CS771: Introduction to Machine Learning

Assignment - 1

1 Derivation for a Simple Arbiter PUF

We will use a simple approach to solve this problem. The method involves finding the differences in time delays for the upper and lower signals, as well as the sum of their delays, to predict the delay in the upper signal.

First, we will derive the difference in time delays for a 32-bit arbiter PUF in a way similar to as outlined in the lectures, and then proceed to derive the sum of these delays.

Also the notations are the same as described in lectures.

1.1 Deriving the difference of Delays

We can see that, t_i^u and t_i^l represent the times at which the signal departs from the ith Multiplexer (MUX) pair. Expressing t_i^u in terms of t_{i-1}^u, t_{i-1}^l, and c_i, we obtain:

$$t_i^u = (1 - c_i) \cdot (t_{i-1}^u + p_i) + c_i \cdot (t_{i-1}^l + s_i) \tag{1}$$

$$t_i^l = (1 - c_i) \cdot (t_{i-1}^l + q_i) + c_i \cdot (t_{i-1}^u + r_i) \tag{2}$$

Thus, following the lecture slides we have, $\Delta_i = t_i^u - t_i^l$. Subtracting equations (1) and (2) we get:

$$\Delta_i = \Delta_{i-1} \cdot d_i + \alpha_i \cdot d_i + \beta_i$$

where α_i and β_i depend on constants that are indeterminable from the physical/measurable perspective and thus we call them system constants and are given by:

$$\alpha_i = \frac{p_i - q_i + r_i - s_i}{2}$$

$$\beta_i = \frac{p_i - q_i - r_i + s_i}{2}$$

where p_i, q_i, r_i, and s_i are system parameters. Also, d_i is governed by the challenge bits (c_i):

$$d_i = (1 - 2 \cdot c_i)$$

$\Delta_0 = 0$. Moreover, observing the recursion unfold carefully, we can simplify this relation further:

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

$$\Delta_2 = \alpha_1 \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1) \cdot d_2 + \beta_2$$

$$\Delta_3 = \alpha_1 \cdot d_3 \cdot d_2 \cdot d_1 + (\alpha_2 + \beta_1) \cdot d_3 \cdot d_2 + (\alpha_3 + \beta_2) \cdot d_3 + \beta_3$$

The only Δ we require is Δ_{32}, the last output.

$$\Delta_{32} = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_{32} \cdot x_{32} + \beta_{32} = \mathbf{w}^T \cdot \mathbf{x} + b = \mathbf{W}^T \cdot \mathbf{X}$$

where $\mathbf{w}$ and $\mathbf{x}$ are 32-dimensional vectors and $\mathbf{W}$ and $\mathbf{X}$ are 33-dimensional, just including the bias term $W_{33} = \beta_{32}$ and $X_{33} = 1$. Each term of $\mathbf{w}$, $\mathbf{x}$ being:

$$b = \beta_{31}, \quad w_0 = \alpha_0, \quad w_i = \alpha_i + \beta_{i-1}, \quad x_i = \prod_{j=i}^{31} d_j$$

1.2 Deriving the sum of delays

We have,

$$t_i^u = (1 - c_i) \cdot (t_{i-1}^u + p_i) + c_i \cdot (t_{i-1}^l + s_i) \tag{1}$$

$$t_i^l = (1 - c_i) \cdot (t_{i-1}^l + q_i) + c_i \cdot (t_{i-1}^u + r_i) \tag{2}$$

Adding both above equations,

$$t_i^u + t_i^l = t_{i-1}^u + t_{i-1}^+(p_i + q_i) + c_i \cdot (s_i + r_i - p_i - q_i)$$

Put $(s_i + r_i - p_i - q_i) = r_i$ and we get,

$$t_i^u + t_i^l = t_{i-1}^u + t_{i-1}^l + (p_i + q_i) + c_i \cdot r_i$$

Substituting the value of $i = 32, 31, \ldots, 2, 1$ and adding the equations:

$$\begin{aligned}
t_{32}^u + t_{32}^l &= t_{31}^u + t_{31}^l + (p_{32} + q_{32}) + c_{32} \cdot r_{32} \\
t_{31}^u + t_{31}^l &= t_{30}^u + t_{30}^l + (p_{31} + q_{31}) + c_{31} \cdot r_{31} \\
&\vdots \\
t_1^u + t_1^l &= t_0^u + t_0^l + (p_1 + q_1) + c_1 \cdot r_1
\end{aligned}$$

We finally get,

$$t_{32}^u + t_{32}^l = \sum_{i=1}^{32} (c_i \cdot r_i) \tag{1}$$

where

$$r_i = s_i + r_i - p_i - q_i$$

We also derived the difference in previous sections as:

$$t_{32}^u - t_{32}^l = \sum_{i=1}^{32} (w_i \cdot x_i) + \beta_{32} \tag{2}$$

where

$$w_1 = \alpha_1, \quad w_i = \alpha_i + \beta_{i-1}, \quad x_i = \prod_{j=i}^{32} d_j, \quad d_i = (1 - 2 \cdot c_i)$$

$$\alpha_i = \frac{p_i - q_i + r_i - s_i}{2}, \quad \beta_i = \frac{p_i - q_i - r_i + s_i}{2}$$

where p_i, q_i, r_i, s_i are system parameters.

Adding (1) and (2),

$$2t_{32}^u = \sum_{i=1}^{32} (c_i \cdot r_i) + \sum_{i=1}^{32} (w_i \cdot x_i) + \beta_{32}$$

Dividing both sides by 2:

$$t_{32}^u = \frac{1}{2}\left(\sum_{i=1}^{32}(c_i \cdot r_i) + \sum_{i=1}^{32}(w_i \cdot x_i) + \beta_{32}\right)$$

Rewriting x_{32} as $1 - 2 \cdot c_{32}$ and taking c_{32} common from $c_{32} \cdot r_{32}$ and $2 \cdot w_{32} \cdot c_{32}$, we get:

$$t_{32}^u = \frac{1}{2}\left(c_{32} \cdot (r_{32} - 2 \cdot w_{32}) + \sum_{i=1}^{31}(c_i \cdot r_i) + \sum_{i=1}^{31}(w_i \cdot x_i) + \beta_{32} + w_{32}\right)$$

In matrix form, this can be expressed as:

$$t_{32}^u = \frac{1}{2}\left(\mathbf{W}^T \cdot \boldsymbol{\varphi}(c) + b\right)$$

where:

$$\boldsymbol{\varphi}(c) = [c_{32}, c_{31}, \ldots, c_1, x_1, x_2, \ldots, x_{31}]^T$$
$$\mathbf{W} = [r_{32} - 2 \cdot w_{32}, r_{31}, \ldots, r_1, w_1, w_2, \ldots, w_{31}]^T$$
$$b = \beta_{32} + w_{32}$$

2 Solution for Question 2:

The dimensionality of the linear model to predict the arrival time of the upper signal for an arbiter PUF is **63**.

3 Solution for Question 3:

For Response0:

$$t_{0,32}^l = \frac{1}{2}\left(c_{32} \cdot (r_{0,32} + 2 \cdot w_{0,32}) + c_{31} \cdot r_{0,31} + \cdots + c_1 \cdot r_{0,1} - w_{0,1} \cdot x_1 - w_{0,2} \cdot x_2 - \cdots - (\beta_{0,32} + w_{0,32})\right)$$

$$t_{1,32}^l = \frac{1}{2}\left(c_{32} \cdot (r_{1,32} + 2 \cdot w_{1,32}) + c_{31} \cdot r_{1,31} + \cdots + c_1 \cdot r_{1,1} - w_{1,1} \cdot x_1 - w_{1,2} \cdot x_2 - \cdots - (\beta_{1,32} + w_{1,32})\right)$$

Now,

$$t_{0,32}^l - t_{1,32}^l = \frac{1}{2}\left(c_{32} \cdot ((r_{0,32} + 2 \cdot w_{0,32}) - (r_{1,32} + 2 \cdot w_{1,32})) + c_{31} \cdot (r_{0,31} - r_{1,31}) + \cdots + c_1 \cdot (r_{0,1} - r_{1,1})\right.$$
$$\left. -(w_{0,1} - w_{1,1}) \cdot x_1 - (w_{0,2} - w_{1,2}) \cdot x_2 - \cdots - (\beta_{0,32} + w_{0,32} - \beta_{1,32} - w_{1,32})\right)$$

In matrix form, this can be expressed as:

$$t_{0,32}^l - t_{1,32}^l = \frac{1}{2}\left(\mathbf{W}^T \cdot \boldsymbol{\varphi}(c) + b\right)$$

where:

$$\boldsymbol{\varphi}(c) = [c_{32}, c_{31}, \cdots, c_1, x_1, x_2, \cdots, x_{31}]^T$$
$$\mathbf{W} = [(r_{0,32} + 2 \cdot w_{0,32}) - (r_{1,32} + 2 \cdot w_{1,32}), r_{0,31} - r_{1,31}, \cdots, -(w_{0,1} - w_{1,1}), -(w_{0,2} - w_{1,2}), \cdots, -(w_{0,31} - w_{1,31})]^T$$
$$b = -((\beta_{0,32} + w_{0,32}) - (\beta_{1,32} + w_{1,32}))$$

Similarly for Response1:

$$t_{0,32}^u = \frac{1}{2}\left(c_{32} \cdot (r_{0,32} - 2 \cdot w_{0,32}) + c_{31} \cdot r_{0,31} + \cdots + c_1 \cdot r_{0,1} + w_{0,1} \cdot x_1 + w_{0,2} \cdot x_2 + \cdots + (\beta_{0,32} + w_{0,32})\right)$$

$$t_{1,32}^u = \frac{1}{2}\left(c_{32} \cdot (r_{1,32} - 2 \cdot w_{1,32}) + c_{31} \cdot r_{1,31} + \cdots + c_1 \cdot r_{1,1} + w_{1,1} \cdot x_1 + w_{1,2} \cdot x_2 + \cdots + (\beta_{1,32} + w_{1,32})\right)$$

Now,

$$t^u_{0,32} - t^u_{1,32} = \frac{1}{2}\left(c_{32}\cdot((r_{0,32}-2\cdot w_{0,32})-(r_{1,32}-2\cdot w_{1,32}))+c_{31}\cdot(r_{0,31}-r_{1,31})+\cdots+c_1\cdot(r_{0,1}-r_{1,1})\right.$$
$$\left.+(w_{0,1}-w_{1,1})\cdot x_1+(w_{0,2}-w_{1,2})\cdot x_2+\cdots+(\beta_{0,32}+w_{0,32}-\beta_{1,32}-w_{1,32})\right)$$

In matrix form, this can be expressed as:

$$t^u_{0,32} - t^u_{1,32} = \frac{1}{2}\left(\mathbf{W}^T\cdot\boldsymbol{\varphi}(c)+b\right)$$

where:

$$\boldsymbol{\varphi}(c) = [c_{32}, c_{31}, \cdots, c_1, x_1, x_2, \cdots, x_{31}]^T$$
$$\mathbf{W} = [(r_{0,32}-2\cdot w_{0,32})-(r_{1,32}-2\cdot w_{1,32}), r_{0,31}-r_{1,31}, \cdots, r_{0,1}-r_{1,1}, w_{0,1}-w_{1,1}, w_{0,2}-w_{1,2}, \cdots, w_{0,31}-w_{1,31}]^T$$
$$b = (\beta_{0,32}+w_{0,32})-(\beta_{1,32}+w_{1,32})$$

4 Solution for Question 4:

The dimensionality of the linear model to predict the Response0 and Response1 for a COCO-PUF is **63**.

5 Solution for Question 5:

*Zipped **Solution** to Assignment 1*

6 Solution for Question 6: Performance Analysis

(a) Performance analysis of LinearSVC with Different Loss Functions:

Loss Function	Accuracy 0	Accuracy 1	Training Time 0	Training Time 1
hinge	0.9257	0.9933	0.217315	0.297098
squared_hinge	0.9255	0.9961	0.225710	0.392835

- **Training Time:** The model with the Hinge loss function has a shorter training time compared to the Squared Hinge loss function. This indicates that the Hinge loss function is computationally less expensive.
- **Test Accuracy for Class 0:** The test accuracy for class 0 is approximately same for both the Hinge loss function (0.9257) and the Squared Hinge loss function (0.9248). This suggests that both functions provide a same fit for class 0.
- **Test Accuracy for Class 1:** The test accuracy for class 1 is marginally higher for the Squared Hinge loss function (0.9961) compared to the Hinge loss function (0.9933). This indicates that the Squared Hinge loss function also performs slightly better for class 1.

In conclusion, while the Squared Hinge loss function results in slightly higher test accuracy, it comes at the cost of increased training time. Depending on the specific requirements and constraints of the application, one might prefer the Hinge loss function for faster training or the Squared Hinge loss function for slightly better accuracy.

(b) Performance analysis based on the value of C:

- **Analysis for LinearSVC with different C values:**
  - **Training Time:** As C increases, the training time decreases slightly from low C to high C. This suggests that higher regularization (lower C) leads to more computational complexity.

C Value	Accuracy 0	Accuracy 1	Training Time 0	Training Time 1
low	0.9251	0.9912	5.164196	5.179319
medium	0.9264	0.9955	4.654420	4.685272
high	0.9254	0.9940	4.089320	4.028488

  - **Test Accuracy for Class 0:** The test accuracy for class 0 is highest for the medium C value (0.9264), followed by the high C value (0.9254), and lowest for the low C value (0.9251). This indicates that the model's performance improves with increased regularization up to a certain point.
  - **Test Accuracy for Class 1:** Similarly, the test accuracy for class 1 is highest for the medium C value (0.9955), followed by the high C value (0.9940), and lowest for the low C value (0.9912). This trend suggests that a medium level of regularization yields the best overall performance.

- **Analysis for LogisticRegression with different C values:**

C Values	Accuracy 0	Accuracy 1	Training Time 0	Training Time 1
low	0.9244	0.9908	0.658161	0.788198
medium	0.9246	0.9976	0.579305	0.720614
high	0.9248	0.9990	0.548228	0.766156

  - **Training Time:** The training time decreases slightly as C increases, from low C to high C. This indicates that higher values of C (lower regularization) require less training time.
  - **Test Accuracy for Class 0:** The test accuracy for class 0 increases from 0.9244 for low C to 0.9248 for high C. This demonstrates that higher values of C (lower regularization) improve the model's accuracy for class 0.
  - **Test Accuracy for Class 1:** Similarly, the test accuracy for class 1 increases from 0.9908 for low C to 0.9990 for high C. This trend suggests that lower regularization leads to better performance for class 1 as well.

In conclusion, the performance of both LinearSVC and LogisticRegression is influenced by the value of the regularization parameter C. For LinearSVC, a medium C value provides the best balance between training time and accuracy. For LogisticRegression, higher C values (lower regularization) result in better accuracy for both classes, though the training time remains relatively stable.

c. Performance analysis of LinearSVC and LogisticRegression with Different Tolerance values:

Model	Tolerance	Accuracy_0	Accuracy_1	Training_Time_0	Training_Time_1
LinearSVC	Low	0.9248	0.9961	0.781200	0.467721
LinearSVC	Medium	0.9248	0.9961	0.394658	0.400392
LinearSVC	High	0.9248	0.9962	0.227513	0.155159
LogisticRegression	Low	0.9246	0.9949	0.205130	0.210783
LogisticRegression	Medium	0.9246	0.9949	0.145411	0.233395
LogisticRegression	High	0.9246	0.9949	0.146326	0.212163

LinearSVC:

- **Training Time:** The training time decreases with higher tolerance values to lower tolerance values. This suggests that lower tolerance values, which require more precise convergence, result in higher training times.
- **Test Accuracy for Class 0:** The test accuracy for class 0 remains relatively stable across different tolerance values, with no variation at 0.9248, indicating that tolerance has a minimal effect on the performance for class 0.

- **Test Accuracy for Class 1:** The test accuracy for class 1 is also consistent across different tolerance values, ranging from 0.9961 to 0.9962. This stability suggests that the tolerance parameter does not significantly impact the model's ability to correctly classify class 1 instances.

LogisticRegression:

- **Training Time:** The training time varies with different tolerance values, giving diiferent values with different tolerance values. Unlike LinearSVC, LogisticRegression does not show a clear trend in training time relative to tolerance.
- **Test Accuracy for Class 0:** The test accuracy for class 0 remains the same across all tolerance values, at 0.9246. This consistency indicates that tolerance does not affect the performance for class 0 in LogisticRegression.
- **Test Accuracy for Class 1:** Similarly, the test accuracy for class 1 is constant at 0.9949 for all tolerance values, showing that the tolerance parameter does not influence the model's performance for class 1.

In **conclusion**, the tolerance parameter has a minor impact on the accuracy of the **LinearSVC** model, with lower tolerance values resulting in higher training times.
On the other hand, tolerance parameter does not significantly affect the performance of the **LogisticRegression** model, as both the training time and accuracy remain stable across different tolerance values.