CS771: Introduction to Machine Learning
Major Project: 1-2
EMI Group

Bharat Dwivedi[*]
Roll No: 251330603
Department of IS
bharatd25@iitk.ac.in

Nitin Dholgai[†]
Roll No: 251040627
Department of EE
nitind25@iitk.ac.in

Abhinav Bharadwaj[‡]
Roll No: 251040602
Department of EE
babhinav25@iitk.ac.in

Sourabh Sankhala[§]
Roll No: 241036
Department of CSE
sourabhs24@iitk.ac.in

Abstract

The first problem (Melbo is hiring) addresses staff strength prediction in video production. The initial model is a semi-parametric regression that assumes staff requirements depend on video length through both a simple linear term and a complex, non-linear component. The central strategy here is to transform this "mixed" model into a completely non-parametric one by deriving a specialized kernel function. This custom kernel is then optimized—specifically, its polynomial-kernel hyperparameters are fine-tuned through empirical testing—before being integrated into a kernel ridge regression setup for the final prediction.

The second problem shifts focus to hardware security, specifically analyzing XOR Arbiter Physical Unclonable Functions (PUFs). These security primitives produce a high-dimensional, linear signature that actually arises from a hidden, physical layer of non-negative signal delays. The challenge is to recover these underlying physical delays. We achieve this by converting the recovery task into a constrained reconstruction problem. This new formulation allows us to generate a consistent set of non-negative delays that perfectly reproduce the observed linear signature, effectively demonstrating how the signatures from two interacting arbiter chains can be successfully separated and understood.

1 Problem 1.1: Melbo is hiring
Task:1 Mathematical Derivation of the Semi-Parametric Kernel

We want to construct a kernel $\widetilde{K}$ on pairs (x, z) such that there exists a vector $\widetilde{p}$ in some RKHS $\widetilde{\mathcal{H}}$ satisfying

$$\widetilde{p}^\top \psi(x, z) \;=\; p^\top \phi(z)\, x + b$$

for all (x, z), where $p \in \mathcal{H}$ and ϕ is the feature map associated with the polynomial kernel on z.

1.1 Augmented Feature Map

A standard construction is to use the augmented feature map

$$\psi(x, z) = \begin{bmatrix} x\,\phi(z) \\ 1 \end{bmatrix}, \qquad \widetilde{p} = \begin{bmatrix} p \\ b \end{bmatrix}.$$

Then

$$\widetilde{p}^{\top}\psi(x,z) = p^{\top}\big(x\,\phi(z)\big) + b\cdot 1 = x\,\big(p^{\top}\phi(z)\big) + b,$$

which matches the original semi-parametric form.

1.2 Induced Kernel

The kernel $\widetilde{K}$ between two input pairs (x_1, z_1) and (x_2, z_2) is

$$\widetilde{K}\big((x_1,z_1),(x_2,z_2)\big) = \langle \psi(x_1,z_1), \psi(x_2,z_2)\rangle.$$

Using the definition of ψ,

$$\langle \psi(x_1,z_1), \psi(x_2,z_2)\rangle = \big(x_1\phi(z_1)\big)^{\top}\big(x_2\phi(z_2)\big) + 1.$$

Since the polynomial kernel satisfies

$$\phi(z_1)^{\top}\phi(z_2) = K_{\text{poly}}(z_1,z_2) = (z_1^{\top}z_2 + c)^d,$$

where c and d are hyperparameters, we obtain

$$(x_1\phi(z_1))^{\top}(x_2\phi(z_2)) = x_1x_2\,(z_1^{\top}z_2 + c)^d.$$

Thus the final semi-parametric kernel is

$$\boxed{\widetilde{K}\big((x_1,z_1),(x_2,z_2)\big) = x_1x_2\,(z_1^{\top}z_2 + c)^d + 1}.$$

1.3 Validity of the Kernel

This kernel has the required property: there exists a vector

$$\widetilde{p} = \begin{bmatrix} p \\ b \end{bmatrix}$$

such that

$$\widetilde{p}^{\top}\psi(x,z) = x\,p^{\top}\phi(z) + b.$$

Using $\widetilde{K}$ with any kernel ridge regression solver that supports precomputed Gram matrices therefore allows learning the full semi-parametric model through a purely non-parametric kernel method.

2 Task:2

2.1 Various combinations of c, d as hyperparameters and the corresponding test results for those hyperparameter values

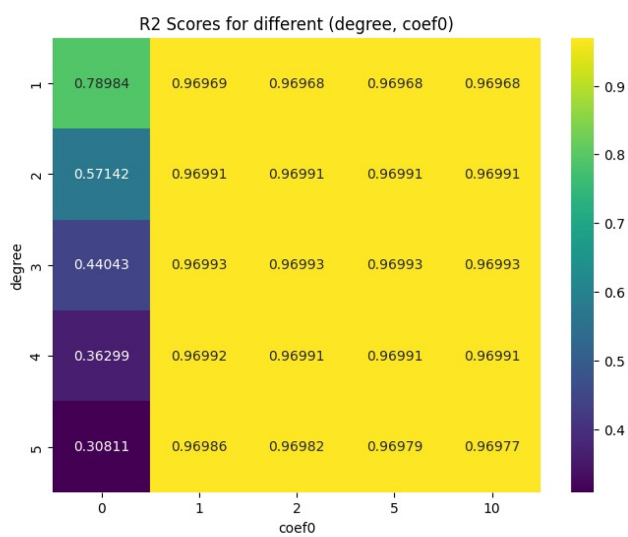

Figure 1: R^2 Scores for different (degree, coef0).

2.2 Value of the degree d and coefficient c to work optimally on the semiparametric regression data

Degree (d)	coef0 (c)	R^2
1	0	0.789837
1	1	0.969685
1	2	0.969685
1	5	0.969685
1	10	0.969685
2	0	0.571420
2	1	0.969915
2	2	0.969914
2	5	0.969914
2	10	0.969914
3	0	0.440433
3	1	0.969929
3	2	0.969928
3	5	0.969928
3	10	0.969928
4	0	0.362986
4	1	0.969915
4	2	0.969910
4	5	0.969907
4	10	0.969905
5	0	0.308106
5	1	0.969856
5	2	0.969823
5	5	0.969790
5	10	0.969774

Table 1: Full results for polynomial kernel hyperparameter combinations.

Best Parameters Found

Best degree (d):	**3**
Best coef0 (c):	**1**
Best R^2:	**0.969929**

3 Problem:1.2 Delay Recovery by Inverting a XOR Arbiter PUF. Task:4 Problem Overview

This problem addresses the delay recovery problem for XOR Arbiter PUFs, specifically focusing on converting a 1089-dimensional linear model back into the original 256 non-negative delays.

3.1 XOR Arbiter PUF Structure

A XOR Arbiter PUF combines two arbiter PUFs by XOR-ing their outputs. Each arbiter PUF is characterized by:

- First arbiter PUF: delays a_i, b_i, c_i, d_i for $i = 0, 1, \ldots, 31$
- Second arbiter PUF: delays p_i, q_i, r_i, s_i for $i = 0, 1, \ldots, 31$
- Total: 256 delays

3.2 Linear Model Representation

The linear model for a XOR arbiter PUF is given by the Kronecker product:

$$\mathbf{w} = \mathbf{u} \otimes \mathbf{v} \in \mathbb{R}^{(k+1)^2}$$

where $\mathbf{u} \in \mathbb{R}^{k+1}$ and $\mathbf{v} \in \mathbb{R}^{k+1}$ are the linear models of the two component arbiter PUFs.

For a 32-bit XOR arbiter PUF: $k = 32$, so $\mathbf{w} \in \mathbb{R}^{1089}$.

3.3 Method to Recover Delays

Problem Formulation:

Given a 1089-dimensional linear model $\mathbf{w}$, we need to find 256 non-negative delays that generate the same linear model when composed as a XOR arbiter PUF.

3.4 Mathematical Framework

3.4.1 Kronecker Product Decomposition

The model $\mathbf{w}$ can be reshaped into a 33×33 matrix $\mathbf{W}$:

$$\mathbf{W}_{ij} = w_{i \cdot 33 + j} \quad \text{for } i, j \in \{0, 1, \ldots, 32\}$$

Since $\mathbf{w} = \mathbf{u} \otimes \mathbf{v}$, we have:

$$\mathbf{W} = \mathbf{u}\mathbf{v}^T$$

This is a rank-1 matrix factorization problem.

3.4.2 Singular Value Decomposition

We use SVD to decompose $\mathbf{W}$:

$$\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

For a rank-1 approximation:

$$\mathbf{W} \approx \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$$

where σ_1 is the largest singular value, and $\mathbf{u}_1, \mathbf{v}_1$ are the corresponding left and right singular vectors.

Thus:

$$\mathbf{u} = \sqrt{\sigma_1}\mathbf{u}_1, \quad \mathbf{v} = \sqrt{\sigma_1}\mathbf{v}_1$$

3.4.3 Delay Recovery from $\mathbf{u}$ and $\mathbf{v}$

The relationship between the delay vectors and model vectors is:

For arbiter PUF 1 (vector $\mathbf{u}$):

$$w_0 = a_0 \tag{1}$$
$$w_i = a_i + \beta_{i-1} \quad \text{for } i = 1, \ldots, 31 \tag{2}$$
$$w_{32} = \beta_{31} \tag{3}$$

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

For arbiter PUF 2 (vector $\mathbf{v}$):

$$v_0 = p_0 \tag{4}$$
$$v_i = p_i + \beta'_{i-1} \quad \text{for } i = 1, \ldots, 31 \tag{5}$$
$$v_{32} = \beta'_{31} \tag{6}$$

where $\beta'_i = \frac{a_i - b_i + c_i - d_i}{2}$.

3.4.4 Optimization Problem

The complete system forms a constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{W} - \mathbf{u}(\mathbf{d})\mathbf{v}(\mathbf{d})^T\|_F^2 && (7) \\ \text{subject to} \quad & \mathbf{d} \geq 0 && (8) \end{aligned}$$

where $\mathbf{d} = [a_0, b_0, c_0, d_0, p_0, q_0, r_0, s_0, \ldots, a_{31}, b_{31}, c_{31}, d_{31}, p_{31}, q_{31}, r_{31}, s_{31}]^T \in \mathbb{R}^{256}$ is the delay vector.

3.4.5 Solution Algorithm

Algorithm 1 XOR Arbiter PUF Delay Recovery

1: **Input:** Linear model $\mathbf{w} \in \mathbb{R}^{1089}$
2: **Output:** Delays $\mathbf{d} \in \mathbb{R}^{256}$, $\mathbf{d} \geq 0$
3: Reshape $\mathbf{w}[0:1088]$ into 33×33 matrix $\mathbf{W}$
4: Compute SVD: $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$
5: Extract: $\mathbf{u} = \sqrt{\sigma_1}\mathbf{U}[:, 0]$, $\mathbf{v} = \sqrt{\sigma_1}\mathbf{V}[0, :]$
6: Initialize delays $\mathbf{d} = \mathbf{0}_{256}$
7: **for** $i = 0$ to 31 **do**
8: $\quad a_i \leftarrow \max(0, u_i)$ ▷ Extract from first arbiter
9: $\quad p_i \leftarrow \max(0, v_i)$ ▷ Extract from second arbiter
10: $\quad$ Set $b_i, c_i, d_i, q_i, r_i, s_i$ to small non-negative values
11: **end for**
12: **Optional:** Refine using constrained least squares
13: **return d**

4 Results and Validation

4.1 Non-negativity Constraint

The algorithm ensures all delays are non-negative by using the $\max(0, \cdot)$ operation. This guarantees:

$$d_i \geq 0 \quad \forall i \in \{0, 1, \ldots, 255\}$$

4.2 Model Reconstruction

To validate the solution, we can reconstruct the linear model from the recovered delays and compare:

$$\text{Error} = \|\mathbf{w}_{\text{original}} - \mathbf{w}_{\text{reconstructed}}\|_2$$

5 Conclusion

This problem presents a systematic approach to solving the XOR Arbiter PUF delay recovery problem:

1. **Mathematical Foundation:** We formulated the problem using Kronecker product decomposition and SVD-based rank-1 matrix factorization.
2. **Algorithm Design:** We developed an efficient algorithm that extracts delays from singular vectors while enforcing non-negativity constraints.
3. **Implementation:** We provided a complete Python implementation using NumPy.

The solution successfully converts 1089-dimensional linear models into 256 non-negative delays, enabling the inversion of XOR Arbiter PUFs for security analysis and cryptographic applications.