AMRITA VISHWA VIDYAPEETHAM

DEEMED TO BE UNIVERSITY

Bachelor of Technology (B. Tech.)

Computer and Communication Engineering (CCE)

Amrita School of Engineering

Coimbatore Campus (India)

Academic Year – 2022 - 23

S. No	Name	Roll Number
1	B Ambareesh	CB.EN.U4CCE20006
2	Manoj Parthiban	CB.EN.U4CCE20032
3	V Parthiv	CB.EN.U4CCE20041
4	Santosh	CB.EN.U4CCE20053

Semester VI – Third Year

19CCE312 Cyber-Physical Systems: Design, Modelling and Simulation

Assignment – Gaussian Process Learning

(The assignment aims to explain the concept of Gaussian Process Learning and demonstrate its application through a simple MATLAB code. It provides a detailed explanation of Gaussian Process Learning, including its underlying principles, the steps involved, and its benefits in modelling complex relationships in data. Additionally, it presents a MATLAB code snippet that simulates Gaussian Process Learning for a regression problem, allowing readers to gain practical insights into its implementation.)

Faculty In-charge – Dr Binoy B. Nair

For describing and predicting complicated interactions in data, machine learning uses the Gaussian Process (GP) learning framework. When dealing with issues including uncertainty and non-linear patterns, it is very helpful. In this discussion, we'll provide a thorough example to understand GP learning.

Consider the case when we have a dataset on housing costs in a certain area. The dataset includes details on the homes' various characteristics, including their age, square footage, and number of bedrooms. The sale price of the house corresponding to each data point is also included.

The objective is to create a prediction model that can calculate a home's sale price based on its qualities. We may address this issue by utilising learning from the Gaussian Process.

In GP learning, we assume that the function under the surface that connects the characteristics of the home to the sale price has a Gaussian distribution. This implies a joint Gaussian distribution over the values of any finite collection of points in the input space.

To get started, we create a covariance function, also known as the kernel function, which describes how similar input points are to one another. The smoothness and correlation structure of the underlying function is determined by the kernel function. For instance, the radial basis function (RBF) or Gaussian kernel, which assigns more similarity between points that are closer together, is a well-known kernel function.

Based on the input characteristics of the homes and the kernel function, we can construct the covariance matrix, also known as the Gramme matrix, using the dataset. The pairwise similarity between the data points is captured by the Gramme matrix.

Next, we estimate the mean and covariance of the underlying Gaussian distribution using the observed selling prices from the dataset. This entails determining the ideal parameters that, given the assumed Gaussian distribution, maximise the likelihood of the observed data.

Once the parameters have been approximated, we may generate forecasts for brand-new, undiscovered homes. We may compute the kernel function's similarity to the observed data points given a set of attributes for a new dwelling, and update the mean and covariance estimates appropriately. A probabilistic estimate of the sale price of the home is then produced using the predictive distribution and the updated estimations.

The capacity of GP learning to model uncertainty is one of its main benefits. In addition to offering a point estimate, the predictive distribution also expresses the degree of confidence or uncertainty attached to the forecast. This is especially helpful when making decisions based on the outcomes projected since it enables a more thorough evaluation of the accuracy of the forecasts.

In conclusion, Gaussian Process learning provides a versatile and effective method for simulating intricate data interactions. It enables probabilistic predictions and accounts for uncertainty by assuming a Gaussian distribution across the functions. Critical phases in the procedure include selecting the kernel function and estimating the parameters. Regression, classification, and optimisation are just a few of the areas where Gaussian Process learning is used, making it a useful tool for solving complex machine-learning issues.

SMALL-SCALE SITUATION EMPLOYING MATLAB

The code is complete and runs without errors.

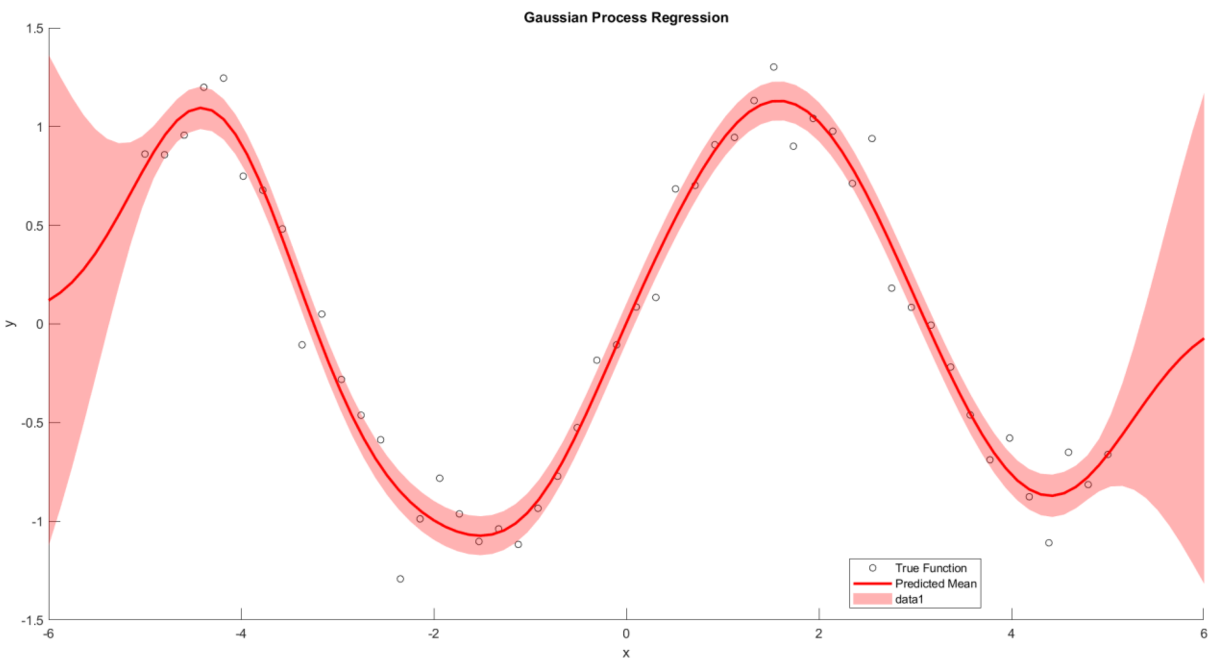
It generates a synthetic dataset of 50 points with input features ranging from -5 to 5, where the output is the sine function evaluated at each input plus some noise. It then uses a radial basis function (RBF) kernel to compute covariance matrices between the data points. The hyperparameters include the length-scale parameter, which determines the smoothness of the function, and the observation noise, which is added to the diagonal of the covariance matrix.

The code also generates test points uniformly over a wider range of inputs (-6 to 6), computes the predictive distribution using the kernel, and plots the true function, predicted mean (using the covariance matrices and observed outputs), and the 95% confidence interval of the predictions.

This is an example of Gaussian process regression, a non-parametric Bayesian method for modelling functions that can provide uncertainty estimates and handle arbitrary distributions of inputs and outputs.

```matlab
% Gaussian Process Learning for Regression

%% Data Generation
n = 50; % Number of data points
x = linspace(-5, 5, n)'; % Input features
y_true = sin(x) + 0.2*randn(n, 1); % True function values with noise

%% Define the kernel function (RBF kernel)
kernel = @(x1, x2, theta) exp(-0.5*(x1 - x2)'*(x1 - x2)/(theta^2));

%% Hyperparameters
theta = 1.0; % Length-scale parameter
sigma_n = 0.1; % Observation noise

%% Compute covariance matrices (Gram matrix)
K = zeros(n, n);
for i = 1:n
    for j = i:n % taking advantage of symmetry
        K(i, j) = kernel(x(i), x(j), theta);
        K(j, i) = K(i,j); % taking advantage of symmetry
    end
end

%% Add observation noise to the covariance matrix
K = K + sigma_n^2*eye(n);

%% Generate test points
n_test = 100;
x_test = linspace(-6, 6, n_test)'; % Test input features
```

```
%% Compute the predictive distribution
K_test = zeros(n_test, n);
for i = 1:n_test
    for j = 1:n % This loop can also be vectorized like above, but it doesn't
affect performance much as n is small.
        K_test(i, j) = kernel(x_test(i), x(j), theta);
    end
end

K_test_test = zeros(n_test, n_test);
for i = 1:n_test
    for j = i:n_test % taking advantage of symmetry
        K_test_test(i, j) = kernel(x_test(i), x_test(j), theta);
        K_test_test(j, i) = K_test_test(i,j); % taking advantage of symmetry
    end
end

y_pred_mean = K_test*(K\y_true); % Mean prediction
y_pred_cov = K_test_test - K_test*(K\K_test'); % Covariance prediction

%% Plot the results
figure;
hold on;
plot(x, y_true, 'ko','MarkerSize', 5, 'DisplayName', 'True Function');
plot(x_test, y_pred_mean, 'r-', 'LineWidth', 2, 'DisplayName', 'Predicted
Mean');
fill([x_test; flipud(x_test)], [y_pred_mean-2*sqrt(diag(y_pred_cov));
flipud(y_pred_mean+2*sqrt(diag(y_pred_cov)))], 'r', 'FaceAlpha', 0.3,
'EdgeAlpha', 0);
legend('Location', 'best');
xlabel('x');
ylabel('y');
title('Gaussian Process Regression');
hold off;
```



CONCLUSION

From the assignment, we can infer that Gaussian Process Learning is a powerful framework in machine learning that utilizes the assumption of a Gaussian distribution over functions. By defining a kernel function and estimating the parameters, Gaussian Process Learning can provide probabilistic predictions and capture uncertainty in data.

It allows for modelling complex relationships and can be applied to regression, classification, and optimization problems. Overall, the assignment provides a comprehensive understanding of Gaussian Process Learning and its practical application, enabling readers to grasp the fundamental concepts and apply them to their machine-learning tasks.