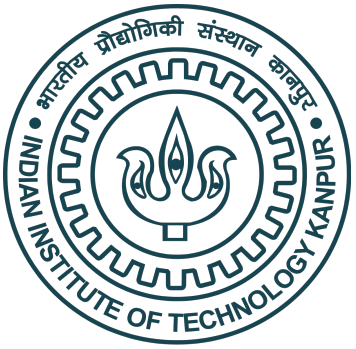

CS771A Assignment 1

Instructor

Dr. Purushottam Kar

Authors

200004	Aarchie	aarchi20@iitk.ac.in
200010	Aayushman	aayushn20@iitk.ac.in
200189	Arpit Kumar	arpitk20@iitk.ac.in
200964	Shubh Tandon	shubht20@iitk.ac.in
201055	Udit Prasad	uditp20@iitk.ac.in

Feb 18, 2023

Group Representative - Udit Prasad

Contents

1 Question 1

1.1 Solution

Given,
$\delta^i_{j_1 j_2}$ be the time taken by ith XOR to give output when input to it is j_1, j_2
As, we have 2 XORROs,
We can denote $\delta^i_{j_1 j_2}[\ 1]$ for upper XORRO, say it is XORRO1 & similarly $\delta^i_{j_1 j_2}[\ 2]$ for lower XORRO, XORRO_2.
Let,
t^1_0 = Time taken by XORRO1 to flip the output from 0 to 1.
t^1_1 = Time taken by XORRO1 to flip the output from 1 to 0.
t^2_0 = Time taken by XORRO2 to flip the output from 0 to 1.
t^2_1 = Time taken by XORRO2 to flip the output from 1 to 0.

So, frequency of XORRO1(f_1) = $\frac{1}{t^1_0+t^1_1}$
So, frequency of XORRO2(f_2) = $\frac{1}{t^2_0+t^2_1}$
We are mainly interested in the difference of $f_1 \& f_2$
If $f_1 > f_2 \implies$ Counter outputs 1.
If $f_2 > f_1 \implies$ Counter outputs 0.

$$f_1 - f_2 = \frac{1}{t^1_0 + t^1_1} - \frac{1}{t^2_0 + t^2_1} = \frac{t^2_0 + t^2_1 - (\ t^1_0 + t^1_1)}{(\ t^1_0 + t^1_1)\ (\ t^2_0 + t^2_1)}$$

But, we are interested in sign $(f_1 - f_2)$.
So, if $(\ t^2_0 + t^2_1)\ -(\ t^1_0 + t^1_1)\ > 0 \implies$ output is 1. -(i)
else if $(\ t^2_0 + t^2_1)\ -(\ t^1_0 + t^1_1)\ < 0 \implies$ output is 0. -(ii)
Now, we will calculate this.
$\delta^i_{j_1 j_2}, j_i$ will be configuration signal (one from $(a_0, a_1, ..., a_{63})$)$\& j_2$ will be the output of the previous XOR.
Let Δ^j_i represents difference in the time taken by XORRO2 & XORRO1 till i^{th} XOR if the input is j.
Induction Hypotheis:

$$Z(\ i)\ = \Delta^0_i + \Delta^1_i = \delta^0_{a_0 0}[\ 2]\ + \delta^0_{a_0 1}[\ 2]\ - \delta^0_{a_0 0}[\ 1]\ - \delta^0_{a_0 1}[\ 1]$$

$$\delta^1_{a_1 1}[\ 2]\ + \delta^1_{a_1 0}[\ 2]\ - \delta^1_{a_1 1}[\ 1]\ - \delta^1_{a_1 1}[\ 1]$$

$$+.....+$$

$$\delta^i_{a_i0}[\ 2]\ + \delta^i_{a_i1}[\ 2]\ - \delta^i_{a_i0}[\ 1]\ - \delta^i_{a_i1}[\ 1]$$

Base case:
$\Delta^0_0 + \Delta^1_0$
$\Delta^0_0 = \delta^0_{a_00}[\ 2]\ - \delta^0_{a_00}[\ 1]$
$\Delta^1_0 = \delta^0_{a_01}[\ 2]\ - \delta^0_{a_00}[\ 1]$ so, the above hypothesis is valid for base case.
Proved.
Induction Step:
Say induction hypothesis holds for i,
then we need to prove, Z(i+1)=Z(i)+$\delta^{i+1}_{a_{i+1}1}[\ 2]\ + \delta^{i+1}_{a_{i+1}0}[\ 2]\ - \delta^{i+1}_{a_{i+1}0}[\ 1]\ - \delta^{i+1}_{a_{i+1}1}[\ 1]$
As $\delta^i_{a_i\alpha}[\ j]$ means jth XORRO's ith XOR gets input as a_i and α, So input to (i+1)th XOR will be a_i XOR α.
We have terms $\delta^i_{a_i0}[\ 2]\ + \delta^i_{a_i1}[\ 2]\ - \delta^i_{a_i0}[\ 1]\ - \delta^i_{a_i1}[\ 1]$
Thus, we will get the terms, (i) a_i XOR 0 (ii) a_i XOR 1 as input to i+1th XOR in both the XORROS i.e. XORRO1 and XORRO2,

Case 1: if $a_i = 0 \implies$
(a_i XOR $0 = 0$) $\implies$ This will add up $\delta^{i+1}_{a_{i+1}0}[\ 2]\ - \delta^{i+1}_{a_{i+1}0}[\ 1]$ to Z(i)
(a_i XOR $1 = 1$) $\implies$ This will add up $\delta^{i+1}_{a_{i+1}1}[\ 2]\ - \delta^{i+1}_{a_{i+1}1}[\ 1]$ to Z(i)

Case 2: $a_i = 1$
(a_i XOR $1 = 0$) $\implies$ This will give $\delta^{i+1}_{a_{i+1}0}[\ 2]\ - \delta^{i+1}_{a_{i+1}0}[\ 1]$,
(a_i XOR $0 = 1$) $\implies$ This will give $\delta^{i+1}_{a_{i+1}1}[\ 2]\ - \delta^{i+1}_{a_{i+1}1}[\ 1]$,
Hence for both cases,

Z(i+1)=Z(i)+$\delta^{i+1}_{a_{i+1}1}[\ 2]\ + \delta^{i+1}_{a_{i+1}0}[\ 2]\ - \delta^{i+1}_{a_{i+1}0}[\ 1]\ - \delta^{i+1}_{a_{i+1}1}[\ 1]$ **Proved**

So, $Z[\ 63]\ = \Delta^0_{63} + \Delta^1_{63}$,
Also, $\Delta^0_{63} = t^2_0 - t^1_0$ and
$\Delta^1_{63} = t^2_1 - t^1_1$
So, $\Delta^0_{63} + \Delta^1_{63} = t^2_1 + t^2_0 - t^1_0 - t^1_1$
Using (i) and (ii),
$\Delta^0_{63} + \Delta^1_{63} > 0 \implies$ output is 1.
$\Delta^0_{63} + \Delta^1_{63} < 0 \implies$ output is 0.

Let $P^1_i = \delta^i_{00}[\ 1]\ + \delta^i_{01}[\ 1]$

$P_i^2 = \delta_{00}^i[\ 2]\ + \delta_{01}^i[\ 2]$
$Q_i^1 = \delta_{10}^i[\ 1]\ + \delta_{11}^i[\ 1]$
$Q_i^2 = \delta_{10}^i[\ 2]\ + \delta_{11}^i[\ 2]$
We can see,
$Z_0 + \Delta_0 = P_i^2 - P_i^1$ (If a_0=0)
$Z_0 = Q_i^2 - Q_i^1$ (If $a_0 = 1$) We can say,
$Z(0) = (1 - a_0)(P_0^2 - P_0^1) + a_0(Q_0^2 - Q_0^1)$
$Z(1) = Z(0) + (1 - a_0)(P_0^2 - P_0^1) + a_0(Q_0^2 - Q_0^1)$
.
.
.
$Z(63) = Z(62) + (1 - a_0)(P_{63}^2 - P_{63}^1) + a_0(Q_{63}^2 - Q_{63}^1)$
$Z(0) = a_0(Q_0^2 - Q_0^1 + P_0^2 - P_0^1) + P_0^2 - P_0^1$
Let $\alpha_i = Q_i^2 - Q_i^1 + P_i^1 - P_i^2$
$\beta_i = P_i^2 - P_i^1$
So, $Z(0) = \frac{a_0\alpha_0+\beta_0}{2}$
Z(1)=Z(0)+$a_1\alpha_1 + \beta_1 = a_0\alpha_0 + \beta_0 + a_1\alpha_1 + \beta_1$
.
.
.
Z(63)=Z(62)+$a_{63}\alpha_{63} + \beta_{63} = a_0\alpha_0 + a_1\alpha_1 + ... + a_{63}\alpha_{63} + \beta_1 + \beta_2 + ... + \beta_{63}$.
Z(63)=$w^T\phi$(c) + B
where $w = [\ \alpha_0\alpha_1\alpha_2...\alpha_{63}]$
$\phi(c) = c$
B=$\beta_1 + \beta_2 + ... + \beta_{63}$
So, this is of the form,

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

□

2 Question 2

2.1 Solution

Advanced XORRO-
no. of challenge bits= R+2S
where R=bits for each simple XORRO PUF ; S= bits for selection of XORRO
For each pair of XORRO PUF, we can
As there are 2^s XORROs so total number of pairs= ${}^{2^s}C_2 = 2^{s-1}(\ 2^s - 1\ )$ pairs
we can set i^{th} counter=

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

which outputs 0 if XORRO selected by lower MUX has higher frequency and 1 in vice-versa case

$$sign(w_i^T\phi(c) + B_i) = \left\{ \begin{array}{ll} +1, & \text{if upper MUX has higher frequency} \\ -1, & \text{otherwise} \end{array} \right\}$$

the choice of upper or lower XORRO depends on first S bits of the challenge bit data
Thus we can deploy $M = 2^{s-1}(\ 2^s - 1\ )$ linear models to each pair of XORRO PUF, and hence we can break the advanced PUF

We can do so by ,

Let Δ'_i represents Z(63) (prev question) for i^{th} pair of XORRO,
here i = 16* index of first XORRO + index of second XORRO
i.e. for pair of k^{th} XORRO and l^{th} XORRO, $i = 16 * k + l$
So, the response will be dictated by Δ'_{16*k+l} if $r_0r_1r_2r_3$ represents k and $r_4r_5r_6r_7$ represents l
if $\Delta'_{16*k+l} > 0 \implies$ counter will output 1
else if $\Delta'_{16*k+l} < 0 \implies$ counter will output 0
Say, we have t = $t_0t_1...t_7$ where t_i can take values 0 and 1
$(1 - (t_0 \oplus r_0))\ \&\ (1 - (t_1 \oplus r_1))\ \&\ (1 - (t_1 \oplus r_1))...\ \&\ (1 - (t_7 \oplus r_7)) = 1$ iff t=r
t can also take values from 0 to 255 (8 bit binary)
We can rewrite this

$$\sum_{t=0}^{255}\prod_{i=0}^{7}(1 - (t_i \oplus r_i)) = 1$$

If, a and b are binary, We can write $a \oplus b = (a - b)^2$

Thus,

$$\sum_{t=0}^{255}\prod_{i=0}^{7}(1-((t_i-r_i)^2))=1$$

Hence,

$$Z'=\sum_{t=0}^{255}(\prod_{i=0}^{7}(1-((t_i-r_i)^2)))\Delta'_t=\Delta'_t$$

where Δ'_t is linear in terms of $a_0, a_1, a_2, ...a_63$(from prev question)
Which means (1 + sign (Z'))/2 will be our required expression

3 Question 4

3.1 Solution:

We are going to analyze the loss and tol hyperparameters.

(a) LinearSVC: all hyperparameters with default value except **loss**

Loss	Training time	Test time	Accuracy
Hinge	3.085 s	2.487 s	93.95 %
Squared Hinge	3.350 s	2.661 s	94.74 %

The hinge loss penalizes linearly whereas Squared Hinge loss penalizes quadratically, therefore Squared hinge loss model is more robust and provides better accuracy atleast in the given problem.

(c)

(i) LinearSVC : All hyperparameters with default value except **tol**

tol	Training time	Test time	Accuracy
0.000001	3.487 s	3.005 s	94.73 %
0.00001	3.519 s	2.676 s	94.73 %
0.0001	3.350 s	2.661 s	94.74 %
0.01	2.418 s	2.990 s	94.74 %
0.1	2.252 s	2.536 s	94.74 %
1.0	1.686 s	2.716 s	94.57 %
5.0	1.664 s	2.680 s	87.75 %

(ii) LogisticRegression : All hyperparameters with default value except **tol**

tol	Training time	Test time	Accuracy
0.000001	3.802 s	2.611 s	93.91 %
0.00001	3.923 s	2.421 s	93.91 %
0.001	3.886 s	2.670 s	93.91 %
0.01	3.500 s	2.589 s	93.91 %
0.1	3.122 s	2.645 s	93.92 %
1	2.804 s	2.684 s	93.78 %
10	2.066 s	3.070 s	91.12 %

The tol hyperparameter in LinearSVC implementation controls the tolerance of the stopping criterion for the solver.
From the analysis of the tables, we can see

1. As **tol** becomes smaller, the solver need to run for more iterations to converge, which results in longer training times. This is because a smaller tol value requires a more precise solution, which may take longer to compute.

2. When **tol** becomes large enough, the accuracy decreases abruptly, but reducing tol beyond some limit increases the accuracy very marginally.