CS771 Assignment 1

Ishan Bawne (200456) **Jhaansi Reddy** (200477) **Jahnavi Kairamkonda** (200482) **Mohil** (200596)

Sharath Kumar V (200916) **Poojitha** (201094)

1 Simple XORRO PUF

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. $\mathbf{w} \in \mathbb{R}^D$, $b \in \mathbb{R}$ such that for all challenges $c \in \{0,1\}^R$, the following expression

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

gives the correct response.

Solution:
We have R XOR's for each XORRO:

Let the delays corresponding to i^th XOR in a PUF be $\delta^i_{00}, \delta^i_{01}, \delta^i_{10}, \delta^i_{11}$ (For inputs corresponding to the inputs 00,01,10,11 respectively).

Suppose we start with input $0a_0$ for 1^{st} XOR:

$$t^0_0 = a_0 \cdot \delta^0_{01} + (1 - a_0) \cdot \delta^0_{00}$$

For input $1a_0$ for $1^{st} XOR$:

$$t^0_1 = a_0 \cdot \delta^0_{11} + (1 - a_0) \cdot \delta^0_{10}$$

This will similarly follow for every other XOR based on output of previous XOR:

$$t^i_0 = a_i \cdot \delta^i_{01} + (1 - a_i) \cdot \delta^i_{00} + t^{i-1}_j$$
$$t^i_1 = a_i \cdot \delta^i_{11} + (1 - a_i) \cdot \delta^i_{10} + t^{i-1}_{1-j}$$

j ∈ {0,1} based on input of $(i-1)^{th}$ XOR.

Suppose output of i^{th} XOR is P_i.

If output was 0 at some time , the input for following cycle would be
$< 0, P_0, P_1, P_2, \ldots, P_{R-2} >$

$P_{R-1} = 1$ (Because inversion is guaranteed)

The inputs for the next cycle (following the above mentioned) would be
$< 1, 1 - P_0, 1 - P_1, 1 - P_2, \ldots, 1 - P_{R-2} >$

Preprint. Under review.

Every input is flipped. The output for another cycle would again be 0 (1 is inverted).

Hence the overall time period would be $t_0^{R-1} + t_1^{R-1}$

$$T = t_0^{R-1} + t_1^{R-1} \tag{1}$$
$$= a_{R-1} \cdot (\delta_{01}^{R-1} + \delta_{11}^{R-1} - \delta_{00}^{R-1} - \delta_{10}^{R-1}) + \delta_{00}^{R-1} + \delta_{10}^{R-1} + t_0^{R-2} + t_1^{R-2} \tag{2}$$

Let
$\alpha_i = \delta_{01}^i + \delta_{11}^i - \delta_{00}^i - \delta_{10}^i$
$\beta_i = \delta_{00}^i + \delta_{10}^i$
substituting values for t_0^{R-2} and t_1^{R-2} in (2) , we get

$$T = \sum_{i=0}^{R-1} (a_i \cdot \alpha_i + \beta_i)$$

Let $\Delta = T_l - T_u$
Now , response for a XORRO PUF (with XORRO u(pper) and XORRO l(ower)) is

$$y = \begin{cases} 0 & f_u < f_l \\ 1 & f_u > f_l \end{cases}$$

$f = \frac{1}{T}$, so it means

$$y = \begin{cases} 0 & T_u > T_l \; or \; \Delta < 0 \\ 1 & T_u < T_l \; or \; \Delta > 0 \end{cases}$$

which simply means

$$y = \frac{1 + sign(\Delta)}{2}$$

$$\begin{aligned} \Delta &= T_l - T_u \\ &= \sum_{i=0}^{R-1} [a_i \cdot (\alpha_i^l - \alpha_i^u)] + \beta_i^l - \beta_i^u \\ &= \mathbf{w}^T \phi(c) + b \end{aligned}$$

where

$$\begin{aligned} \mathbf{w} &= [(\alpha_0^l - \alpha_0^u), (\alpha_1^l - \alpha_1^u), ..., (\alpha_{R-1}^l - \alpha_{R-1}^u)] \\ \phi(c) &= c \\ c &= [a_0, a_1, ..., a_{R-1}] \\ b &= \sum_{i=0}^{R-1} (\beta_i^l - \beta_i^u) \end{aligned}$$

2 Advanced XORRO PUF

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:
The above linear model can be trained for data sets of every pair of 2^S XORRO'S
Number of possible pairs of XORROs is $\binom{2^S}{2} = 2^{S-1}(2^S - 1)$
So we can have M = $2^{S-1}(2^S - 1)$ models for of each pair of all pssible XORRO PUFs.
If each of it is represented as m_{pq} where

$p = p_0 \cdot 2^0 + p_1 \cdot 2^1 + ... + p_{S-1} \cdot 2^{S-1}$

$q = q_0 \cdot 2^0 + q_1 \cdot 2^1 + ... + q_{S-1} \cdot 2^{S-1}$
$p < q$
then

$$m_{pq} = \begin{cases} \frac{1+sign(\mathbf{w}_{pq}^T \phi(c)+b_{pq})}{2} & \text{if p is first XORRO, q is second XORRO} \\ \frac{1-sign(\mathbf{w}_{pq}^T \phi(c)+b_{pq})}{2} & \text{if q is first XORRO, p is second XORRO} \end{cases}$$

[1] where

$$\begin{aligned} \mathbf{w}_{pq} &= [(\alpha_0^q - \alpha_0^p), (\alpha_1^q - \alpha_1^p), \ldots, (\alpha_{R-1}^q - \alpha_{R-1}^p)] \\ \phi(c) &= c \\ c &= [a_0, a_1, \ldots, a_{R-1}] \\ b_{pq} &= \sum_{i=0}^{R-1} (\beta_i^q - \beta_i^p) \end{aligned}$$

4 Outcomes of experiments

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)

Note: The values given in the tables are the results when the code was executed in a machine not on the Google Colab.

[1] p as first XORRO means that $[p_0, p_1, \ldots, p_{S-1}]$ is fed to upper MUX as select bits, p as second XORRO means that $[p_0, p_1, \ldots, p_{S-1}]$ is fed to lower MUX as select bits

Solution: (a)

Table 1: Changing loss hyperparameter

$loss \longrightarrow$	Hinge loss		Squared-hinge loss	
	Train-time	Test-accuracy	Train-time	Test-accuracy
LinearSVC	$3.5sec$	93.96%	$3.7sec$	94.74%

As show in the below table there is slight decrease in accuracy from squared hinge loss to hinge loss.

(b)

Table 2: Changing C hyperparameter

$C \longrightarrow$	Low(C=0.01)		Medium(C=1)		High(C=100)	
	Train-time	Test-accuracy	Train-time	Test-accuracy	Train-time	Test-accuracy
LinearSVC	$3.0sec$	90.39%	$3.8sec$	94.74%	$3.5sec$	93.92%
LogisticRegression	$3.2sec$	82.51%	$3.6sec$	93.92%	$4.8sec$	94.94%

In the value of Test-accuracy peaks for certain C. This is because for large values of C, the optimization will choose a 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. The peak of Test-accuracy in our case ocuurs when C is ≈ 1 in LinearSVC and ≈ 100 in LogisticRegression. The graphs obtained by changing the values of C is given below

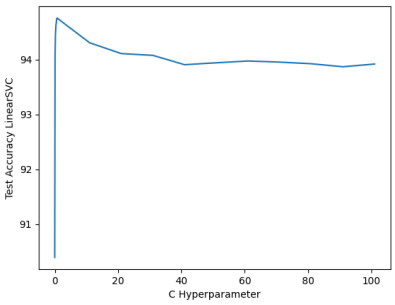

Figure 1: Changing C hyperparameter for LinearSVC

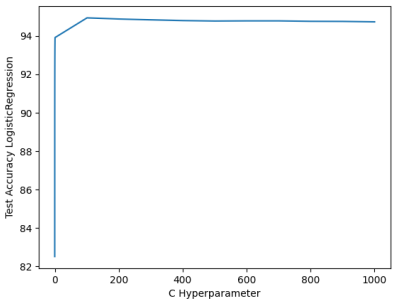

Figure 2: Changing C hyperparameter for LogisticRegression

(c)

Table 3: Changing tol hyperparameter

$tol \longrightarrow$	Low($tol = 10^{-6}$)		Medium($tol = 10^{-4}$)		High($tol = 10^{-2}$)	
	Train-time	Test-accuracy	Train-time	Test-accuracy	Train-time	Test-accuracy
LinearSVC	$3.9sec$	94.73%	$3.8sec$	94.74%	$3.4sec$	94.74%
LogisticRegression	$3.6sec$	93.91%	$3.6sec$	93.91%	$3.4sec$	93.91%

Test-accuracy remains almost the same till a certain value $\approx 10^{-1}$. Then it drastically decreases. The graphs obtained by changing the values of tol is given below

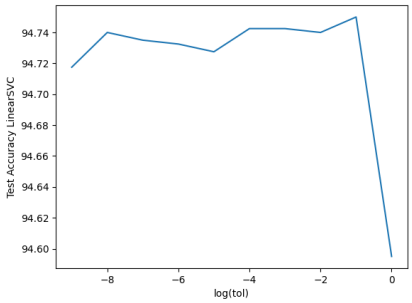

Figure 3: Changing tol hyperparameter for LinearSVC

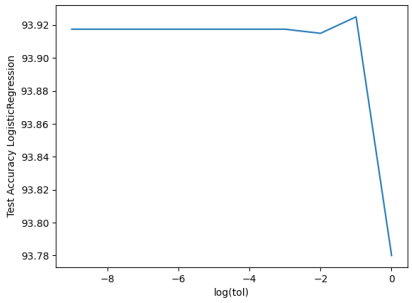

Figure 4: Changing tol hyperparameter for LogisticRegression

(d)

Table 4: Changing penalty

$penalty \longrightarrow$	l2		l1	
	Train-time	Test-accuracy	Train-time	Test-accuracy
LinearSVC	$3.8sec$	94.74%	$8.65sec$	94.59%
LogisticRegression	$3.6sec$	93.91%	$7.87sec$	93.44%

As seen in the above table there is a huge increase in Train-time when we change the penalty from $l2$ to $l1$ in both LinearSVC and LogisticRegression.