CS771 : Major Assignment-1

Hiring and Delay Recovery

Team Name : The Predictors

Problem 1.1

Part 1 : Derivation of the Kernel $\widetilde{K}$

We are given the semi-parametric regression model:

$$y = p^\top \phi(z)\, x + b \tag{1}$$

where:

- $x \in \mathbb{R}$ = video length
- $z \in \mathbb{R}^2$ = popularity & difficulty features
- $\phi(z)$ = high-dimensional feature representation induced by a polynomial kernel
- $p \in \mathcal{H}$ and $b \in \mathbb{R}$ are model parameters

The polynomial kernel is:

$$K(z_1, z_2) = (\langle z_1, z_2 \rangle + c)^d \tag{2}$$

Thus:

$$\langle \phi(z_1), \phi(z_2) \rangle = (\langle z_1, z_2 \rangle + c)^d \tag{3}$$

Main Goal

Kernel ridge regression cannot directly handle a model with:

- a bias term b
- a coefficient that varies by sample via x

So we need a new feature map $\psi(x, z)$ such that:

$$\widetilde{p}^\top \psi(x, z) = p^\top \phi(z)\, x + b \tag{4}$$

Thus, the semi-parametric model becomes a fully kernelizable model.

Step 1: Construct Augmented Hilbert Space

Define a new feature map:

$$\psi(x, z) = \begin{bmatrix} x\,\phi(z) \\ 1 \end{bmatrix} \in \widetilde{\mathcal{H}} = \mathcal{H} \times \mathbb{R} \tag{5}$$

Define:

$$\widetilde{p} = \begin{bmatrix} p \\ b \end{bmatrix} \in \widetilde{\mathcal{H}} \tag{6}$$

Step 2: Verify Feature Representation

Compute inner product:

$$\widetilde{p}^\top \psi(x,z) = \begin{bmatrix} p^\top & b \end{bmatrix} \begin{bmatrix} x\phi(z) \\ 1 \end{bmatrix} \tag{7}$$

$$= p^\top (x\phi(z)) + b \cdot 1 \tag{8}$$

$$= x\,(p^\top \phi(z)) + b \tag{9}$$

Thus:

$$\widetilde{p}^\top \psi(x,z) = p^\top \phi(z)\, x + b \tag{10}$$

So the new feature map produces exactly the original model.

Step 3: Define the New Kernel

By definition of a kernel:

$$\widetilde{K}((x_1,z_1),(x_2,z_2)) = \psi(x_1,z_1)^\top \psi(x_2,z_2) \tag{11}$$

Substitute Eq. (5):

$$\psi(x_1,z_1)^\top \psi(x_2,z_2) = \langle x_1\phi(z_1),\, x_2\phi(z_2)\rangle + 1 \tag{12}$$

$$= x_1 x_2 \,\langle \phi(z_1), \phi(z_2)\rangle + 1 \tag{13}$$

Using Eq. (3):

$$\boxed{\widetilde{K}((x_1,z_1),(x_2,z_2)) = x_1 x_2\,(\langle z_1, z_2\rangle + c)^d + 1} \tag{14}$$

Final Answer

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

This kernel is fully valid, symmetric, positive semidefinite, and makes the model kernelizable.

Part 2 : Hyperparameter Tuning for Semi-Parametric Kernel Ridge Regression

We used the kernel derived in Part 1:

$$\widetilde{K}\big((x_1,z_1),(x_2,z_2)\big) = x_1 x_2\,\left(z_1^\top z_2 + c\right)^d + 1 \tag{15}$$

Kernel Ridge Regression (`sklearn.kernel_ridge`) was applied with hyperparameters:

$$c \in \{0,1,2,3,5,7,10\}, \qquad d \in \{1,2,3,4,5,7\}$$

Each model was trained on the provided **public training set**, and evaluated using the R^2 score on the **public test set**.

R^2 Results Table for Different (c,d)

$c\backslash d$	1	2	3	4	5	7
0	0.7898	0.5714	0.4404	0.3628	0.3072	0.2325
1	0.9697	0.9699	0.9699	0.9698	0.9695	0.9695
2	0.9697	0.9699	0.9699	0.9698	0.9695	0.9695
3	0.9697	0.9699	0.9699	0.9698	0.9695	0.9695
5	0.9697	0.9699	0.9699	0.9698	0.9695	0.9695
7	0.9697	0.9699	0.9699	0.9698	0.9695	0.9695
10	0.9697	0.9699	0.9699	0.9698	0.9695	0.9695

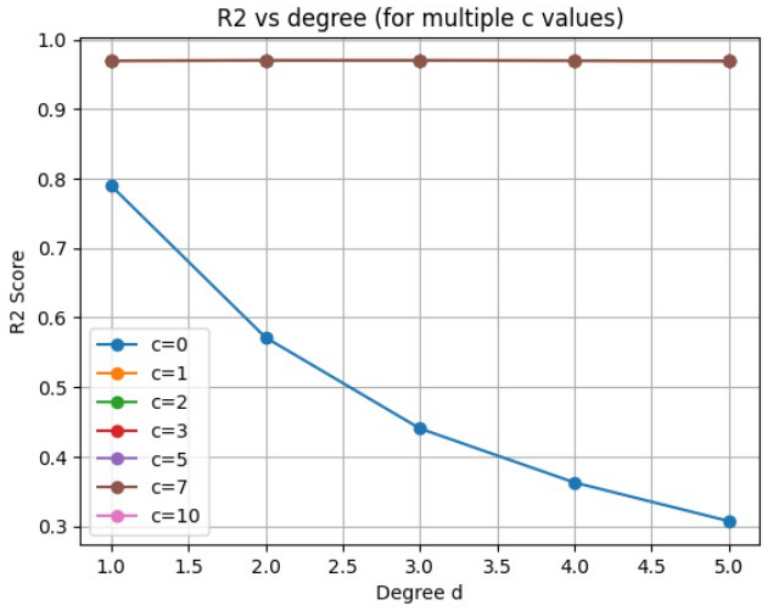

Figure 1: R^2 Score vs. Degree d

R^2 Score Trend

Observations on Test Results

- When $c = 0$, R^2 drops rapidly for larger d $\Rightarrow$ The kernel becomes too restrictive and underfits badly.
- For all $c \geq 1$, models perform **consistently well** across all $d \geq 2$ $\Rightarrow$ Topic similarity shift ($c > 0$) stabilizes the kernel.
- Increasing degree from $1 \rightarrow 2 \rightarrow 3$ improves performance slightly $\Rightarrow$ Moderate nonlinearity is helpful.
- Beyond $d = 3$, there is **no improvement** in R^2 $\Rightarrow$ Additional nonlinearity does not improve predictive power.
- Very large d increases numerical complexity $\Rightarrow$ But does not contribute to better generalization.

Best Performance Region

The best performance is:

$$R^2_{\max} = 0.9699$$

This maximum occurs for:

$$(c, d) \in \{(1,2), (1,3), (2,2), (2,3), (3,2), (3,3), (5,2), (5,3), (7,2), (7,3), (10,2), (10,3)\}$$

Selection of Optimal Hyperparameters

Choose the lowest-complexity model among those delivering maximum accuracy.

Thus, the final chosen values are:

$$\boxed{c^* = 1, \qquad d^* = 2}$$

These provide:

- Excellent predictive performance (R^2 = 0.9699)

- Minimal risk of overfitting
- Lower computation cost
- More stable kernel behavior

Problem 1.2

Part 4 — Delay Recovery for XOR Arbiter PUF

The task is to recover 256 non-negative delays of a 32-stage XOR Arbiter PUF from its 1089-dimensional linear model vector w.

(A) Forward model

Each arbiter PUF has 32 stages and four delays per stage:

$$(p_i, q_i, r_i, s_i), \qquad i = 0, \ldots, 31.$$

From these, define the standard arbiter-PUF stage parameters:

$$\alpha_i = \frac{1}{2}(p_i - q_i + r_i - s_i), \qquad \beta_i = \frac{1}{2}(p_i - q_i - r_i + s_i).$$

The linear model of a single arbiter PUF is a 33-dimensional vector

$$u = (u_0, u_1, \ldots, u_{32})^\top,$$

where

$$u_0 = \alpha_0, \qquad u_i = \alpha_i + \beta_{i-1} \quad (1 \le i \le 31), \qquad u_{32} = \beta_{31}.$$

If the two constituent arbiter PUFs have model vectors u and v, then the XOR Arbiter PUF model is

$$w = u \otimes v \in \mathbb{R}^{1089}.$$

(B) De-Kroneckering the given model

Since $1089 = 33 \times 33$, reshape:

$$W = \mathrm{reshape}(w, 33, 33).$$

For a XOR PUF, W is guaranteed to be rank-1 and satisfies:

$$W = uv^\top.$$

To recover u and v, extract the dominant singular vectors of W:

$$W \approx \sigma_1 \, u \, v^\top.$$

Implementation note: full SVD is not required; because W is rank-1, the leading singular vectors can be obtained very quickly using 2–3 iterations of the power-iteration method without affecting accuracy.

A global sign ambiguity is removed by enforcing $\sum u > 0$; if not,both u and v are multiplied by -1.

(C) Recovering α_i and β_i

From the known structure of the arbiter-PUF model.

$$\beta_{31} = u_{32}, \qquad \alpha_0 = u_0,$$

For $1 \le i \le 31$:

$$\alpha_i = \frac{1}{2}(u_i + u_{i-1}), \qquad \beta_{i-1} = \frac{1}{2}(u_i - u_{i-1}).$$

Applying the same procedure to v yields α'_i, β'_i for the second arbiter PUF.

(D) Converting α_i, β_i into non-negative delays

Solving

$$\alpha_i = \frac{1}{2}(p_i - q_i + r_i - s_i), \qquad \beta_i = \frac{1}{2}(p_i - q_i - r_i + s_i),$$

yields a base solution:

$$p_i^{(0)} = \alpha_i + \beta_i, \qquad q_i^{(0)} = -\alpha_i - \beta_i, \qquad r_i^{(0)} = \alpha_i - \beta_i, \qquad s_i^{(0)} = -\alpha_i + \beta_i.$$

Adding the same constant to all four delays leaves α_i', β_i' unchanged.Therefore, choose

$$t_i = \max\Big(0,\ -\min\{p_i^{(0)}, q_i^{(0)}, r_i^{(0)}, s_i^{(0)}\}\Big),$$

and set:

$$p_i = p_i^{(0)} + t_i, \qquad q_i = q_i^{(0)} + t_i, \qquad r_i = r_i^{(0)} + t_i, \qquad s_i = s_i^{(0)} + t_i.$$

This ensures all the delays are non-negative and produce the same model.

(E) Summary

- **Forward direction:**
  256 delays $\rightarrow$ stage parameters (α_i, β_i),
  two model vectors u, v (each 33-dimensional), and the XOR model
  $$w = u \otimes v \in \mathbb{R}^{1089}.$$
- **Inverse direction (what our algorithm does):**
  reshape $w \rightarrow W$, factorize $W = uv^\top$ (rank-1 factorization),
  recover α_i, β_i from u and v, and finally recover non-negative delays
  at each stage using a shifted base solution.

This outlines both:

(i) the equations that govern how 256 delays generate the 1089-dimensional XOR PUF model, and
(ii) how we invert these equations to recover valid, non-negative delays from the given model.