CS771 Assignment 1

Surendra Kumar Ahirwar (211083)
Narendra Kumar Ahirwar (220696)
Satmeet Singh Saluja (200890)
Nagmani Nayan (220679)
Garima Verma (231080037)
Sameer Verma (220949)

Problem 1.1 - Part 1: Mathematical Derivation

Derivation of a Linear Model to Predict ML-PUF Responses

Let $\mathbf{c} \in \{0, 1\}^8$ be the 8-bit challenge.

Step 1: Linear Model for Signal Timing in Arbiter PUF. For a simple arbiter PUF, the time it takes for signals to reach the finish line can be modeled linearly.

For an 8-bit challenge $\mathbf{c}$, let:

- $t_0^u(\mathbf{c})$ be the time for the upper signal in PUF0
- $t_0^l(\mathbf{c})$ be the time for the lower signal in PUF0
- $t_1^u(\mathbf{c})$ be the time for the upper signal in PUF1
- $t_1^l(\mathbf{c})$ be the time for the lower signal in PUF1

Each of these times can be expressed as a linear function of a feature map $\phi(\mathbf{c})$:

$$t_0^u(\mathbf{c}) = \mathbf{w}_0^{u\top}\phi(\mathbf{c}) + b_0^u$$
$$t_0^l(\mathbf{c}) = \mathbf{w}_0^{l\top}\phi(\mathbf{c}) + b_0^l$$
$$t_1^u(\mathbf{c}) = \mathbf{w}_1^{u\top}\phi(\mathbf{c}) + b_1^u$$
$$t_1^l(\mathbf{c}) = \mathbf{w}_1^{l\top}\phi(\mathbf{c}) + b_1^l$$

Here, $\phi(\mathbf{c}) \in \mathbb{R}^9$ is the standard feature map for arbiter PUFs (including a bias term).

Step 2: Modeling Response0 and Response1. For the ML-PUF, Response0 is determined by comparing the lower signal from PUF0 with the lower signal from PUF1:

$$\text{Response0} = \frac{1 + \text{sign}(t_0^l(\mathbf{c}) - t_1^l(\mathbf{c}))}{2}$$

Similarly, Response1 compares the upper signals:

$$\text{Response1} = \frac{1 + \text{sign}(t_0^u(\mathbf{c}) - t_1^u(\mathbf{c}))}{2}$$

Defining:

$$\delta_0(\mathbf{c}) = t_0^l(\mathbf{c}) - t_1^l(\mathbf{c}) = (\mathbf{w}_0^l - \mathbf{w}_1^l)^\top\phi(\mathbf{c}) + (b_0^l - b_1^l)$$
$$\delta_1(\mathbf{c}) = t_0^u(\mathbf{c}) - t_1^u(\mathbf{c}) = (\mathbf{w}_0^u - \mathbf{w}_1^u)^\top\phi(\mathbf{c}) + (b_0^u - b_1^u)$$

Preprint. Under review.

We can express:

$$\text{Response0} = \frac{1 + \text{sign}(\delta_0(\mathbf{c}))}{2}$$
$$\text{Response1} = \frac{1 + \text{sign}(\delta_1(\mathbf{c}))}{2}$$

Each $\delta_i(\mathbf{c})$ is a linear function of $\phi(\mathbf{c})$.

Step 3: XOR-PUF Trick for Final Response. The ML-PUF response is the XOR of Response0 and Response1:

$$r(\mathbf{c}) = \text{Response0} \oplus \text{Response1}$$

This can be expressed using sign functions:

$$r(\mathbf{c}) = \frac{1 - \text{sign}(\delta_0(\mathbf{c}) \cdot \delta_1(\mathbf{c}))}{2}$$

Justification:

- If δ_0 and δ_1 have the same sign, their product is positive $\to$ result is 0
- If δ_0 and δ_1 have opposite signs, their product is negative $\to$ result is 1

Step 4: Construction of the Feature Map. Expanding $\delta_0(\mathbf{c}) \cdot \delta_1(\mathbf{c})$:

$$\begin{aligned}
\delta_0(\mathbf{c}) \cdot \delta_1(\mathbf{c}) &= [(\mathbf{w}_0^l - \mathbf{w}_1^l)^\top \phi(\mathbf{c}) + (b_0^l - b_1^l)] \cdot [(\mathbf{w}_0^u - \mathbf{w}_1^u)^\top \phi(\mathbf{c}) + (b_0^u - b_1^u)] \\
&= [(\mathbf{w}_0^l - \mathbf{w}_1^l)^\top \phi(\mathbf{c})] \cdot [(\mathbf{w}_0^u - \mathbf{w}_1^u)^\top \phi(\mathbf{c})] \\
&+ (b_0^l - b_1^l) \cdot [(\mathbf{w}_0^u - \mathbf{w}_1^u)^\top \phi(\mathbf{c})] \\
&+ (b_0^u - b_1^u) \cdot [(\mathbf{w}_0^l - \mathbf{w}_1^l)^\top \phi(\mathbf{c})] \\
&+ (b_0^l - b_1^l) \cdot (b_0^u - b_1^u)
\end{aligned}$$

Let $\mathbf{v}_0 = \mathbf{w}_0^l - \mathbf{w}_1^l$ and $\mathbf{v}_1 = \mathbf{w}_0^u - \mathbf{w}_1^u$, with $c_0 = b_0^l - b_1^l$ and $c_1 = b_0^u - b_1^u$.

The first term is quadratic in $\phi(\mathbf{c})$:

$$\begin{aligned}
(\mathbf{v}_0^\top \phi(\mathbf{c})) \cdot (\mathbf{v}_1^\top \phi(\mathbf{c})) &= \phi(\mathbf{c})^\top \mathbf{v}_0 \mathbf{v}_1^\top \phi(\mathbf{c}) \\
&= \sum_{i=1}^{9} \sum_{j=1}^{9} v_{0,i} v_{1,j} \phi_i(\mathbf{c}) \phi_j(\mathbf{c})
\end{aligned}$$

Explicit Feature Map $\tilde{\phi}(\mathbf{c})$.

$$\tilde{\phi}(\mathbf{c}) = \begin{bmatrix} \{\phi_i(\mathbf{c})\phi_j(\mathbf{c})\}_{1 \le i,j \le 9} \\ \{\phi_i(\mathbf{c})\}_{1 \le i \le 9} \\ 1 \end{bmatrix}$$

That is, $\tilde{\phi}(\mathbf{c})$ is a vector containing:

- All pairwise products of the 9 elements of $\phi(\mathbf{c})$ (including the bias term),
- All 9 elements of $\phi(\mathbf{c})$,
- The constant 1.

Final Model. The ML-PUF response can be predicted by:

$$r(\mathbf{c}) = \frac{1 - \text{sign}(\tilde{\mathbf{W}}^\top \tilde{\phi}(\mathbf{c}))}{2}$$

where $\tilde{\mathbf{W}} \in \mathbb{R}^{91}$ incorporates all the PUF-specific parameters.

Problem 1.1 - Part 2: Dimensionality of the Feature Map $\tilde{D}$

The dimensionality of $\tilde{\phi}(\mathbf{c})$ is:

- Quadratic terms: $9 \times 9 = 81$
- Linear terms: 9
- Constant term: 1

Therefore, $\tilde{D} = 81 + 9 + 1 = 91$

Problem 1.1 - Part 3: Kernel Choice for Perfect Classification

Required Decision Boundary The ML-PUF response is determined by:

$$r(\mathbf{c}) = \frac{1 - \text{sign}(\delta_0(\mathbf{c}) \cdot \delta_1(\mathbf{c}))}{2}$$

where $\delta_0(\mathbf{c})$ and $\delta_1(\mathbf{c})$ are linear functions of $\mathbf{c}$. Expanding their product gives:

$$\delta_0(\mathbf{c}) \cdot \delta_1(\mathbf{c}) = \underbrace{\phi(\mathbf{c})^\top \mathbf{v}_0 \mathbf{v}_1^\top \phi(\mathbf{c})}_{\text{Quadratic}} + \text{Linear Terms} + \text{Constant}$$

This requires learning a **second-degree polynomial decision boundary**.

Optimal Kernel A **Polynomial Kernel of Degree 2** is required:

$$K(\mathbf{c}, \mathbf{c}') = (\gamma \langle \mathbf{c}, \mathbf{c}' \rangle + \text{coef0})^2$$

with parameters:

$$\gamma = 1, \quad \text{degree} = 2, \quad \text{coef0} = 1$$

Justification

1. **Feature Space Alignment**: Expanding the kernel gives:

   $$K(\mathbf{c}, \mathbf{c}') = (\mathbf{c}^\top \mathbf{c}' + 1)^2$$

   This implicitly computes all quadratic terms $\{c_i c_j\}$, linear terms $\{c_i\}$, and a constant - matching the explicit feature map $\tilde{\phi}(\mathbf{c})$ derived earlier.
2. **Perfect Separability**: The kernel SVM's implicit quadratic mapping aligns with the ML-PUF's XOR-of-linear-responses structure and guarantees zero training error (perfect classification) under hard-margin conditions.
3. **Parameter Choices**: $\gamma = 1$ preserves original scale of challenge bits (0/1); coef0 $= 1$ introduces the constant term needed for affine components; degree 2 matches the quadratic nature of $\delta_0 \cdot \delta_1$.

Parameter Notes: Large C enforces hard margins (no training errors); $\gamma = 1$ ensures proper scaling for binary challenges $\{0, 1\}^8$; coef0=1 matches the constant term in $\tilde{\phi}(\mathbf{c})$.

Alternative Kernels

- **RBF**: Universal but needs infinite dimensions to approximate quadratic
- **Matern**: Designed for smoothness, mismatches discrete challenges
- **Linear Kernel**: Insufficient (cannot model XOR)

Thus, the polynomial kernel of degree 2 is both *minimal* and *sufficient*.

$\boxed{\text{Solution: Use polynomial kernel with degree=2, gamma=1, coef0=1}}$

Part 4: Arbiter PUF Inversion: Delay Recovery from Linear Model

1. Problem Setup

Given a 65-dimensional linear model

$$W = \big[w_0,\, w_1,\, \ldots,\, w_{63},\, b\big]^{\mathsf{T}} \in \mathbb{R}^{65},$$

The 256-dimensional nonnegative delay vector to recovered

$$\delta = \big[d_0^{(0)},\, d_0^{(1)},\, d_0^{(2)},\, d_0^{(3)},\, \ldots,\, d_{63}^{(3)}\big]^{\mathsf{T}} \in \mathbb{R}^{256}_{\geq 0}.$$

2. Delay-to-Weight Mapping Model

Assuming each of the 64 stages consists of four physical delays $d_i^{(0)}, d_i^{(1)}, d_i^{(2)}, d_i^{(3)} \geq 0$. The PUF's feature mapping then produces a linear weight vector W via a matrix A which accumulates the contributions of these delays into the 64 weights and one bias.

3. Constructing the System of Equations

Define

$$A \;\in\; \{0,1\}^{65\times 256},$$

where for $j = 0, \ldots, 63$ the row $A_{j,:}$ has ones in exactly those columns corresponding to the four delays of stage $i \geq j$, and zeros otherwise, so that

$$w_j \;=\; \sum_{k=1}^{4} \sum_{i=j}^{63} d_i^{(k)}, \quad j = 0, \ldots, 63.$$

The last row of A encodes the bias contribution (for instance, by placing a 1 in a dedicated "bias-delay" column). Stacking all weights and the bias gives

$$W \;=\; A\,\delta, \quad W \in \mathbb{R}^{65},\; \delta \in \mathbb{R}^{256}_{\geq 0}.$$

4. Optimization Problem to Invert the Model

Since there are infinitely many nonnegative solutions to $A\delta = W$, we choose the one minimizing residual error in the least-squares sense:

$$\delta^\star \;=\; \arg\min_{\delta \in \mathbb{R}^{256}} \;\|A\,\delta - W\|_2^2 \quad \text{subject to} \quad \delta \geq 0.$$

This is a standard *nonnegative least-squares* (NNLS) problem.

5. Summary of Method

1.

Build the 65×256 matrix A encoding how each physical delay contributes to each linear-model weight and the bias.

Solve the NNLS problem $\min_{\delta \geq 0} \|A\delta - W\|_2^2$ using an appropriate solver.

Return the 256-vector $\delta^\star \geq 0$ which reproduces W.

Part 7: Outcomes of Experiment for different Hyperparameters in Linear SVC and Logistic Regression

A. Changing Loss Hyperparameter in Linear SVC

Table 1: Impact of Loss Function on Training Time and Test Accuracy

Hyperparameter	**Training Time (s)**	**Test Accuracy**
Hinge	1.945	1.000
Squared Hinge	1.4307	1.000

B. Changing C Value

(i) Linear SVC

Table 2: Effect of Regularization Strength C in Linear SVC

C **value**	**Training Time (s)**	**Test Accuracy**
1.00×10^{-2}	0.700	0.983
1	4.252	1.000
1.00×10^{3}	1.410	1.000

(ii) Logistic Regression

Table 3: Effect of Regularization Strength C in Logistic Regression

C **value**	**Training Time (s)**	**Test Accuracy**
1.00×10^{-2}	1.617	0.993
1	4.163	1.000
1.00×10^{3}	0.946	1.000

D. Changing Penalty Hyperparameter

(i) Linear SVC

Table 4: Impact of Penalty (ℓ_1 vs. ℓ_2) in Linear SVC

Hyperparameter	**Training Time (s)**	**Test Accuracy**
ℓ_1	10.2994	1.000
ℓ_2	1.2780	1.000

(ii) Logistic Regression

Table 5: Impact of Penalty (ℓ_1 vs. ℓ_2) in Logistic Regression

Hyperparameter	**Training Time (s)**	**Test Accuracy**
ℓ_1	2.884	1.000
ℓ_2	1.568	1.000