CS771: Assignment 1

Kajal Deep - 200483 **Kuldeep Singh Chouhan - 200530** **Kumar Arpit - 200532**

Sandeep Kumar Bijarnia - 200856 **Yash Raj Mittal - 2001148**

1. By giving a detailed mathematical derivation (as given in the lecture slides), show how a simple XORRO PUF can be broken by a single linear model. Recall that the simple XORRO PUF has just two XORROs and has no select bits and no multiplexers (see above figure and discussion on Simple XORRO PUF). Thus, the challenge to a simple XORRO PUF has just R bits. More specifically, give derivations for a map $\phi : 0, 1^R \rightarrow \mathbb{R}^D$ mapping R-bit 0/1-valued challenge vectors to D-dimensional feature vectors (for some $D > 0$) and show that for any simple XORRO PUF, there exists a linear model i.e. $w \in \mathbb{R}^D, b \in R$ such that for all challenges $c \in 0, 1^R$, the following expression

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

gives the correct response.

Solution:

We start by considering only a single XORRO and try to calculate its time period.

We assume that the XORRO has achieved Equilibrium. Let's start with instance when the current Input is 0.

The delay created by 0^{th} XOR can formulated as:

$$\Delta_0^0 = (1 - a_0)\delta_{00}^0 + a_0\delta_{01}^0$$

The upper response for the next XOR gate will be XOR of both the previous inputs.

The delay created by 1^{th} XOR can formulated as:

$$\Delta_0^1 = (1 - a_1)\delta_{(0 \oplus a_0)0}^1 + a_1\delta_{(0 \oplus a_0)1}^1$$

Similarly, for i^{th} XOR delay can formulated as:

$$\Delta_0^i = (1 - a_i)\delta_{(0 \oplus a_0 \oplus .....a_{i-1})0}^i + a_i\delta_{(0 \oplus a_0 \oplus .....a_{i-1})1}^i$$

Hence, the total time taken by the input bit 0 to reach back to the first XOR gate as 1 is:

$$t_0 = \sum_{i=0}^{R-1} \Delta_0^i$$

Now, consider the case when initial input is 1.

Similar to above, the delay created by i^{th} XOR can formulated as:

$$\Delta_1^i = (1 - a_i)\delta_{(1 \oplus a_0 \oplus .....a_{i-1})0}^i + a_i\delta_{(1 \oplus a_0 \oplus .....a_{i-1})1}^i$$

Hence, the total time taken when the initial input is 1:

$$t_1 = \sum_{i=0}^{R-1} \Delta_1^i$$

The time period for XORRO Oscillation will be $t_0 + t_1$

$$T = t_0 + t_1 = \sum_{i=0}^{R-1} (\Delta_0^i + \Delta_1^i)$$
$$\implies T = \sum_{i=0}^{R-1} (1 - a_i)(\delta^i_{(0 \oplus a_0 \oplus .....a_{i-1})0} + \delta^i_{(1 \oplus a_0 \oplus .....a_{i-1})0})$$
$$+ a_i(\delta^i_{(0 \oplus a_0 \oplus .....a_{i-1})1} + \delta^i_{(1 \oplus a_0 \oplus .....a_{i-1})1})$$

Now, since it is guaranteed to us that exactly odd number of select bits would be one, we are sure that in a complete cycle both 0 and 1 arrive at the first XOR gate exactly once and consequently at eaxh XOR gate in the circuit. So, at each gate the delay would be $\Delta_i^0 + \Delta_i^1$ and thus we have the time period for the circuit as:

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

where

$$A = [a_0, a_0, \ldots, a_{R-1}]$$
$$D = [d_0, d_1, \ldots, d_{R-1}] \mid d_i = (\delta^i_{01} + \delta^i_{11} - \delta^i_{00} - \delta^i_{10})$$
$$b = \sum_{i=0}^{R-1} (\delta^i_{00} + \delta^i_{10})$$

Now, we consider a simple XORRO PUF. The simple XORRO PUF compares the frequency of the two PUFs and outputs accodingly. That is, if the upper XORRO has a higher frequency, it outputs 1 and 0 otherwise.

So, equivalently, if the upper XORRO has a lower time period then we output 1 and 0 otherwise. So, the output bit is simply $(Sign(T_2 - T_1) + 1)/2$ (where T_2 is the time period of the upper XORRO and T_1 of the lower XORRO). Finally, we have the output of a simple XORRO PUF as:

$$\frac{1 + sign(A^T(D_2 - D_1) + (b_2 - b_1))}{2}$$

where A, D_1, D_2, b_1, b_2 are as defined above.

Thus

$$\mathbf{w} = [a_0, a_0, \ldots, a_{R-1}]$$
$$\phi(c) = [d_0, d_1, \ldots, d_{R-1}] \mid d_i = ({}^2\delta^i_{01} - {}^1\delta^i_{01} + {}^2\delta^i_{11} - {}^1\delta^i_{11} - {}^2\delta^i_{00} + {}^1\delta^i_{00} - {}^2\delta^i_{10} + {}^1\delta^i_{10})$$
$$b = \sum_{i=0}^{R-1} ({}^2\delta^i_{00} - {}^1\delta^i_{10} + {}^2\delta^i_{10} - {}^1\delta^i_{00})$$

where the left superscript describes which PUF we are talking about, 1 or 2.

2. Show how to extend the above linear model to crack an Advanced XORRO PUF. Do this by treating an advanced XORRO PUF as a collection of multiple simple XORRO PUFs. For example, you may use $M = 2^{S-1}(2^S - 1)$ linear models, one for each pair of XORROs, to crack the advanced XORRO PUF.

   Solution:

   Now, we can extend the above linear model in multiple ways to crack the advanced XORRO PUF. The simple way would be to create a distinct model for each pair of PUFs (i.e. $M = 2^{S-1}(2^S - 1)$) and choosing the model depending on the select bits but we propose a more clever way here, involving a single linear model, albeit with a feature vector of higher dimension (equal to $2^S(R+1)$).

   Remember that for a single XORRO, we had the time period given,

$$T = A^T_{(1\times R)} D_{(R\times 1)} + b$$

   First, we remove the explicit bias term by including it inside D itself, increasing the dimension of our feature vector by 1. (Note that we retain the same notation for A and D for convenience).

$$T = A^T_{(1\times (R+1))} D_{((R+1)\times 1)} + b$$

   where,

$$A = [a_0, a_0, \ldots, a_{R-1}, 1]$$
$$D = [d_0, d_1, \ldots, d_{R-1}, 1]^T$$

   (d_i's and b were as defined above.)

   Now, let us consider the case when we have 2^S XORROs. Now, what we do essentially is a "one hot encoding of the A vector". What we mean by this is as follows,

$${}^i\mathbb{A} = [[0...0]_{1\times(R+1)}, [0...0]_{1\times(R+1)}, ...[a_0, a_1, ...a_{R-1}, 1]_{1\times(R+1)}...[0...0]_{1\times(R+1)}]_{(2^S(R+1)\times 1)}$$

   that is the first $i \times (R+1)$ bits are 0, followed by A, followed by remaining bits as 0. Hence, i is the decimal representation of the S select bits. And our weight vector $\mathbb{D}$ is given by:

$$\begin{aligned}\mathbb{D} &= [{}^0D^T, {}^1D^T, {}^2D^T, \ldots {}^{2^{S-1}}D^T]^T \\ &= [{}^0d_0, {}^0d_1, ...{}^0d_{R-1}, {}^0b, {}^1d_0, {}^1d_1, ...{}^1d_{R-1}, {}^1b, ...{}^{2^{S-1}}d_0, {}^{2^{S-1}}d_1, ...{}^{2^{S-1}}d_{R-1}, {}^{2^{S-1}}b]^T_{(2^S(R+1)\times 1)}\end{aligned}$$

   Note that the delay of i^{th} PUF is given simply by:

$$T = {}^i\mathbb{A}^T\mathbb{D}$$

   As,

$$\begin{aligned}T &= {}^i\mathbb{A}^T\mathbb{D} \\ &= [\mathbf{0}\ \mathbf{0} \ldots {}^iA \ldots \mathbf{0}][{}^0D\ {}^1D \ldots {}^iD \ldots {}^{2^{S-1}}D] \\ &= {}^iAD\end{aligned}$$

   Thus, the difference in its time period when 1^{st} 4 select bits represent p and next 4 represent q is

$$\begin{aligned}T^{pq} &= {}^p\mathbb{A}^T\mathbb{D} - {}^q\mathbb{A}^T\mathbb{D} \\ &= ({}^p\mathbb{A} - {}^q\mathbb{A})^T\,\mathbb{D}\end{aligned}$$

   And thus we get our linear model,

$$\frac{1 + sign(({}^p\mathbb{A} - {}^q\mathbb{A})^T\,\mathbb{D})}{2}$$

3. Write code to solve this problem by learning the M linear models $\mathbf{w}^1 \ldots \mathbf{w}^M$. You may use any linear classifier formulation e.g. LinearSVC, LogisticRegression, RidgeClassifier etc., to learn the linear model. However, the use of non-linear models is not allowed. You are allowed to use scikit-learn routines to learn the linear model you have chosen (i.e. you do not need to solve the SVM/LR/RC optimization problems yourself). Submit code for your chosen method in submit.py. Note that your code will need to implement at least 2 methods namely

   (a) $my\_fit()$ that should take CRPs for the advanced XORRO PUF as training data and learn the M linear models.

   (b) $my\_predict()$ that should take test challenges and predict responses on them.

   We will evaluate your method on a different dataset than the one we have given you and check how good is the method you submitted (see below for details). **Note that your learnt model must be a collection of one or more linear models.** The use of nonlinear models such as decision trees, random forests, nearest neighbors, neural networks, kernel SVMs etc is not allowed.

   Solution:

   [Code Attached]

4. Report outcomes of experiments with both the sklearn.svm.LinearSVC and sklearn.linear_model.LogisticRegression methods when used to learn the ensemble linear model. In particular, report how various hyperparameters affected training time and test accuracy using tables, charts. Report these experiments with both LinearSVC and LogisticRegression methods even if your own submission uses just one of these methods or some totally different linear model learning method e.g. RidgeClassifier) In particular, you must report the affect of training time and test accuracy of at least 2 of the following:

   (a) changing the loss hyperparameter in LinearSVC (hinge vs squared hinge)

   (b) setting C hyperparameter in LinearSVC and LogisticRegression to high/low/medium values

   (c) changing the tol hyperparameter in LinearSVC and LogisticRegression to high/low/medium values

   (d) changing the penalty (regularization) hyperparameter in LinearSVC and LogisticRegression ($l2$ vs $l1$)

   You may of course perform and report all the above experiments and/or additional experiments not mentioned above (e.g. changing the solver, max_iter etc) but reporting at least 2 of the above experiments is required. You do not need to submit code for these experiments - just report your findings in the PDF file. Your submitted code should only include your final method (e.g. learning M models using LinearSVC) with hyperparameter settings that you found to work the best.

   Solution:

   The table given below compares LinearSVC and LogisticRegression in default mode, $loss$ hyperparameter in LinearSVC, C hyperparameter in LinearSVC and LogisticRegression and tol hyperparameter in LinearSVC and LogisticRegression for their training time, testing time and accuracy.

Hyperparameters		LinearSVC			LogisticRegression		
		Training Time (in s)	Testing Time (in s)	Accuracy (in %)	Training Time (in s)	Testing Time (in s)	Accuracy (in %)
Default		4.74	0.78	99.07	13	0.78	98.75
loss	Hinge	7.26	1.66	98.72	-	-	-
	Squared Hinge	4.23	0.75	99.07	-	-	-
c	0.001	1.55	0.71	98.72	10.7	1.43	84.17
	0.1	3.2	0.76	98.72	20	1.34	97.68
	1	5.25	0.86	99.06	12.91	0.79	98.75
	10	3.87	0.78	99.16	16.08	2.85	99.15
	100	6.43	1.55	99	14.48	2.08	99.27
	1000	3.41	0.71	98.86	12.39	1.13	99.3
tol	0.00001	6.46	1.22	99.07	11.04	2.62	98.75
	0.0001	8.78	2.96	99.07	11.81	2.78	98.75
	0.001	7.63	2.63	99.08	14.76	2.67	98.75
	0.01	6.82	1.25	99.06	10.98	1.14	98.75
	0.1	5.96	1.11	99.07	11.61	2.76	98.75
	1	4.62	1.17	98.98	7.99	1.16	98.74
	10	5.23	2.81	94.43	5.88	1.41	98.71

   The following figures show the above variations graphically.

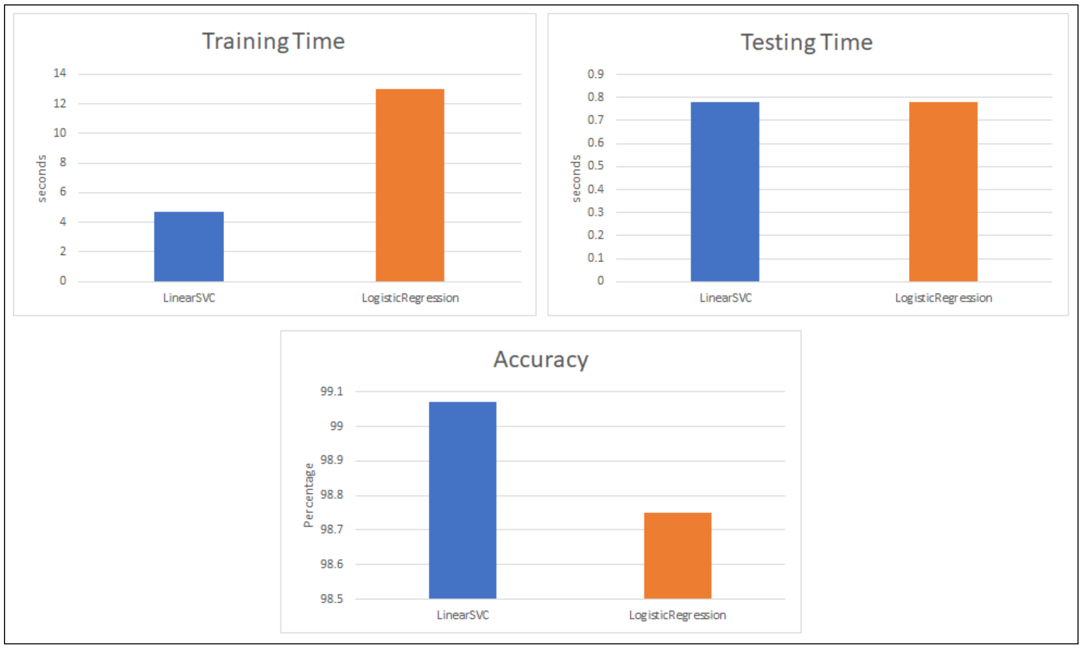

Figure 1: LinearSVC vs LogisticRegression

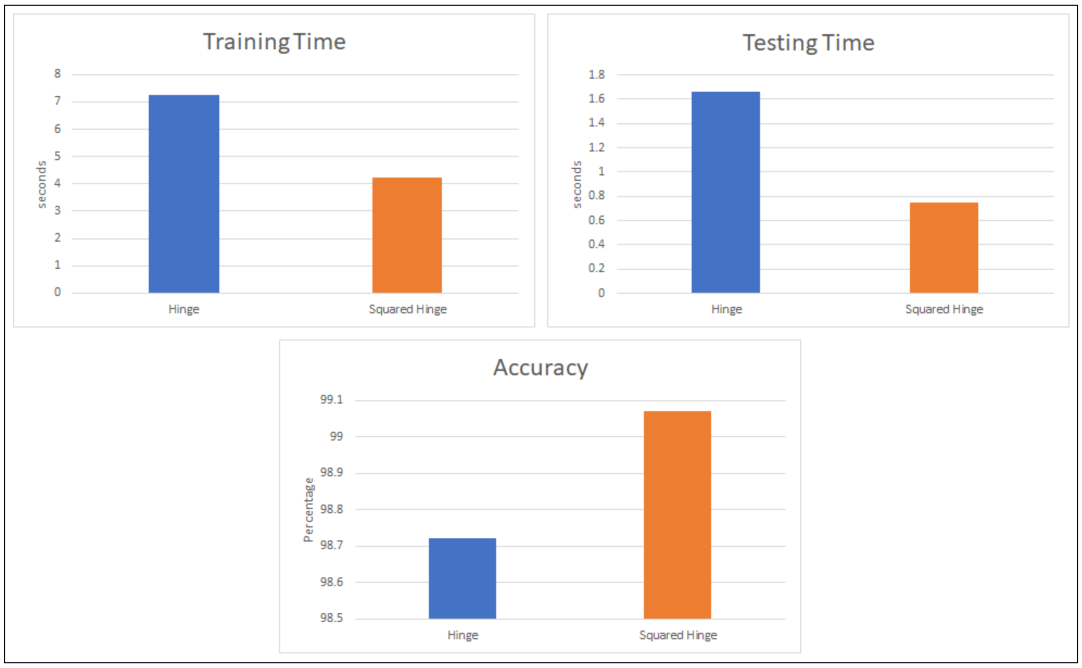

Figure 2: Hinge vs Squared Hinge

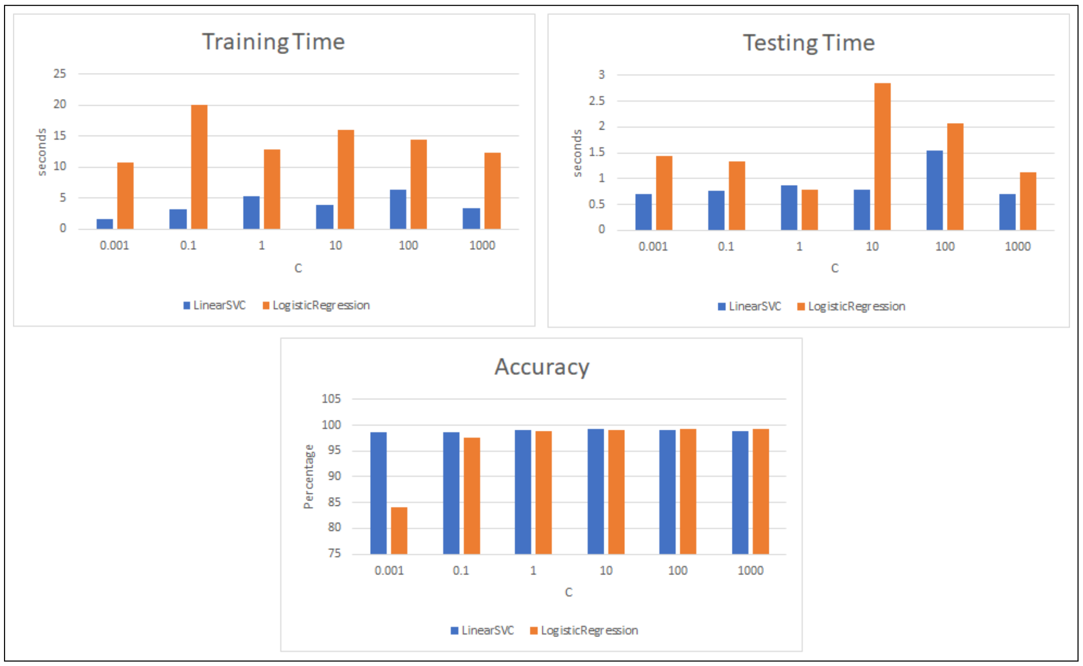

Figure 3: Varying hyperparameter - C

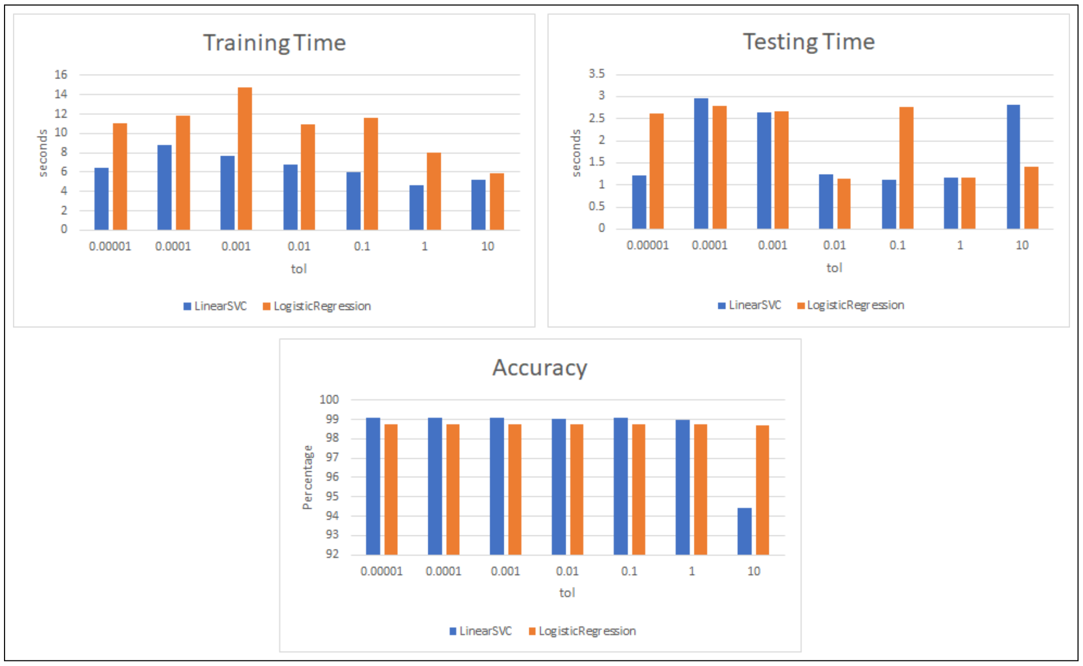

Figure 4: Varying hyperparameter - tol