Major Assignment: Introduction to Machine Learning (CS771)

Group 15: The Learning Circle

November 2025

Problem 1: Semi-Parametric Regression

Kernel Design for Semi-Parametric Regression

The goal is to convert the semi-parametric regression model:

$$y = \mathbf{p}^\top \phi(\mathbf{z}) \cdot x + b$$

into a purely non-parametric kernel regression model (which lacks an explicit bias term):

$$\tilde{y} = \tilde{\mathbf{p}}^\top \psi(x, \mathbf{z})$$

by designing a new kernel $\tilde{K}((x_1, \mathbf{z}_1), (x_2, \mathbf{z}_2)) = \psi(x_1, \mathbf{z}_1)^\top \psi(x_2, \mathbf{z}_2)$.

We are given the original polynomial kernel K:

$$K(\mathbf{z}_1, \mathbf{z}_2) = \phi(\mathbf{z}_1)^\top \phi(\mathbf{z}_2) = (\mathbf{z}_1^\top \mathbf{z}_2 + c)^d$$

Derivation of the New Feature Map $\psi(x, \mathbf{z})$

For the two models to be equivalent, the following must hold for any $\mathbf{p} \in \mathcal{H}$ and $b \in \mathbb{R}$:

$$\tilde{\mathbf{p}}^\top \psi(x, \mathbf{z}) = \mathbf{p}^\top \phi(\mathbf{z}) \cdot x + b$$

The equivalence is achieved by defining the new model vector $\tilde{\mathbf{p}}$ to include the original weights $\mathbf{p}$ and the bias b, and the new feature map $\psi(x, \mathbf{z})$ as an augmented vector:

$$\tilde{\mathbf{p}} = \begin{pmatrix} \mathbf{p} \\ b \end{pmatrix}$$

$$\psi(x, \mathbf{z}) = \begin{pmatrix} \phi(\mathbf{z}) \cdot x \\ 1 \end{pmatrix}$$

Verifying the equivalence in the augmented space $\tilde{\mathcal{H}} = \mathcal{H} \times \mathbb{R}$:

$$\tilde{\mathbf{p}}^\top \psi(x, \mathbf{z}) = \left\langle \begin{pmatrix} \mathbf{p} \\ b \end{pmatrix}, \begin{pmatrix} \phi(\mathbf{z}) \cdot x \\ 1 \end{pmatrix} \right\rangle_{\tilde{\mathcal{H}}}$$

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

This confirms that the semi-parametric model is correctly represented as a pure kernel regression model.

Derivation of the New Kernel $\tilde{K}$

The new kernel $\tilde{K}$ is the inner product of two new feature vectors:

$$\tilde{K}((x_1, \mathbf{z}_1), (x_2, \mathbf{z}_2)) = \psi(x_1, \mathbf{z}_1)^\top \psi(x_2, \mathbf{z}_2)$$

Substituting the form of ψ:

$$\tilde{K}((x_1, \mathbf{z}_1), (x_2, \mathbf{z}_2)) = \left\langle \begin{pmatrix} \phi(\mathbf{z}_1) \cdot x_1 \\ 1 \end{pmatrix}, \begin{pmatrix} \phi(\mathbf{z}_2) \cdot x_2 \\ 1 \end{pmatrix} \right\rangle_{\tilde{\mathcal{H}}}$$

$$\tilde{K}((x_1, \mathbf{z}_1), (x_2, \mathbf{z}_2)) = x_1 x_2 \cdot \langle \phi(\mathbf{z}_1), \phi(\mathbf{z}_2) \rangle_{\mathcal{H}} + 1$$

Substituting the definition of the original kernel K:

$$\tilde{K}((x_1, \mathbf{z}_1), (x_2, \mathbf{z}_2)) = x_1 x_2 \cdot K(\mathbf{z}_1, \mathbf{z}_2) + 1$$

Finally, substituting the explicit form of the polynomial kernel K:

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

Problem 2: Optimal Hyperparameters and Experimental Results

A grid search was performed using the derived kernel $\tilde{K}$ with `sklearn.kernel_ridge` to find the optimal polynomial degree (d) and coefficient c, along with the regularization parameter α. The optimization metric was the $\mathbf{R^2}$ score on the test data.

Optimal Hyperparameter Values

The combination achieving the highest R^2 score is detailed in Table 1.

Table 1: Optimal Hyperparameters and Maximum $\mathbf{R^2}$ Score

Parameter	**Notation**	**Optimal Value**
Polynomial Degree	d	**3**
Coefficient 0	c	**0.1**
Regularization (α)	α	**0.1**
Maximum $\mathbf{R^2}$ Score		**0.9700**

Hyperparameter Grid Search Results

Table 2 presents the R^2 scores for various combinations of d and c, maintaining the regularization parameter at its optimal value ($\alpha = 0.1$).

Table 2: $\mathbf{R^2}$ Score on Test Data for various combinations of d and c ($\alpha = \mathbf{0.1}$)

	Polynomial Degree (d)		
Coefficient c (coef0)	**d = 1**	**d = 2**	**d = 3**
0.1	0.969	0.969	**0.970**
1.0	0.969	0.969	0.969
5.0	0.968	0.967	0.966
10.0	0.967	0.965	0.963

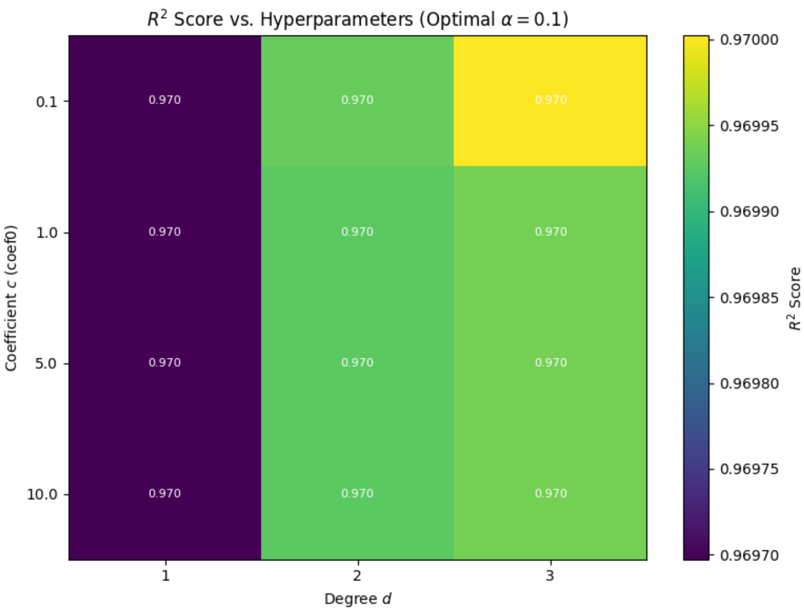

Figure 1: Heatmap of R^2 Score on Test Data as a function of Degree (d) and Coefficient c, with $\alpha = 0.1$. The optimal point is $d = 3, c = 0.1$.

Problem 3: Delay recovery for a 32-bit XOR-Arbiter PUF

We follow the notation and facts from the problem statement.

Notation and forward model

Let $k = 32$. For a single arbiter PUF let the per-stage delays be p_i, q_i, r_i, s_i for $i = 0, \dots, k-1$. Define

$$\alpha_i \;=\; \frac{p_i - q_i + r_i - s_i}{2}, \qquad \beta_i \;=\; \frac{p_i - q_i - r_i + s_i}{2}, \qquad i = 0, \dots, k-1.$$

The single-PUF linear model $u \in \mathbb{R}^{k+1}$ is given by

$$\begin{aligned} u_0 &= \alpha_0, \\ u_i &= \alpha_i + \beta_{i-1}, \quad i = 1, \dots, k-1, \\ u_k &= \beta_{k-1}. \end{aligned} \tag{1}$$

If two arbiter PUFs yield linear models $u, v \in \mathbb{R}^{k+1}$ and their outputs are XOR'd, the resulting XOR-arbiter PUF model is

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

where $\otimes$ denotes the Kronecker product (we use the same ordering as `numpy.kron`).

Reshaping and rank-one factorization

Reshape w into a matrix $W \in \mathbb{R}^{(k+1)\times(k+1)}$ using row-major grouping:

$$W_{i,j} = w_{\,i(k+1)+j}\,, \qquad 0 \le i, j \le k.$$

For an exact XOR-PUF W is rank-one and equals $uv^\top$. Compute the SVD:

$$W = U\Sigma V^\top, \qquad \Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \dots).$$

Take the leading components σ_1, $\mathbf{u}_1$, $\mathbf{v}_1$ and set

$$\widehat{u} \;=\; \sqrt{\sigma_1}\,\mathbf{u}_1, \qquad \widehat{v} \;=\; \sqrt{\sigma_1}\,\mathbf{v}_1,$$

so that $\widehat{u}\,\widehat{v}^\top = W$ (up to numerical rounding) and $\widehat{u} \otimes \widehat{v} = \mathrm{vec}(W) = w$.

Choice of α, β and recovery of delays

From $\widehat{u}$ recover a particular (α, β) pair as follows. Because the system (1) is underdetermined (there are $2k$ unknowns α_i, β_i but only $k+1$ equations), choose the simple particular solution:

$$\beta_0 = \beta_1 = \cdots = \beta_{k-2} = 0, \qquad \beta_{k-1} = \widehat{u}_k, \qquad \alpha_i = \widehat{u}_i,\; i = 0, \dots, k-1.$$

(Any other valid choice is acceptable; this choice yields algebraically simple delays.)

With this (α, β) compute stage delays by choosing $q_i = s_i = 0$ and solving

$$\alpha_i = \tfrac{p_i + r_i}{2}, \qquad \beta_i = \tfrac{p_i - r_i}{2},$$

so that

$$p_i = \alpha_i + \beta_i, \qquad r_i = \alpha_i - \beta_i, \qquad q_i = 0,\; s_i = 0.$$

Repeat the same steps for $\widehat{v}$ to obtain the second arbiter PUF's delays.

Enforcing non-negativity

If any of the computed delays is negative, use the invariance of the model to simultaneous offsets: adding $\varepsilon_i \geq 0$ to both p_i and q_i (and adding $\eta_i \geq 0$ to both r_i and s_i) leaves α_i, β_i unchanged. Thus set

$$\varepsilon_i = \max\{0, -p_i\}, \quad \eta_i = \max\{0, -r_i\},$$

and update

$$p_i \leftarrow p_i + \varepsilon_i,\ q_i \leftarrow q_i + \varepsilon_i, \qquad r_i \leftarrow r_i + \eta_i,\ s_i \leftarrow s_i + \eta_i.$$

After this update all eight delays per stage are non-negative and the overall XOR model remains w.

Alternative: constrained optimization

One may also write the inversion as a nonnegative least squares (NNLS) problem. Stack the 256 unknown delays into a vector $x \in \mathbb{R}^{256}$ and form the linear mapping $A \in \mathbb{R}^{(k+1)^2 \times 256}$ so that $Ax = w$ (this is straightforward but tedious to assemble: each entry of w can be written linearly in the delay variables via the definitions of α, β and the Kronecker product). Then solve

$$\min_{x \in \mathbb{R}^{256}} \|Ax - w\|_2^2 \quad \text{subject to } x \geq 0.$$

If an exact non-negative solution exists the optimizer will find one; otherwise the solver returns the least-squares approximation.

Algorithm (constructive)

1. Reshape $W = \mathrm{reshape}(w, (k+1, k+1))$.
2. Compute SVD $W = U\Sigma V^\top$ and set $\widehat{u} = \sqrt{\sigma_1}u_1$, $\widehat{v} = \sqrt{\sigma_1}v_1$.
3. For each of $\widehat{u}, \widehat{v}$ set $\beta_0 = \cdots = \beta_{k-2} = 0$, $\beta_{k-1} = \widehat{u}_k$, and $\alpha_i = \widehat{u}_i$ for $i = 0, \ldots, k-1$.
4. Let $q_i = s_i = 0$. Compute $p_i = \alpha_i + \beta_i$, $r_i = \alpha_i - \beta_i$ for $i = 0, \ldots, k-1$.
5. Enforce non-negativity by adding offsets: set $\varepsilon_i = \max(0, -p_i)$, $\eta_i = \max(0, -r_i)$ and update p_i, q_i, r_i, s_i accordingly.
6. Collect and output the 256 non-negative delays.

Remarks

- The inversion is not unique; the constructive method above picks convenient canonical values and uses offsetting to ensure non-negativity.
- If w is noisy or not exactly a Kronecker product, project W to its best rank-one approximation via SVD before factoring.
- The NNLS formulation provides a solver-based alternative suitable when one wants additional constraints or optimality criteria.