Lecture Note on Reproducing Kernel Hilbert Spaces (RKHS) and Kernel Regression in One Dimension

Yiping Lu

February 25, 2025

1 Introduction

Reproducing Kernel Hilbert Spaces (RKHS) are Hilbert spaces of functions in which evaluation at any point can be represented as an inner product. This lecture note provides an introduction to the concept of RKHS and demonstrates its application in regression problems with a simple MATLAB example.

2 Reproducing Kernel Hilbert Spaces (RKHS)

2.1 Definition and Basic Properties

Let $\mathcal{H}$ be a Hilbert space of functions defined on a set $\mathcal{X}$. A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *reproducing kernel* for $\mathcal{H}$ if:

1. For every $x \in \mathcal{X}$, the function $K(\cdot, x)$ belongs to $\mathcal{H}$.
2. (Reproducing Property) For every $f \in \mathcal{H}$ and every $x \in \mathcal{X}$, the evaluation of f at x can be written as:
$$f(x) = \langle f, K(\cdot, x) \rangle_{\mathcal{H}}.$$

This property implies that the kernel K "reproduces" the values of the functions in $\mathcal{H}$ through the inner product.

2.2 Mercer's Theorem and Positive Definiteness

A kernel K is said to be *positive definite* if for any finite set of points $\{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}$ and any real numbers $c_1, c_2, \ldots, c_n$, it holds that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0.$$

Mercer's theorem further tells us that for such kernels, there exists an eigen-expansion that connects the kernel with the feature space in which the functions live. This is the theoretical underpinning for many kernel methods in machine learning.

3 Kernel Regression in One Dimension

Kernel regression aims to estimate an unknown function $f : \mathbb{R} \to \mathbb{R}$ from noisy observations $\{(x_i, y_i)\}_{i=1}^n$. One popular method is **kernel ridge regression** where we solve for a function in the RKHS that minimizes a regularized empirical risk.

3.1 Problem Formulation

Given training data $\{(x_i, y_i)\}_{i=1}^n$, kernel ridge regression seeks to find

$$f^* = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \|f\|_{\mathcal{H}}^2,$$

where $\lambda > 0$ is a regularization parameter. By the Representer Theorem, the solution can be written as

$$f^*(x) = \sum_{i=1}^n \alpha_i K(x, x_i).$$

The coefficients $\alpha = [\alpha_1, \ldots, \alpha_n]^\top$ are obtained by solving the linear system:

$$(K + \lambda I)\alpha = y,$$

where $K \in \mathbb{R}^{n \times n}$ is the kernel matrix with entries $K_{ij} = K(x_i, x_j)$.

4 MATLAB Example Code for 1D Kernel Regression

Below is an example MATLAB code that demonstrates kernel ridge regression using a Gaussian kernel. This code is self-contained and does not rely on any external packages.

Listing 1: MATLAB Code for 1D Kernel Ridge Regression

```
% Clear workspace
clear; close all; clc;

% Generate synthetic data
n = 20;                              % Number of data points
x = linspace(-3, 3, n)';              % Input values (column vector)
true_func = @(x) sin(x) + 0.5*x;       % True underlying function
noise = 0.5 * randn(n,1);              % Gaussian noise
y = true_func(x) + noise;            % Noisy observations

% Kernel parameters
sigma = 1.0;                         % Bandwidth for Gaussian kernel
lambdas = [0.0001, 0.1, 10];       % Different regularization parameters

% Construct the Gaussian kernel matrix K
K = zeros(n, n);
for i = 1:n
    for j = 1:n
        K(i,j) = exp(-((x(i) - x(j))^2) / (2 * sigma^2));
```

```
    end
end

% Create a fine grid for predictions
x_test = linspace(min(x)-1, max(x)+1, 200)';
true_vals = true_func(x_test);

% Prepare the figure for plotting
figure; hold on;
colors = {'r-', 'g-', 'b-', 'm-'};

% Loop over different lambda values
for idx = 1:length(lambdas)
    lambda = lambdas(idx);
    % Compute coefficients alpha for kernel ridge regression:
    % (K + lambda*I)*alpha = y
    alpha = (K + lambda * eye(n)) \ y;

    % Define prediction function using the kernel expansion
    f_pred = @(x_new) arrayfun(@(xi) sum(alpha .* exp(-((xi - x).^2) / (2 * sigma^2))), x_new);
    y_pred = f_pred(x_test);

    % Plot prediction for this lambda value
    plot(x_test, y_pred, colors{idx}, 'LineWidth', 2, 'DisplayName', sprintf('\\lambda = %.3f', lambda));
end

% Plot the noisy data and true function
plot(x, y, 'ko', 'MarkerFaceColor', 'k', 'DisplayName', 'Noisy data');
plot(x_test, true_vals, 'k--', 'LineWidth', 1.5, 'DisplayName', 'True function');

xlabel('x');
ylabel('f(x)');
title('Kernel Ridge Regression with Different Regularization \lambda');
legend('show');
grid on;

% Save the figure as a PDF file
print('RKHS.pdf', '-dpdf');
```

5 Discussion and Conclusion

In this note, we have introduced the concept of RKHS and explained the reproducing property that enables us to work with functions through inner products. We also demonstrated how the representer theorem allows us to express the solution of a regularized regression problem as a finite linear combination of kernel functions evaluated at the training points.

The MATLAB example provided illustrates a simple kernel ridge regression using a Gaussian

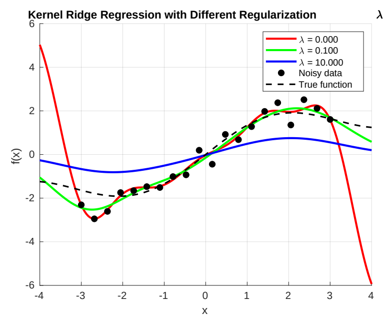

Figure 1: RKHS regression result in one Dimension. The overfit/underfit is controled by the regularization parameters

kernel. The code constructs the kernel matrix, solves for the coefficients, and visualizes the regression result against the noisy data and the true underlying function.

This approach forms the basis for many advanced techniques in machine learning and statistics where kernel methods are used for regression, classification, and more.