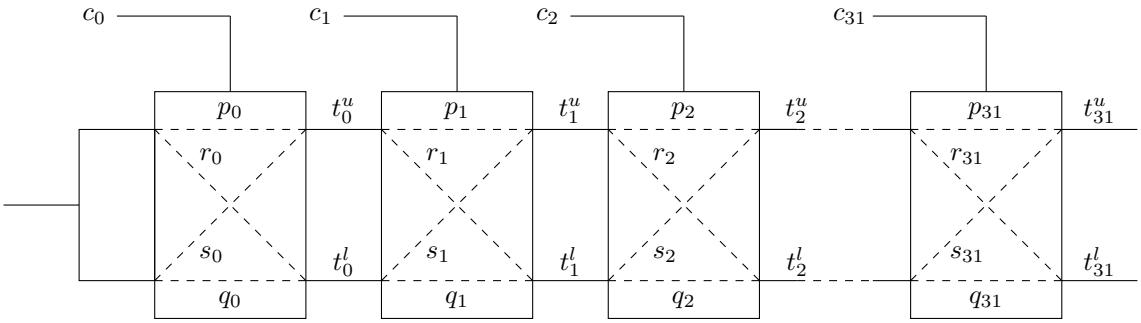
CS771A Assignment-1

Group No: 57
Team **Lambda**
Priyanshu Choudhary(210781)
Harsh Khandelwal(210407)
Paritosh Pankaj(210702)
Aman Chahr(210107)
Ashray Narayan(210218)

1 Problem 1.1

1.1 Part 1

Here, we will try to present an analytical proof that melbo could learn a linear model which accurately predicts the classfication results of the proposed Companion ArbiteR PUF (CAR-PUF) problem.



We are using a 32-bit challenge and all Mux are different.

So, $p_1 \neq p_2 \neq p_3 \neq \ldots \neq q_1 \neq q_2 \neq q_3 \neq \ldots$.

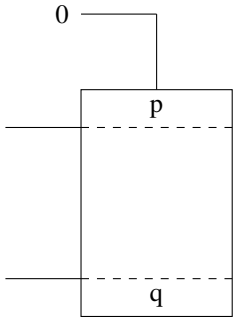
$t_i^l =$ Time taken by the upper signal to reach i^{th} Mux.

$t_i^u =$ Time taken by the lower signal to reach i^{th} Mux.

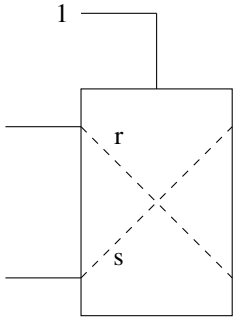
$i \in [0, 31]$, $c_i \in \{0, 1\}$.

if $c_i = 0$ then the signal passes directly.

Preprint. Under review.



if $c_i = 1$ then the signal alters.



Note

t_1^l and t_1^u depend on $t_0^l, t_0^u, c_1, p_1, q_1, r_1, s_1$, and so on.

c_1 dictates which provides delay t_0^u, t_0^l will get carried forward.

$p_i, q_i, r_i, s_i \rightarrow$ gives delay in i^{th} Mux itself.

For 1^{st} Mux

$t_1^u = (1 - c_1)(t_0^u + p_1) + c_1(t_0^l + s_1)$.

$t_1^l = (1 - c_1)(t_0^l + q_1) + c_1(t_0^u + r_1)$.

Let $\Delta_i = t_i^u - t_i^l$ denote the lag.

if $\Delta_i < 0$, it means the upper signal reaches first.

if $\Delta_i > 0$, it means the lower signal reaches first.

:

$$
\begin{aligned}
\Delta_1 &= t_1^u - t_1^l \\
&= (1 - c_1)(t_0^u + p_1 - t_0^l - q_1) + c_1(t_0^l + s_1 - t_0^u - r_1) \\
&= (1 - c_1)(\Delta_0 + p_1 - q_1) + c_1(-\Delta_0 + s_1 - r_1) \\
&= (1 - 2c_1)\Delta_0 + (q_1 - p_1 + s_1 - r_1)c_1 + (p_1 - q_1) \\
&= (1 - 2c_1)\Delta_0 - \frac{1}{2}(p_1 - q_1 - s_1 + r_1)2c_1 + (p_1 - q_1) \\
&= (1 - 2c_1)\Delta_0 - \frac{1}{2}(p_1 - q_1 - s_1 + r_1)2c_1 + (p_1 - q_1) \\
&+ \frac{1}{2}(p_1 - q_1 - s_1 + r_1) - \frac{1}{2}(p_1 - q_1 - s_1 + r_1) \\
&= (1 - 2c_1)\Delta_0 + \frac{1}{2}(p_1 - q_1 - s_1 + r_1)(1 - 2c_1) + \\
&\frac{1}{2}(p_1 - q_1 - r_1 + s_1)
\end{aligned}
$$

To make Notation simpler.

Let, $d_i = (1 - 2c_i)$

$\Delta_1 = \Delta_0 \cdot d_1 + \alpha_1 \cdot d_1 + \beta_1$

where $\alpha_1 = \frac{p_1 - q_1 + r_1 - s_1}{2}$, $\beta_1 = \frac{p_1 - q_1 + r_1 - s_1}{2}$.

We can safely take $\Delta_{-1} = 0$ (absorb initial delays into p_0, q_0, r_0, s_0).

We can observe pattern by performing same procedure recursively:

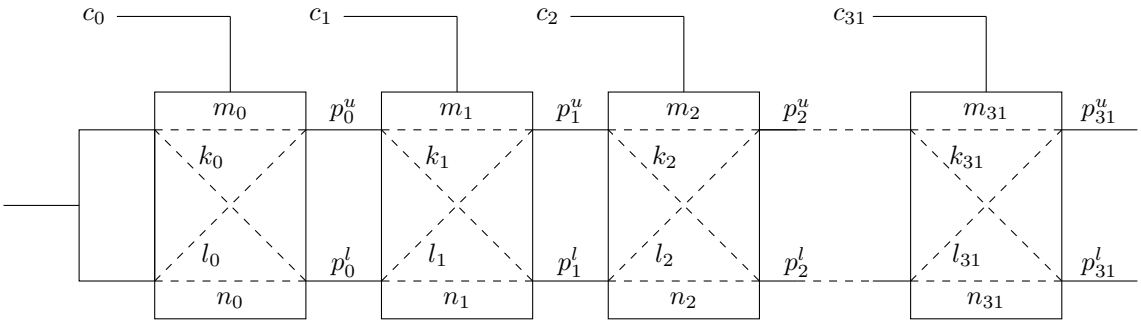
$$\Delta_0 = \alpha_0 \cdot d_0 + \beta_0 \ (since\, \Delta_{-1} = 0)$$
$$\Delta_1 = \Delta_0 \cdot d_1 + \alpha_1 \cdot d_1 + \beta_1 \ (\, now\, plugin\, value\, of\, \Delta_0)$$
$$\Delta_1 = \alpha_0 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_1 + \beta_1$$
$$\Delta_2 = \alpha_0 \cdot d_2 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1) \cdot d_2 + \beta_2$$

Now,

$$\begin{aligned} \Delta_{31} &= w_0 \cdot x_0 + w_1 \cdot x_1 + \ldots + w_{31} \cdot x_{31} + \beta_{31} \\ &= \mathbf{w}^\top \mathbf{x} + b \\ Where,\ x_i &= d_i \cdot d_{i+1} \cdot \ldots \cdot d_{31} \\ w_0 &= \alpha_0 \\ w_i &= \alpha_i + \beta_{i-1}\ for (i > 0) \end{aligned}$$

For Working Puf, We represent perimeter vector w with u
so,
$\Delta_w = u^T + b$,
where, b is biasterm

Reference PUF



We are using a 32-bit challenge. The steps would be similar as above:

All Mux are different.

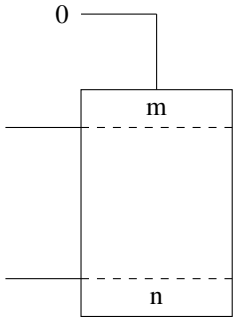
So, $m_1 \neq m_2 \neq m_3 \neq \ldots \neq n_1 \neq n_2 \neq n_3 \neq \ldots$.

$p_i^l =$ Time taken by the upper signal to reach i^{th} Mux.

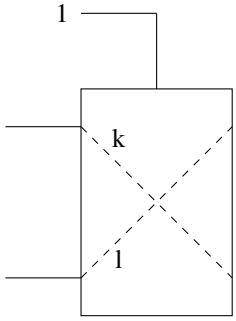
$p_i^u =$ Time taken by the lower signal to reach i^{th} Mux.

$i \in [0, 31]$, $c_i \in \{0, 1\}$.

if $c_i = 0$ then the signal passes directly.



if$c_i = 1$ then the signal alters.



Note

p_1^l and p_1^u depend on $p_0^l, p_0^u, c_1, m_1, n_1, k_1, l_1$, and so on.
c_1 dictates which provides delay p_0^u, p_0^l will get carried forward.
$m_i, n_i, k_i, l_i \rightarrow$ gives delay in i^{th} Mux itself.

For 1^{st} Mux

$p_1^u = (1 - c_1)(p_0^u + m_1) + c_1(p_0^l + l_1)$.
$p_1^l = (1 - c_1)(p_0^l + n_1) + c_1(p_0^u + k_1)$.
Let $\Delta_i = p_i^u - p_i^l$ denote the lag.
if $\Delta_i < 0$, it means the upper signal reaches first.
if $\Delta_i > 0$, it means the lower signal reaches first.
:

$$
\begin{aligned}
\Delta_1 &= p_1^u - p_1^l \\
&= (1 - c_1)(p_0^u + m_1 - p_0^l - n_1) + c_1(p_0^l + l_1 - p_0^u - k_1) \\
&= (1 - c_1)(\Delta_0 + m_1 - n_1) + c_1(-\Delta_0 + l_1 - k_1) \\
&= (1 - 2c_1)\Delta_0 + (n_1 - m_1 + l_1 - k_1)c_1 + (m_1 - n_1) \\
&= (1 - 2c_1)\Delta_0 - \frac{1}{2}(m_1 - n_1 - l_1 + k_1)2c_1 + (m_1 - n_1) \\
&= (1 - 2c_1)\Delta_0 - \frac{1}{2}(m_1 - n_1 - l_1 + k_1)2c_1 + (m_1 - n_1) \\
&+ \frac{1}{2}(m_1 - n_1 - l_1 + k_1) - \frac{1}{2}(m_1 - n_1 - l_1 + k_1) \\
&= (1 - 2c_1)\Delta_0 + \frac{1}{2}(m_1 - n_1 - l_1 + k_1)(1 - 2c_1) + \\
&\frac{1}{2}(m_1 - n_1 - k_1 + l_1)
\end{aligned}
$$

To make Notation simpler.

Let, $d_i = (1 - 2c_i)$

$\Delta_1 = \Delta_0 \cdot d_1 + \alpha_1 \cdot d_1 + \beta_1$

where $\alpha_1 = \frac{m_1 - n_1 + k_1 - l_1}{2}$, $\beta_1 = \frac{m_1 - n_1 + k_1 - l_1}{2}$.

We can safely take $\Delta_{-1} = 0$ (absorb initial delays into m_0, n_0, k_0, l_0).

We can observe pattern by performing same procedure recursively:

$$\begin{aligned}
&\Delta_0 = \alpha_0 \cdot d_0 + \beta_0 \ (since\, \Delta_{-1} = 0) \\
&\Delta_1 = \Delta_0 \cdot d_1 + \alpha_1 \cdot d_1 + \beta_1 \ (\, now\, plugin\, value\, of\, \Delta_0) \\
&\Delta_1 = \alpha_0 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_1 + \beta_1 \\
&\Delta_2 = \alpha_0 \cdot d_2 \cdot d_1 \cdot d_0 + (\alpha_1 + \beta_0) \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1) \cdot d_2 + \beta_2
\end{aligned}$$

Now,

$$\begin{aligned}
\Delta_{31} &= w_0 \cdot x_0 + w_1 \cdot x_1 + \ldots + w_{31} \cdot x_{31} + \beta_{31} \\
&= \mathbf{w}^\top \mathbf{x} + b \\
Where,\ x_i &= d_i \cdot d_{i+1} \cdot \ldots \cdot d_{31} \\
w_0 &= r_0 \\
w_i &= r_i + \delta_{i-1}\ for (i > 0)
\end{aligned}$$

For Reference Puf, We represent perimeter vector w with v
so,
$\Delta_r = v^T + h$,
where, h is biasterm

$$\begin{aligned}
\Delta_w - \Delta_r &= (u - v)^T X + (b - h) \\
&= A^T X + Z \\
A^T &= (u - v)^T \\
Z &= (b - h)
\end{aligned}$$

- **NOW,** $\Delta_w = u^T + b$

  $\Delta_r = v^T + h$

$$|\Delta_w - \Delta_r| > \tau$$
$$|\Delta_w - \Delta_r|^2 > \tau^2$$
$$|(u-v)^T X + (b-h)|^2 > \tau^2$$

Let, $(v-u)^T = w^T$

$$(b-h) = p$$
$$|(w^T X + p)|^2 > \tau^2$$
$$((\sum w_i x_i) + p)^2 > \tau^2$$
$$(\sum w_i x_i)^2 + p^2 + 2p(\sum w_i x_i) - \tau^2 > 0$$
$$\sum (w_i x_i)^2 + p^2 + \sum_i \sum_j 2 w_i w_j x_i x_j + 2p(\sum w_i x_i) - \tau^2 > 0$$
$$x_i = +1, -1$$
$$X_i^2 = 1$$
$$\sum (w_i)^2 + p^2 + \sum_i \sum_j 2 w_i w_j x_i x_j + \sum 2 p w_i x_i - \tau^2 > 0$$
$$\sum_i \sum_j 2 w_i w_j x_i x_j + \sum 2 p w_i x_i + [\sum (w_i)^2 + p^2 - \tau^2] > 0$$
$$\tilde{w}^T \tilde{x} + \tilde{b} > 0$$

Where, $\tilde{w}^T = [2bw_0, 2bw_1, 2bw_2, \ldots, 2w_0w_1, 2w_0w_2, \ldots, 2w_iw_j]$

$$\tilde{x} = [x_0, x_1, \ldots, x_{31}, x_0x_1, x_0x_2, \ldots, x_ix_j]$$
$$\tilde{b} = \sum (w_i)^2 + b^2 - \tau^2$$

- The above proof clearly shows that a linear model could be learnt to solve the classification problem for CAR-PUF. The features vector, learnt weights and bias of the linear model and their corresponding dimensions are described below:
- $\tilde{w}$: Dimension: $32 + \frac{32 \times 31}{2} = 528$
- $\tilde{x} = 32 + \frac{32 \times 31}{2} = 528$
- $\tilde{w} \in R^{528}$, $\tilde{x} \in R^{528}$, $\tilde{b} \in R$
- Map $\Phi(c) = \tilde{x}$
- $\frac{1+\text{sign}(\tilde{w}^T \Phi(c) + \tilde{b})}{2} = r$

1.2 Part 2

1.2.1 Map creation

This section contains the details of how we created the map for our training data as a feature extraction step. In order to create the map, we first converted the training data to matrix of 1s and -1s instead of 1s and 0s, which is more suitable format for binary classification. Then we created the map as described in the Part 1, although we did not use the Khatri Rao product of the scipy package, and created the map using custom code.

1.2.2 Learning the Linear model

In order to learn the linear model, we have used the LogisticRegression class of the sklearn module with a relatively high value of the regularization parameter (C=100), The above model yields an accuracy of 99.31 percent when fed in the validation code provided to us.

1.3 Part 3

Now we will present the results of what happens when we are using LinearSVC and Logistic Regression to learn the linear model. We have presented a table of results to indicate how the model accuracy and other benchmarks vary when we alter various hyperparameters of the linear models namely, Loss Hyperparameter for LinearSVC and Value of C LinearSVC and LogisticRegression to high/low/medium values

- **Changing the loss hyperparameter in LinearSVC (hinge vs squared hinge)**

Loss Hyperparameter	t_train (in s)	t_map (in s)	Accuracy(in %)
Hinge Loss	13.15	0.08	99.09
Squared Hinge Loss	13.1	0.09	99.23

- **Setting C in LinearSVC to high/low/medium values**

Regularization parameter(C)	t_train (in s)	t_map (in s)	Accuracy(in %)
C= 0.01 (low)	5.29	0.06	99.07
C= 1.0 (mid)	12.7	0.06	99.2
C= 100.0 (high)	11.91	0.05	99.04

- **Setting C in LogisticRegression to high/low/medium values**

Regularization parameter(C)	t_train (in s)	t_map (in s)	Accuracy(in %)
C= 0.01 (low)	2.11	0.12	97.4
C= 1.0 (mid)	1.96	0.08	99.1
C= 100.0 (high)	2.48	0.07	99.31

We could easily conclude that for the case of loss hyperparameter of LinearSVC, the performance of squared hinge loss is slightly better than hinge loss in terms of model accuracy (both approaches take equivalent time to train).

When it comes to varying the regularization hyperparameter, for Logistic , choosing a very small value of regularization parameter leads to underfitting of data to some extent and the accuracy is less than 99 percent in those cases. As we gradually increase the value of the regularization hyperparameter, the underfitting and bias of the model reduces and the accuracy of both the model improves(although the training time of the linear model also increases). For linearSVC, medium value of regularization hyperparameter is seen as optimal as a small value leads to underfitting and a large value leads to slight overfitting, hence medium value should be preferred. The LinearSVC in general takes more time to train than the LogisitcRegression model.

THANK YOU!!