Team Melbros

Date: 18 February 2023

Akshat Rajani	210812
Aniruddh Pramod	210142
Harsh Bihany	210406
Monil Lodha	210630
Shreyash Kumar	211009

Abstract

This is our solution to the first mini-project of the course CS771. We are required to show (with a mathematical derivation) how a simple XORRO PUF can be broken by a single linear model, that this linear model can be extended to crack an Advanced XORRO PUF, and finally report the outcomes of our models, and the effects of hyperparameter tuning on model training time and accuracy.

1 Problem 1: Cracking the simple XORRO PUF

A simple XORRO PUF has two XORROs, both fed with the same R-bit challenge **a** as their config bit.

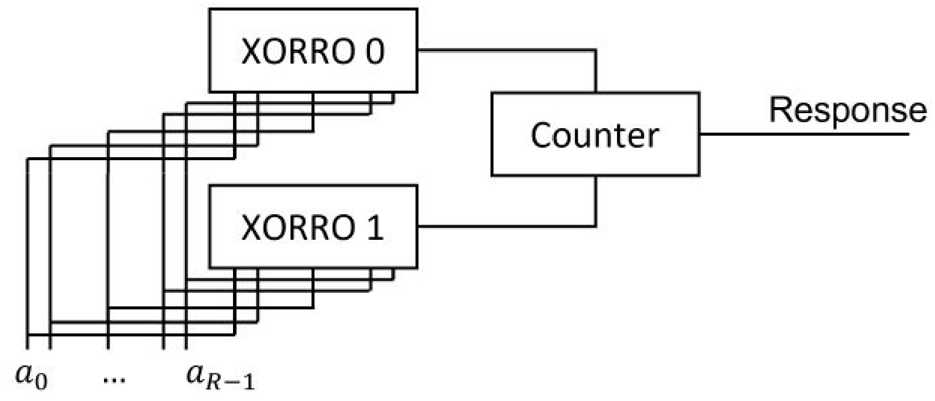


Figure 1: A simple XORRO PUF

Lets break down one XORRO

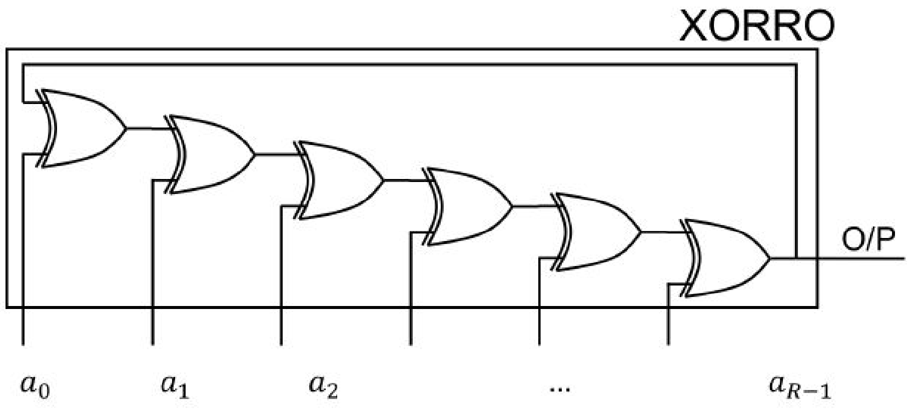


Figure 2: Inside a XOR ring oscillator

Let the output of i^{th} XOR gate be A_i. So, we have

$A_0 = XOR(A_{R-1}, a_0)$
$A_i = XOR(A_{i-1}, a_i) \qquad \forall i \in \{1, 2, ..., R-1\}$

In each cycle, the output of XORRO gets flipped (0 to 1 or 1 to 0). Thus, the XORRO undergoes two complete cycles in order to return the same output. That means, in each oscillation, every XOR gate is implemented twice and complementary output is produced at each XOR gate (since the final output is different). So, the time delay t_i incurred by i^{th} XOR gate in a single oscillation can be written as:

$$t_i = \delta^i_{A_{i-1}a_i} + \delta^i_{\overline{A}i-1a_i} \quad where\ \overline{A}_{i-1} = 1 - Ai - 1$$
$$\therefore t_i = \delta^i_{0a_i} + \delta^i_{1a_i}$$
$$\therefore t_i = a_i(\delta^i_{01} + \delta^i_{11}) + (1 - a_i)(\delta^i_{00} + \delta^i_{10})$$
$$\therefore t_i = a_i(\delta^i_{01} + \delta^i_{11} - \delta^i_{00} - \delta^i_{10}) + (\delta^i_{00} + \delta^i_{10})$$

The time period T of XORRO is therefore,

$$T = \sum_{i=0}^{R-1} t_i = \sum_{i=0}^{R-1} a_i(\delta^i_{01} + \delta^i_{11} - \delta^i_{00} - \delta^i_{10}) + \sum_{i=0}^{R-1}(\delta^i_{00} + \delta^i_{10})$$

The sign of the difference in time period of the two XORROs will decide the response of the Simple XORRO PUF.

For any quantity Q, define

$$\Delta Q \overset{\text{def}}{=} (Q)_1 - (Q)_0$$
$$\therefore \Delta T = \sum_{i=0}^{R-1} a_i(\Delta\delta^i_{01} + \Delta\delta^i_{11} - \Delta\delta^i_{00} - \Delta\delta^i_{10}) + \sum_{i=0}^{R-1}(\Delta\delta^i_{00} + \Delta\delta^i_{10})$$
$$\therefore \Delta T = w_1x_1 + w_2x_2 + ... + w_{R-1}x_{R-1} + b$$

$$\begin{aligned} where \quad w_i &= \Delta\delta^i_{01} + \Delta\delta^i_{11} - \Delta\delta^i_{00} - \Delta\delta^i_{10} \quad && \forall i \in \{0, 1, ..., R-1\} \\ x_i &= a_i && \forall i \in \{0, 1, ..., R-1\} \\ b &= \sum_{i=0}^{R-1}(\Delta\delta^i_{00} + \Delta\delta^i_{10}) \end{aligned}$$

Thus, the map $\phi : \{0, 1\}^R \to \mathbb{R}^D$ is simply $\phi(\mathbf{a}) = \mathbf{a}$
Observe that the feature vector ϕ is R dimensional i.e, **D=R**.
Thus, there exists $\mathbf{w} \in \mathbb{R}^R$, b $\in \mathbb{R}$ such that for all challenges c $\in \{0, 1\}^R$, the following expression

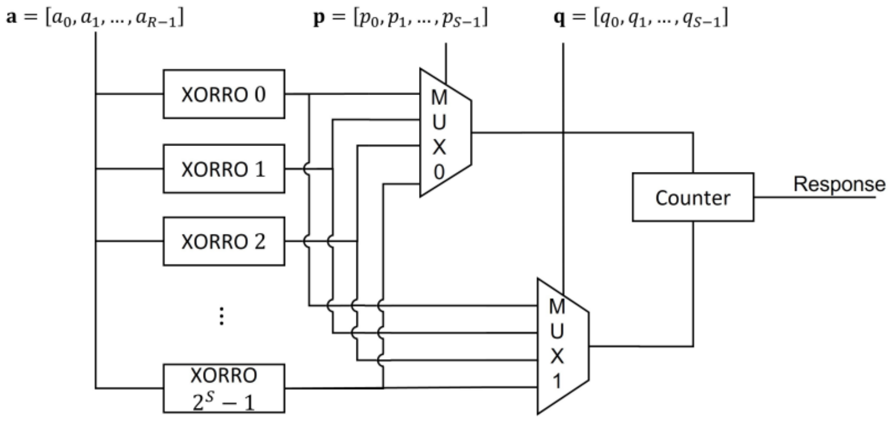
$$\frac{1 + sign(w^T\phi(c) + b)}{2}$$

gives the correct response

2 Problem 2: Cracking the Advanced XORRO PUF

2.1 Overview

Melbo presents us with a new challenge, including 2^S XORROs.



$\mathbf{a} = [a_0, a_1, \ldots, a_{R-1}]$
$\mathbf{p} = [p_0, p_1, \ldots, p_{S-1}]$
$\mathbf{q} = [q_0, q_1, \ldots, q_{S-1}]$
XORRO 0
XORRO 1
XORRO 2
XORRO $2^S - 1$
MUX 0
MUX 1
Counter
Response

Figure 3: An Advanced XORRO PUF using 2^S XORROs

The math regarding the problem statement remains the same as that of a simple XORRO PUF. Our task is to find an efficient method to reduce the problem down to two XORROs
We make $2^{S-1} * (2^S - 1)$ linear models, corresponding to each XORRO PUF pair that can be formed between the 2^S XORROs. We start off by allocating a 4-dimensional array to judiciously club each XORRO PUF pair and the training data available for that pair. The trained model for each pair is then returned to a different array which is further used in testing.

2.2 Procedure

Considering the soundness of the training data and imposing a reasonable space constraint, we presume each model will be trained on a number of datasets not exceeding 10,000. The *linear_models_data* has a matrix corresponding to each XORRO PUF pair, consisting of 10,000 rows, and *R+1* columns (R challenge bits and 1 output bit). The models are then trained by the classifier and each model is associated with its respective XORRO PUF pair in the *linear_models* array, which is used to make predictions on the test data.

2.3 Optimization

Our algorithm dictates that the XORRO PUF pairs be unordered, which provides us with a larger number of datapoints for training each model. Note that the first two dimensions of the *linear_models_data* constitute a lower triangular matrix. For entries of the form *linear_models_data*[i][j][r_k][R+1] having j $>$ i, i and j are flipped along with the output bit to accommodate the datapoint into our array.

3 Problem 4: Playing with Hyperparameters

3.1 Loss hyperparameter in LinearSVC

The 'loss function' characterizes how well the model performs on a given dataset. `sklearn.svm.LinearSVC` provides two loss functions, namely: 'hinge' and 'squared_hinge'.

Loss function	Training Time	Accuracy
'hinge' loss	0.726	0.939
'squared_hinge' loss	1.006	0.947

Table 1: Effect of loss functions on LinearSVC Model

3.2 'C'(hyperparameter) in LinearSVC and LogisticRegression

'C' is our regularization parameter. The strength of the regularization is inversely proportional to C. For large values of 'C' the model chooses smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points

(LinearSVC)

'C' is still the regulariazation parameter. We intend to penalize the extreme parameters in our model, that lead to overfitting. A high value of C tells the model to give more weight to the training data. A lower value of C will indicate the model to give complexity more weight at the cost of fitting the data.

(LogisticRegression)

3.2.1 LinearSVC

The semilog plots in figure 4 and 5 show the effect of changing the value of 'C' on Training Time and Accuracy for a LinearSVC model.

3.2.2 LogisticRegression

The semilog plots in figure 6 and 7 show the effect of changing the value of 'C' on Training Time and Accuracy for a LogisticRegression model.

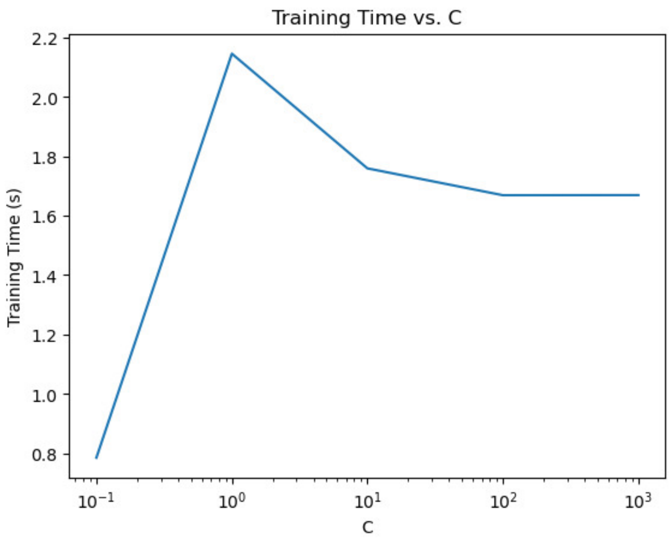


Figure 4: Effect of 'C' on Training Time for LinearSVC

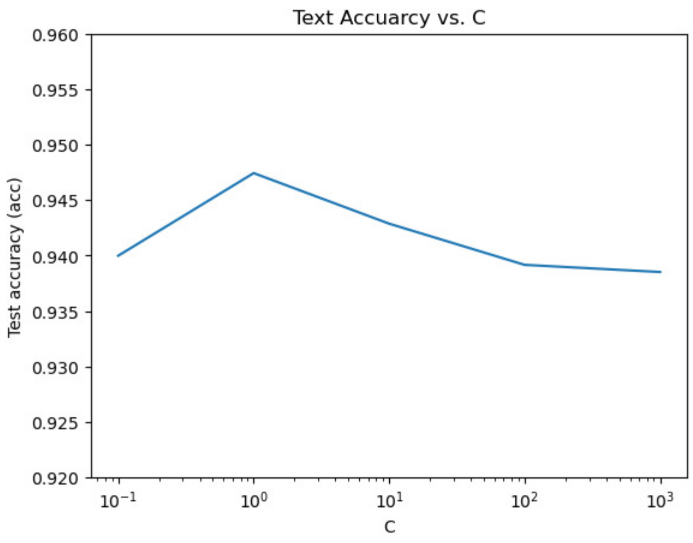


Figure 5: Effect of 'C' on Accuracy for LinearSVC

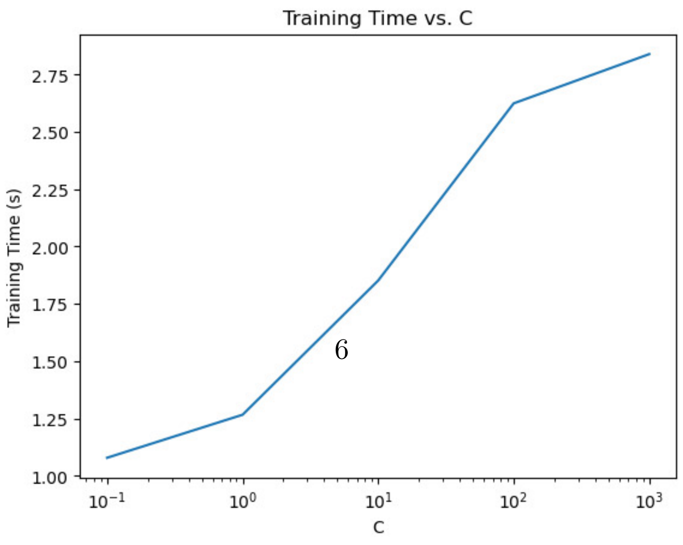


Figure 6: Effect of 'C' on Training Time for LogisticRegression

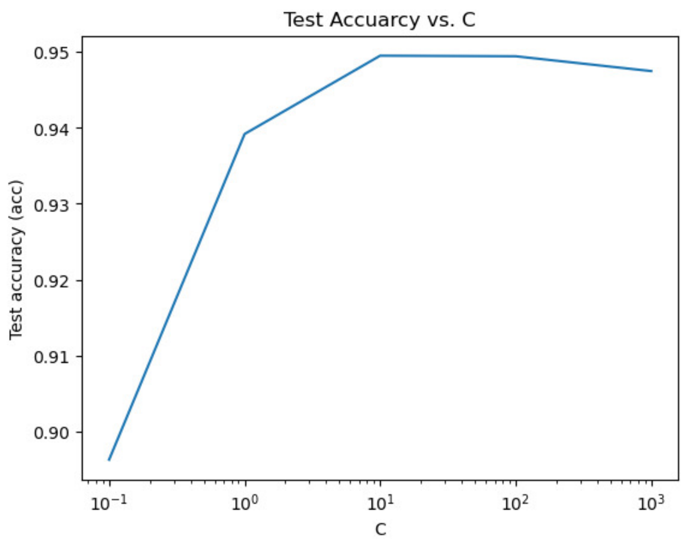


Figure 7: Effect of 'C' on Accuracy for LogisticRegression

3.3 'tol'(hyperparameter) in LinearSVC and LogisticRegression

'tol' is the parameter that decides the tolerance for stopping criteria. This tells sci-kit learn to stop trying to achieve convergence after it comes close enough to it. A higher tolerance means that the model underfits and stops training early, while more tolerance means that the model overfits and isn't allowed to stop prematurely.

3.3.1 LinearSVC

The semilog plots in figure 8 and 9 show the effect of changing the value of tol on Training Time and Accuracy for a LinearSVC model.

3.3.2 LogisticRegression

The semilog plots in figure 10 and 11 show the effect of changing the value of tol on Training Time and Accuracy for a LogisticRegression model.

3.4 Regularization (penalty) hyperparameter in LinearSVC and LogisticRegression

The penalty parameter specifies the norm used in the penalization term.

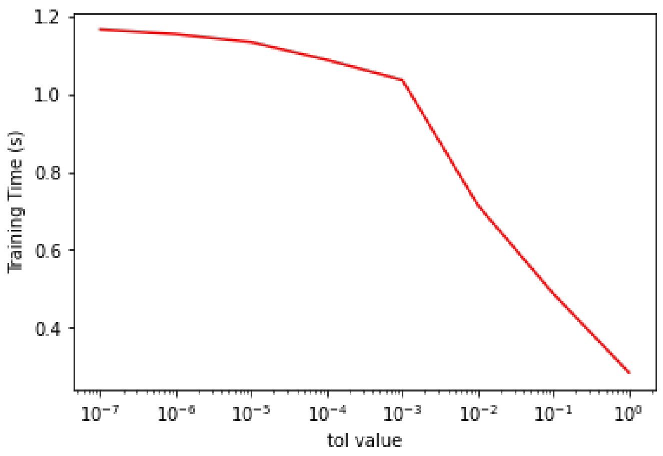


Figure 8: Effect of tol on Training Time for LinearSVC

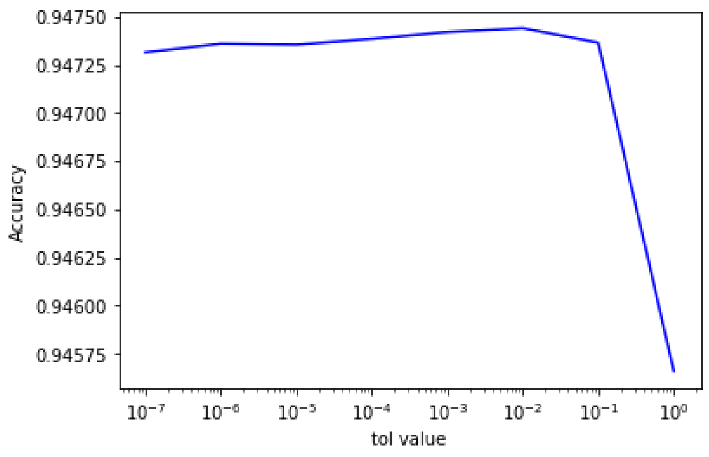


Figure 9: Effect of tol on Accuracy for LinearSVC

3.4.1 LinearSVC

LinearSVC does not allow the use of l1 style penalty with hinge or squared_hinge loss while solving the dual optimization problem, hence we set dual parameter to false for the experiments with LinearSVC. Table 2 summarises the results of the experiments.

Penalty Style	Training Time	Accuracy
l1	5.085	0.946
l2	0.263	0.947

Table 2: Effect of Penalty on LinearSVC Model

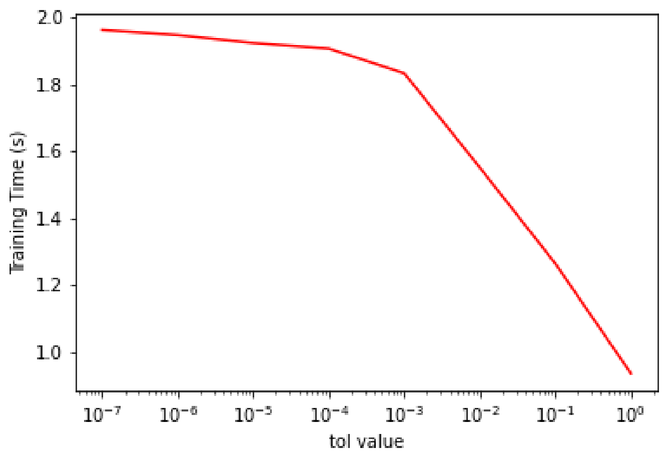


Figure 10: Effect of tol on Training Time for Logistic Regression

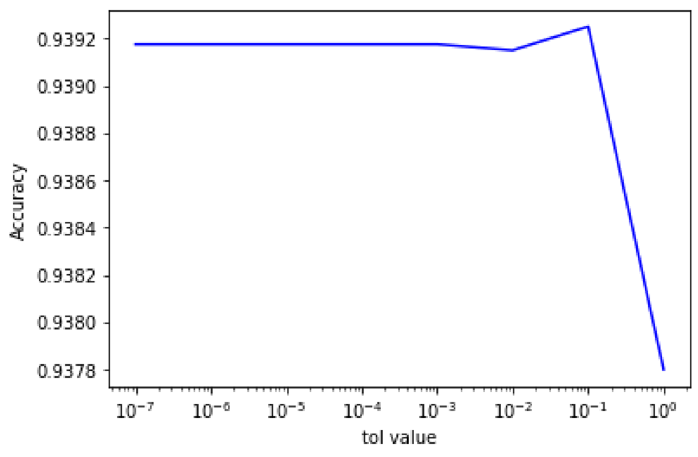


Figure 11: Effect of tol on Accuracy for Logistic Regression

3.4.2 LogisticRegression

LogisticRegression does not allow the use of l1 penalty with the default lbfgs solver, hence we use the liblinear solver for the experiments in this case. Table 3 summarises the results of the experiments.

Penalty Style	Training Time	Accuracy
l1	0.525	0.935
l2	0.283	0.940

Table 3: Effect of Penalty on LogisticRegression Model

4 Conclusion