CH5019 MATHEMATICAL FOUNDATION OF DATA SCIENCE
PROJECT

Rishwanth.P.B(ED21B052)

1 Question 1

Given below is the code snippet of the code implementation of **OLS**,**LMS**,**LTS**.

1.1 Ordinary Least Squares Method

```
function [MSE, RB, MAD, avg_theta] = OLS(Y_data, phi, N, R, n_iter, lr, theta_true)
    % Initialize storage for parameter vectors and gradients
    theta_all = zeros(3, R);
    grad_all = zeros(3, R);

    % Perform parameter estimation for each realization
    for r = 1:R
        Y = Y_data(:, r);
        theta = zeros(3, 1);
        grad = zeros(3, 1);

        % Perform Gradient Descent (OLS estimation)
        for iter = 1:n_iter
            % Compute predicted values based on current parameters (theta)
            y_pred = phi * theta;

            % Compute gradient of the loss function (MSE) with respect to theta
            grad = -1 * phi' * (Y - y_pred) / N;

            % Update parameters (theta) using gradient descent
            theta = theta - lr * grad;
        end

        % Store the parameter vector (theta) and gradient for the current realization
        theta_all(:, r) = theta;
        grad_all(:, r) = grad;
    end

    % Compute average parameter vector and gradient across realizations
    avg_theta = mean(theta_all, 2);
    avg_grad = mean(grad_all, 2);

    % Initialize storage for metrics
    MSE = zeros(3, 1);
    RB = zeros(3, 1);
    MAD = zeros(3, 1);

    % Compute metrics for each parameter across all realizations
    for i = 1:3
        % Extract estimated parameter values (theta_i) across all realizations
        theta_est = theta_all(i, :);

        % Compute bias (mean) and variance of the estimated parameter values
        bias_theta = mean(theta_est) - theta_true(i);
        var_theta = var(theta_est);

        % Compute Mean Square Error (MSE) for the parameter
        MSE(i) = bias_theta^2 + var_theta;

        % Compute Robust Bias (RB) for the parameter
        RB(i) = median(theta_est) - theta_true(i);

        % Compute Median Absolute Deviation (MAD) for the parameter
        MAD(i) = median(abs(theta_est - theta_true(i)));
    end
end
```

Figure 1: OLS

1.2 Results for OLS

Result for Y=3X1+5X2+e is given below:

```
Average Estimated Parameters using OLS:

    2.9965
    4.9961
    0.0162

Mean Square Error (MSE) for OLS:

    0.0274
    0.0378
    0.0559

Robust Bias (RB): for OLS

    0.0083
    0.0081
    0.0171

Median Absolute Deviation (MAD) for OLS:

    0.1170
    0.1279
    0.1481
```

Figure 2: OLS RESULT

1.3 Least Median Squares Method

```
function [MSE, RB, MAD, avg_beta] = LMS(Y_data, phi, N, R, n_iter, lr,theta_true)
    beta_all = zeros(3, R);
    grad_all = zeros(3, R);

    for r = 1:R
        Y = Y_data(:, r);
        beta = zeros(3, 1);
        gradient = zeros(3, 1);

        for iter = 1:n_iter
            y_pred = phi * beta;
            residuals = Y - y_pred;

            % Square the residuals
            squared_residuals = residuals.^2;
            huber_delta = median(squared_residuals);
            weights = min(1, huber_delta ./ squared_residuals);

            gradient = -1 * phi' * (weights .* residuals) / N;
            beta = beta - lr * gradient;
        end

        beta_all(:, r) = beta;
        grad_all(:, r) = gradient;
    end

    avg_beta = mean(beta_all, 2);
    avg_grad = mean(grad_all, 2);

    MSE = zeros(3, 1);
    RB = zeros(3, 1);
    MAD = zeros(3, 1);

    for i = 1:3
        beta_est = beta_all(i, :);

        % Compute bias (mean) and variance of the estimated parameter values
        bias_beta = mean(beta_est) - theta_true(i);
        var_beta = var(beta_est);

        % Compute Mean Square Error (MSE) for the parameter
        MSE(i) = bias_beta^2 + var_beta;

        % Compute Robust Bias (RB) for the parameter
        RB(i) = median(beta_est) - theta_true(i);

        % Compute Median Absolute Deviation (MAD) for the parameter
        MAD(i) = median(abs(beta_est - theta_true(i)));
    end
end
```

Figure 3: LMS

1.4 Results for LMS

Result for Y=3X1+5X2+e is given below:

```
Average Estimated Parameters using LMS:
    2.9550
    4.9320
    0.0484
Mean Square Error (MSE) for LMS:
    0.0622
    0.0941
    0.1131
Robust Bias (RB) for LMS:
   -0.0326
   -0.0446
    0.0379
Median Absolute Deviation (MAD) for LMS:
    0.1926
    0.1741
    0.2219
```

Figure 4: LMS RESULT

1.5 Least Trimmed Squares Method

```
function [MSE, RB, MAD, avg_gamma] = LTS(Y_data, phi, N, R, n_iter, lr, q, gamma_true)
    gamma_all = zeros(3, R);
    grad_all = zeros(3, R);

    for r = 1:R
        Y = Y_data(:, r);
        gamma = zeros(3, 1);
        grad = zeros(3, 1);

        for iter = 1:n_iter
            % Compute residuals
            res = Y - phi * gamma;

            abs_res = abs(res);
            sorted_res = sort(abs_res);

            % Select the q smallest residuals
            threshold = sorted_res(q);
            inliers_mask = (res <= threshold);

            if sum(inliers_mask) > 0
                grad = -1 * phi' * (inliers_mask .* res) / N;
            else
                grad = zeros(size(gamma)) / N;
            end

            % Update parameters (gamma) using gradient descent
            gamma = gamma - lr * grad;
        end

        gamma_all(:, r) = gamma;
        grad_all(:, r) = grad;
    end

    avg_gamma = mean(gamma_all, 2);
    avg_grad = mean(grad_all, 2);

    MSE = zeros(3, 1);
    RB = zeros(3, 1);
    MAD = zeros(3, 1);

    for i = 1:3
        gamma_est = gamma_all(i, :);

        % Compute bias (mean) and variance of the estimated parameter values
        bias_gamma = mean(gamma_est) - gamma_true(i);
        var_gamma = var(gamma_est);

        % Compute Mean Square Error (MSE) for the parameter
        MSE(i) = bias_gamma^2 + var_gamma;

        % Compute Robust Bias (RB) for the parameter
        RB(i) = median(gamma_est) - gamma_true(i);

        % Compute Median Absolute Deviation (MAD) for the parameter
        MAD(i) = median(abs(gamma_est - gamma_true(i)));
    end
end
```

Figure 5: LTS

1.6 Results for LTS

Result for Y=3X1+5X2+e is given below:

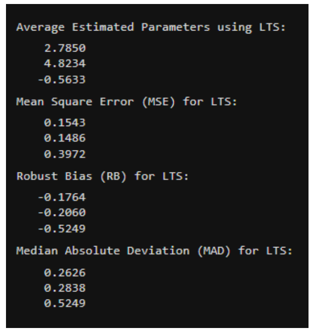
Average Estimated Parameters using LTS:
2.7850
4.8234
-0.5633
Mean Square Error (MSE) for LTS:
0.1543
0.1486
0.3972
Robust Bias (RB) for LTS:
-0.1764
-0.2060
-0.5249
Median Absolute Deviation (MAD) for LTS:
0.2626
0.2838
0.5249

Figure 6: LTS RESULT

1.7 Inference

Based on the estimated parameters and metrics provided above, it can be concluded that **OLS** outperforms **LMS**, which in turn outperforms **LTS**.

2 Results on Real Data

Table 1: Parameter Estimates and MSE for Different Methods

Method	Parameter Estimates	MSE on Test Set
OLS	0.2996 0.1676 1.9764 -0.0222 0.0308 0.0405 -0.5114	0.22689
LMS	0.3025 -0.0011 0.1233 -0.0385 0.0142 0.0341 -0.5762	0.97284
LTS	0.3118 -0.0009 0.0015 -0.0406 0.0137 0.0424 -0.6031	1.0817

2.1 Inference

Similar results have been can observed from the table and can be concluded that **OLS** outperforms **LMS**, which in turn outperforms **LTS** on the basis of **MSE** metric.

3 Question 2

Given below is the code implementation of **Lomb Scargle periodogram**.

```
function [periodogram,ai,bi] = lomb_scargle_periodogram(y, t, frequencies, learning_rate, max_iterations)
    periodogram = zeros(size(frequencies));
    ai = zeros(size(frequencies));
    bi = zeros(size(frequencies));
    nan_indices = isnan(y);
    y(nan_indices) = 0;

    for i =1: length(frequencies)
        for iter = 1:max_iterations
            f= frequencies(i);
            residual = y - (ai(i) * cos(2*pi*f * t) + bi(i) * sin(2*pi*f * t));
            gradient_a = -2 * sum(residual .* cos(2*pi*f * t));
            gradient_b = -2 * sum(residual .* sin(2*pi*f * t));
            ai(i) = ai(i) - learning_rate * gradient_a/1000;
            bi(i) = bi(i) - learning_rate * gradient_b/1000;
        end
        periodogram(i) = periodogram(i) +sum(residual.^2);



    end
    for iter = 1:max_iterations
            omega = 0;
            residual = y - (ai(i) * cos(omega * t) + bi(i) * sin(omega * t));
            gradient_a = -2 * sum(residual .* cos(omega * t));
            gradient_b = -2 * sum(residual .* sin(omega * t));
            ai(i) = ai(i) - learning_rate * gradient_a/1000;
            bi(i) = bi(i) - learning_rate * gradient_b/1000;
    end
    zero_chi_square=sum(residual.^2);
    for i =1:length(frequencies)
        periodogram(i)=(zero_chi_square-periodogram(i))/2;
    end
    y(nan_indices) = NaN;
end
```

Figure 7: Lomb Scargle Periodogram

3.1 Results for (a)

Result for signal containing sinusoidal signals of frequency 10 Hz and 17 Hz.

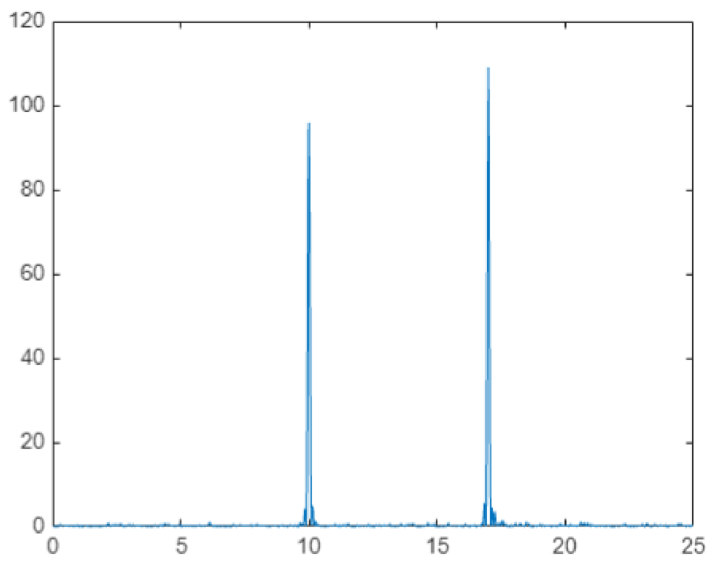

Figure 8: LS Periodogram

3.2 Results of LS Periodogram on Tesla Stock Data

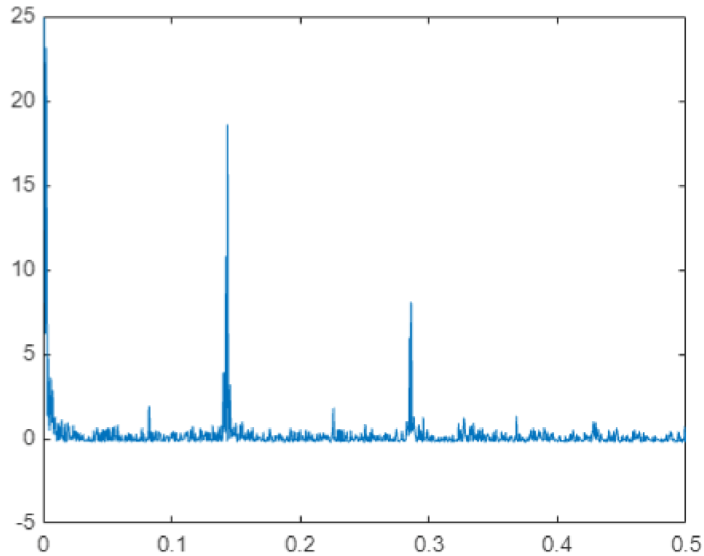

Figure 9: LS Periodogram on Tesla Stock Prices

Reconstructing the signal using the dominant frequencies, we get:

- Normalized Mean Square Error using Lomb-Scargle Periodogram: 1.0009
- Normalized Mean Absolute Percentage using Lomb-Scargle Periodogram: 100.1085%

3.3 Results of ARIMA model on Tesla Stock Data

ARIMA(2,1,1) Model (Gaussian Distribution):

	Value	StandardError	TStatistic	PValue
Constant	4.9232e-05	6.237e-05	0.78935	0.42991
AR{1}	1.0251	0.022035	46.522	0
AR{2}	-0.02514	0.017922	-1.4028	0.16069
MA{1}	-0.96985	0.014511	-66.836	0
Variance	0.0043211	5.4345e-05	79.512	0

Figure 10: ARIMA on Tesla Stock Prices

Reconstructing the signal using the dominant frequencies, we get:

- Normalized Mean Square Error using Lomb-Scargle Periodogram: 524.7689
- Normalized Mean Absolute Percentage using Lomb-Scargle Periodogram: 6378.1207%

3.4 Inference

From the Mean Squared Error and the Mean Absolute Percentage Error we can conclude that **Lomb Scargle Periodogram** method is a better model for estimating the Tesla stock price.

4 Code Repository

The GitHub repo link can be found here.