**ROB313 Assignment 2**

1. **Assignment Objectives**

The objectives of Assignment 2 are as follows:

1. Derive an expression for the weights of a generalized linear model (GLM) with least-squares loss and Tikhonov regularization.
2. Given a GLM, derive how **α** can be estimated, comparing to a dual representation example from class.
3. Given a grid of shape and regularization parameters, implement a radial basis function (RBF) model that minimizes least-squares loss using a Gaussian kernel.
4. With a designed dictionary of basis functions, implement a greedy regression algorithm using orthogonal matching pursuit metric.
5. **Question 1 – Tikhonov Regularization**
   1. **Expression for the weights of the given generalized linear model (Q1)**

Text, letter

Description automatically generated

1. **Question 2 – Generalized Linear Model** 
   1. **Estimation of α (Q2)**

Text, letter

Description automatically generated

* 1. **Comparison to dual representation in class (Q2)**

The dual representation found in class obtains the result **α** = (**K** + λ**I**)-1**y**. This is different from the result for α obtained above because…

1. **Question 3 – Radial Basis Function (RBF) Model**
   1. **Code Structure Overview & Implementation Strategies (Q3)**

Outside of any functions, I entered the given question values for the different parameters to test. The grid of shape parameters consisted of θ = {0.05, 0.1, 0.5, 1, 2}, and the grid of regularization parameters consisted of λ = {0.001, 0.01, 0.1, 1}.

4.1.1 Gaussian kernel function

The first function implemented was drawn from the class-provided notes for a multi-dimensional gaussian kernel. Given points in two arrays with a lengthscale parameter defaulted to θ = 1, the function returns the Gram matrix. The kernel is evaluated using the squared Euclidian distances between the points in the arrays.

4.1.2 Radial basis function model

An RBF function was implemented to minimize the root mean square error (RMSE) loss. The optimal parameters θ and λ were found by iterating through every possible combination of θ and λ from the provided values stated earlier and choosing that combination where the RMSE was minimum.

In this function, I added in parameters into the function to improve the usability of this model, which minimized the number of lines needed to comment/uncomment to run different parts. The first parameter is the dataset that is desired to run, which is either mauna\_lao or rosenbrock, and either of the dataset is automatically loaded depending on the input. Furthermore, I added a parameter, which was a boolean operator; the default value, validation=True, caused the RBF model to operate on the validation set, which found the minimum RMSE loss, which determined the optimal θ and λ parameter values. If the validation parameter is set to False, the RBF model operates on the testing data, allowing one to obtain the RMSE loss for every parameter combination as well. The testing loss was found by finding the loss corresponding to the optimal parameter values found earlier with the validation set. The other two parameters to this function are the list of possible shape and regularization parameters, which were already defined earlier.

For the radial basis function model, a kernel matrix was constructed using the gaussian kernel helper function explained in section 4.1.1. Regularization was also implemented to prevent overfitting of the data using the built in numpy “np.eye” function. Then, the Cholesky factorization was completed using the also built in cho\_factor function, which enabled calculating an estimate for the alpha values. Finally, the error was calculated for each prediction, and the optimal θ and λ corresponded to those that resulted in the smallest error.

* 1. **Results (Q3)**

The following table summarizes the RMSE losses on the validation and testing sets of the mauna\_loa and rosenbrock datasets. The validation RMSE was found by running the RBF model with every possible combination of shape parameters (θ’s) and regularization parameters (λ’s); the reported validation RMSE’s were the lowest RMSE found. As well, the optimal θ and optimal λ were those parameters that resulted in the lowest RMSE. The full results for every θ and λ combination are provided in the Appendix for the validation sets as an example (for Q3).

Using the optimal parameters found (resulting in the smallest RMSE loss), the RBF model was used on the testing data. For each dataset, the resulting test RMSE was provided using their optimal θ and λ.

*Table 1: RMSE with selected hyperparameters*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Optimal θ** | **Optimal λ** | **Validation RMSE** | **Test RMSE** |
| **Mauna\_loa** | θ = 1 | λ = 0.001 | 0.124479 | 0.154843 |
| **Rosenbrock** | θ = 2 | λ = 0.001 | 0.193240 | 0.084572 |

1. **Question 4 – Greedy Regression Algorithm**
   1. **Code Structure Overview & Implementation Strategies (Q4)**

5.1.1 Basis helper function structure

A greedy regression algorithm was implemented using over 400 basis functions. First, in order to design a dictionary of candidate basis functions, I observed the mauna\_loa dataset structure by plotting it. The data seemed to be a combination of sinusoidal and polynomial elements, so I tested out a few different implementations on a graphing calculator and then on top of the mauna\_loa data. The different parameters were adjusted and played with to see which ranges may fit the dataset the best; this helped in determining a ball park range of parameters to loop over. For example, the graph below shows a random function found from trying out different parameters on a 0.1\*sin(80\*x) + x function.

*Graph 1: Playing with different parameters to understand how to design the basis functions.*

Chart

Description automatically generated

As a result, I created three helper functions that would construct the basis functions: sinusoidal, exponential, and polynomial. The parameters were different inputs, such as phase shifts and amplitudes to adjust the function outputs.

5.1.2 Basis dictionary maker

Another helper function created was called basis\_maker(), which returns a list of possible basis functions. For the sinusoidal function, there were two for loops to iterate over around 20 angular frequency values and phase shifts; in total, this created at least 400 basis functions. In addition, there were for loops to iterate over the different parameters (degree) for the exponential function and polynomial function. All of these basis functions were appended to the returned list.

5.1.3 Greedy regression algorithm

Finally, the greedy regression algorithm was implemented in a greedy() function which took in defaulted inputs, which were a list of basis functions and the dataset to analyze (mauna\_loa). Similar to the previous question, the data was automatically loaded within the function to improve usability of the code. The test and validation sets of data were combined since it was required to do so to predict on the test set.

Following the general pseudocode provided from class, I implemented a while loop with two stop conditions; the first was to ensure that the norm of the residual vector (comparing to an epsilon value of 0.1) and the provided minimum description length (MDL) did not grow by continuously updating it and comparing it to its previous value.

NOT DONE

* 1. **Results (Q4)**

Chart, histogram

Description automatically generatedChart, scatter chart

Description automatically generated

Test RMSE: 0.05819999

1. **Appendix**

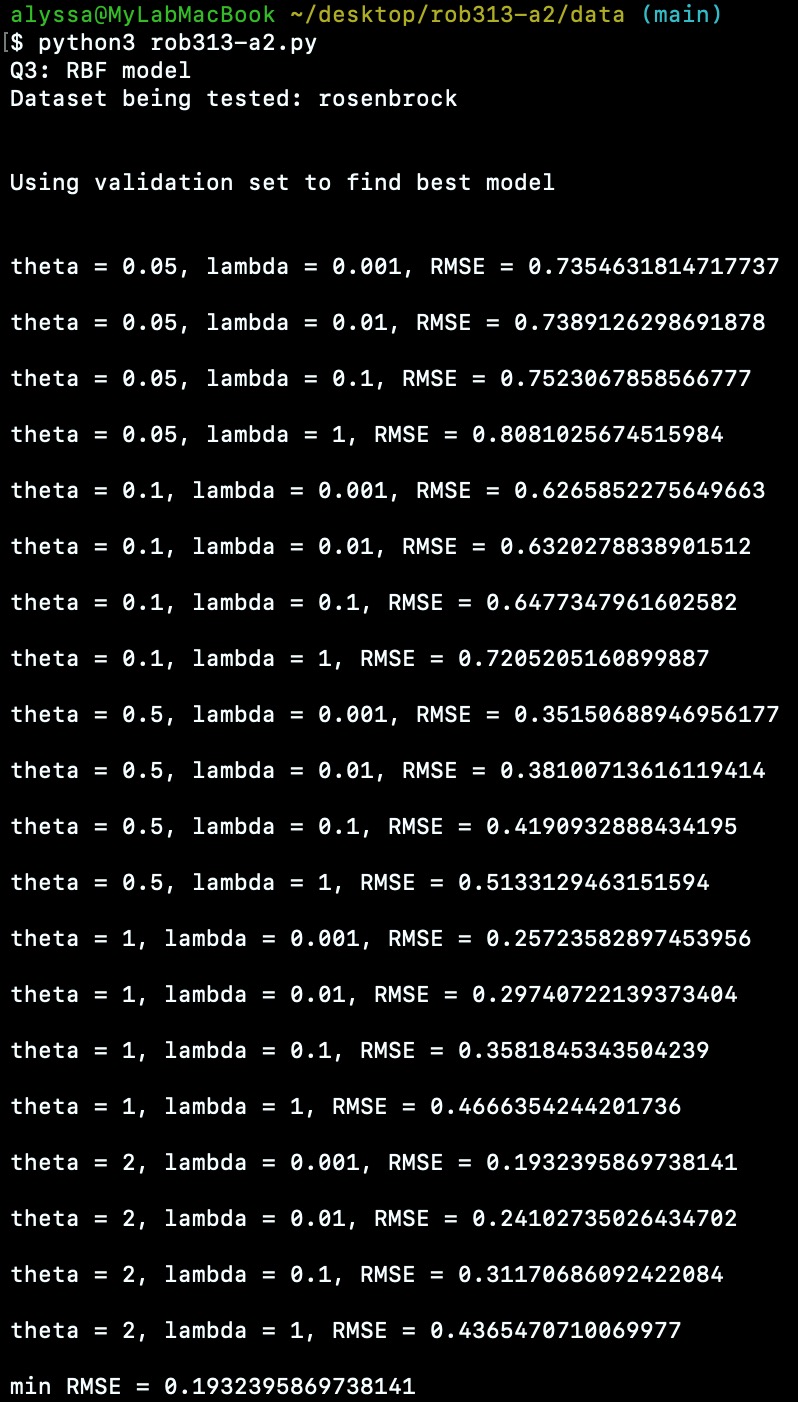
**Q3: All validation RMSE from every θ and λ parameter combination for mauna\_loa**

Text

Description automatically generatedText

Description automatically generated

**Q3: All validation RMSE from every θ and λ parameter combination for rosenbrock**

Text

Description automatically generated