Amirhossein Shanaghi

Convex Optimization

HW2 Report

June 2024

Instructor : Negin Bagherpour

Problem 1

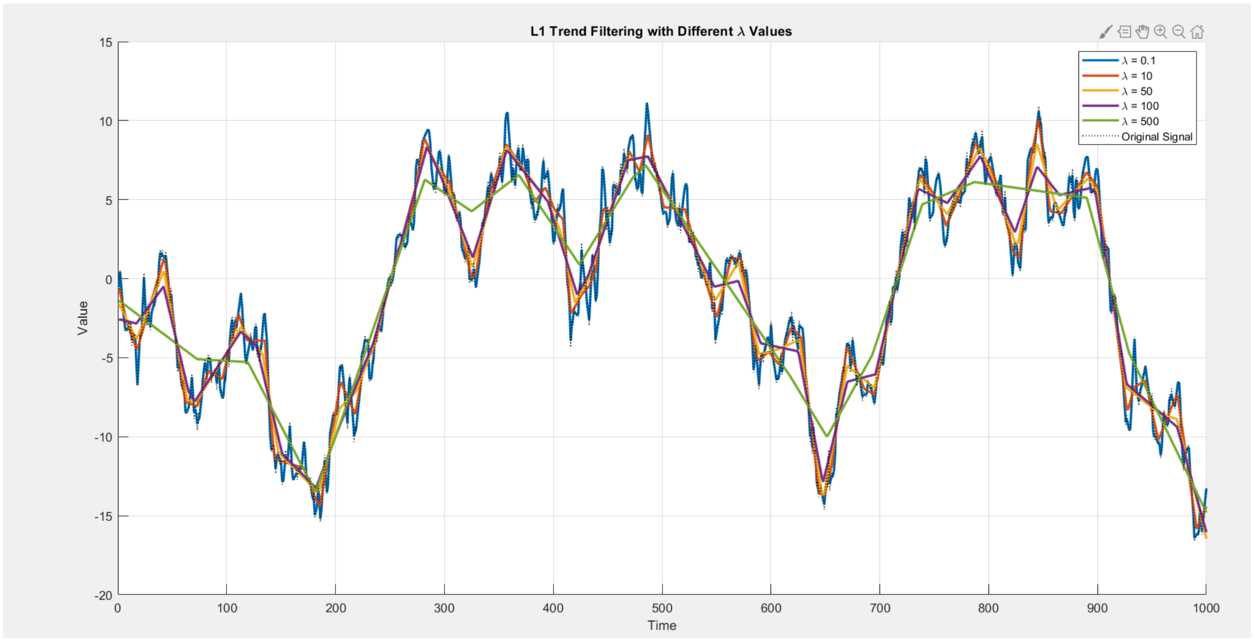
Effect of Lambda (Regularization Parameter) on L1 Trend Filtering

The regularization parameter λ plays a crucial role in the L1 trend filtering method, affecting the balance between the fidelity to the observed data and the smoothness of the estimated trend. The following explains the impact of different values of λ based on the plot provided.

1. **Low λ (e.g., λ=0.1)**:
   - When λ is small, the trend line closely follows the original signal.
   - The regularization term λ‖Dx‖1 has less influence, allowing the estimated trend to capture more of the noise and fluctuations in the data.
   - This can result in an overfitted model where the trend line is less smooth and exhibits more rapid changes.
2. **Moderate λ (e.g., λ=10 or λ=50)**:
   - With moderate values of λ, the trend line strikes a balance between following the data and smoothing out noise.
   - The regularization term starts to play a more significant role, reducing the impact of small fluctuations and providing a smoother trend.
   - This balance helps in capturing the underlying structure of the data while ignoring some of the noise.
3. **High λ (e.g., λ=100 or λ=500)**:
   - For large values of λ, the trend line becomes very smooth and less sensitive to fluctuations in the data.
   - The regularization term λ‖Dx‖1 dominates, leading to a trend that may overlook smaller features and variations in the data.
   - While the trend is smoother, it may not capture important short-term changes, resulting in a model that underfits the data.

Conclusion

- **Low λ**: Leads to a trend line that closely follows the data, potentially capturing noise and overfitting.
- **High λ**: Produces a very smooth trend line, potentially missing important features and underfitting the data.
- **Moderate λ**: Achieves a balance, capturing the underlying trend while smoothing out noise.



Problem 2

We are going to explain the process of fitting a polynomial regression model to data using polynomial regression for **general case with n-point regression with a degree polynomial m that n>m+1** . We use a regularization method to prevent overfitting in regression model. By generating data with random noise and then fitting a regression model, we demonstrate how to achieve a balance between model complexity and predictive accuracy.

Assumptions and Definitions

1. **Number of Data Points (n)**
2. **Degree of Polynomial (m)**
3. **Independent Variable (x)**: Linearly spaced values from 0 to 50
4. **Dependent Variable (y)**: A combination of sine and cosine functions with added random noise
5. **Number of Cross-Validation Folds (num_folds)**: 5
6. **Lambda Values (lambda_values)**: A range of values tested for the regularization parameter in Ridge regression

Explanation of Variables and Methods

- **Independent and Dependent Variables (x and y)**: The independent variable x is a sequence of linearly spaced values. The dependent variable y is generated by applying sine and cosine functions to x and adding random noise. This introduces complexity into the data, making the regression task more challenging.
- **Design Matrix (X)**: The design matrix X is constructed to include polynomial terms of x up to the degree m. This matrix is used in the regression model to capture the polynomial relationships between x and y.
- **Cross-Validation Setup (cv)**: Cross-validation is used to evaluate the performance of the regression model. For each fold, the model is trained on a subset of the data and validated on the remaining part to ensure generalizability.

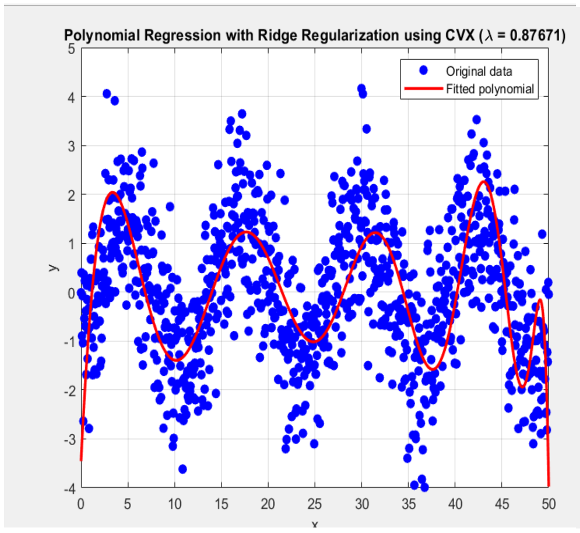
- **Lambda Values (lambda_values)**: These are the values for the regularization parameter tested during the cross-validation process. The goal is to find the lambda value that minimizes the mean squared error (MSE) on the validation sets.
- **Mean Squared Error (MSE)**: This is the metric used to evaluate the performance of the regression model. It measures the average squared difference between the observed actual outcomes and the outcomes predicted by the model.

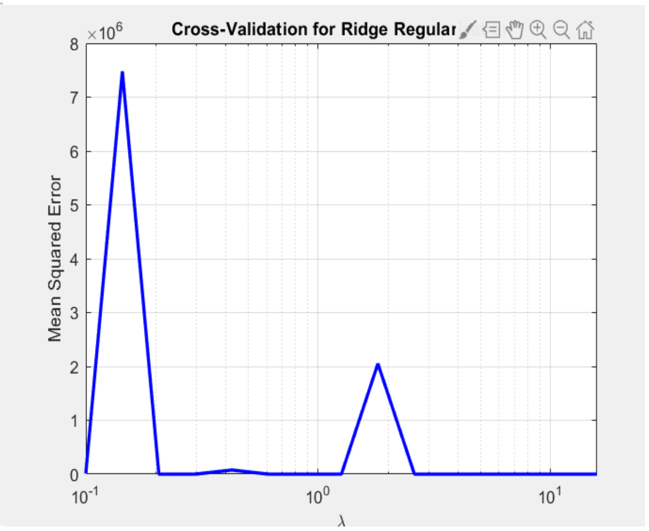
Methods

1. **Generating Complex Data**: We start by generating a dataset with 1000 data points. The independent variable x is linearly spaced from 0 to 50. The dependent variable y is computed as the sum of sine and cosine functions applied to x, plus some random noise to add complexity.
2. **Creating the Design Matrix**: The design matrix X is created to include polynomial terms up to the 15th degree. Each column of X corresponds to a power of x, starting from x^0 (which is a column of ones for the intercept) up to x^15.
3. **Cross-Validation and Finding the Best Lambda**: We perform 5-fold cross-validation to evaluate different values of the regularization parameter lambda. For each lambda value, the dataset is divided into five folds. In each fold, the model is trained on four folds and validated on the remaining fold. The MSE for each lambda value is calculated as the average MSE across all folds.
4. **Solving the Ridge Regression Problem with the Best Lambda**: After identifying the best lambda value that minimizes the MSE, we solve the Ridge regression problem using this optimal lambda. The final model is trained on the entire dataset with the best lambda value to obtain the estimated polynomial coefficients.
5. **Plotting the Results**: We plot the original data points and the fitted polynomial to visualize the model's performance. Additionally, we plot the MSE values against the lambda values on a logarithmic scale to show the results of the cross-validation process.

Conclusion

In this problem, we successfully used Ridge regression and cross-validation to fit a polynomial model to complex data. This approach helps prevent overfitting and ensures a more accurate and reliable model. The results demonstrate that testing different lambda values and selecting the optimal one through cross-validation significantly improves the model's performance. The balance between model complexity and predictive accuracy is achieved by appropriately regularizing the model.





Problem3

Problem Definition

Given a matrix **A** of size 100×200 and a vector **b** of size 100x1, we want to find a sparse vector X of size 200x1 that satisfies **AX=b**. The sparsity constraint means that the solution **X** should have as many zero elements as possible.

Methods

1. Data Generation:

- **Matrix A**: A random matrix of size 100×200 with elements drawn from a normal distribution.
- **Sparse Solution x0**: A sparse vector with 10% non-zero elements.
- **Vector b**: Constructed as AX0=b ensuring the system has a sparse solution.

2. ℓ1-norm Heuristic:

- This heuristic aims to find a solution by minimizing the ℓ1-norm of x. The ℓ1-norm is the sum of the absolute values of the elements of x.

- The optimization problem is formulated as:

$$\text{minimize} \quad \|x\|_1 \quad \text{subject to} \quad Ax = b$$

  This approach is effective in promoting sparsity because the ℓ1-norm encourages smaller coefficients to be driven to zero.

3. Log-based Heuristic:

- This is an iterative method that aims to find a sparse solution by minimizing a weighted ℓ1-norm, where the weights are adjusted based on the current solution.

- The optimization problem in each iteration is formulated as:

$$\text{minimize} \quad \sum_i W_i |x_i| \quad \text{subject to} \quad Ax = b$$

  The weights are updated according to the rule:

$$W_i = \frac{1}{\delta + |x_i|}$$

- The algorithm iterates, refining the weights and the solution until convergence.

Implementation

1. **Setting Parameters**:
    - Define the number of equations m and unknowns n.
    - Set a small threshold value δ to determine which elements are considered zero.
2. **Generating Data**:
    - Create the matrix A and the sparse vector x0.
    - Calculate the vector b as b=Ax0.
3. **Finding the Sparse Solution Using ℓ1-norm Heuristic**:
    - Use CVX to solve the optimization problem.
    - Count the number of non-zero elements in the solution.
4. **Iterative Log-based Heuristic**:
    - Initialize weights and iteratively solve the weighted ℓ1-norm optimization problem using CVX.
    - Update the weights based on the current solution and repeat the process.
    - Track the number of non-zero elements in each iteration.

Results

- **ℓ1-norm Heuristic**:
    - Found a solution with a specified number of non-zero elements.
- **Log-based Heuristic**:
    - Iteratively refined the solution, reducing the number of non-zero elements over several iterations.
    - The plot shows the number of non-zero elements (cardinality) versus the iteration number.
    -

Conclusion

The ℓ1-norm heuristic and log-based heuristic effectively find sparse solutions for the linear system AX= b. The ℓ1-norm heuristic provides a straightforward approach to promoting sparsity, while the log-based heuristic offers an iterative refinement that can further reduce the number of non-zero elements.