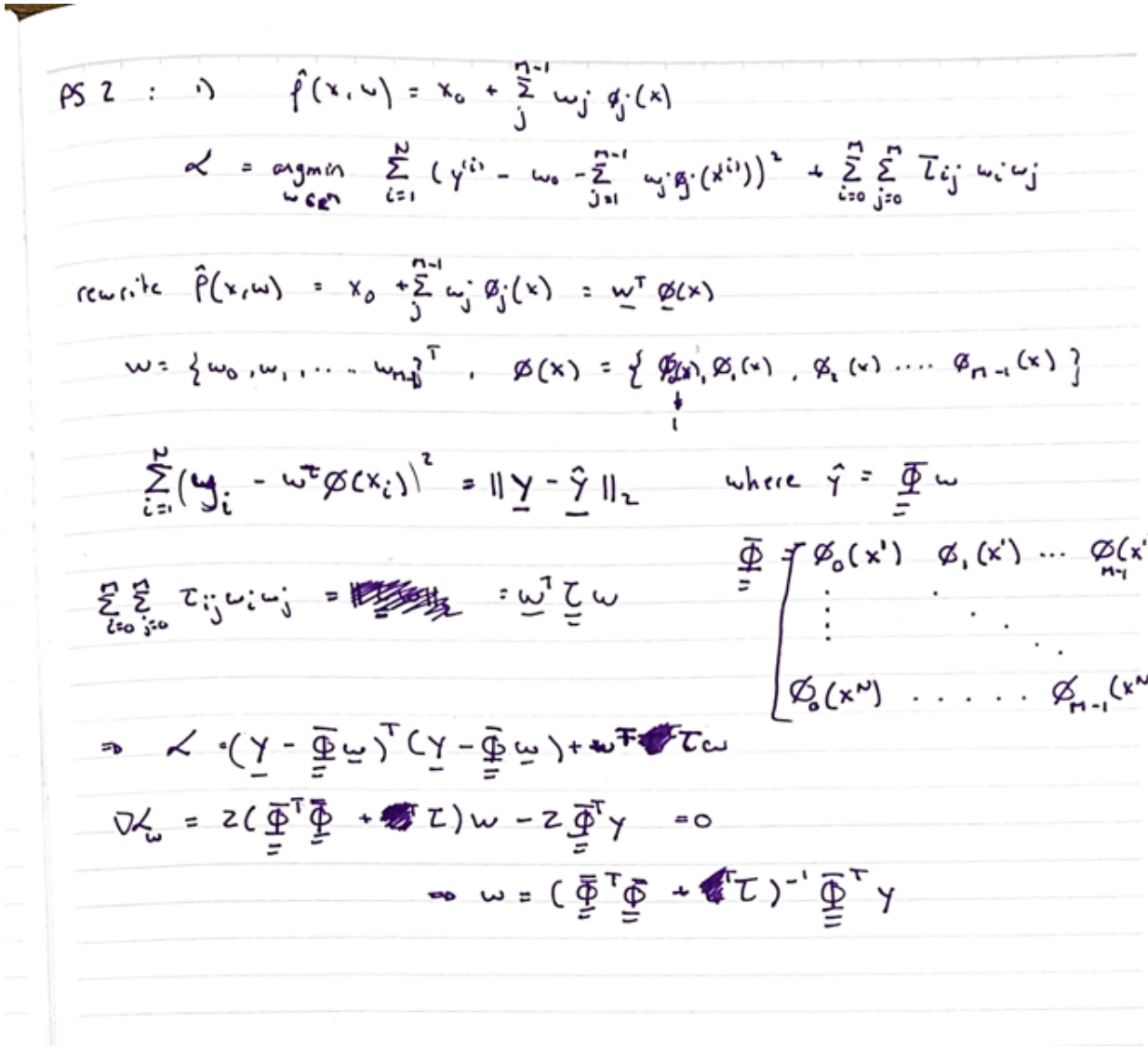
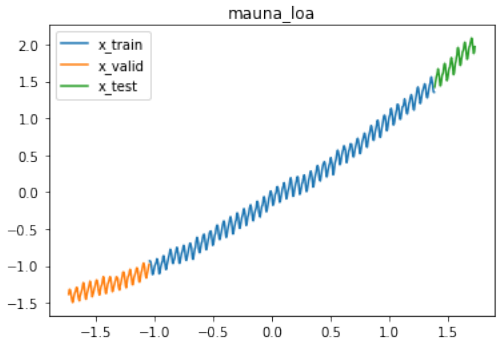
ROB 313: Assignment 2

Ali Seifeldin 1003894431



In this assignment, we attempt to speed up computation processes by utilizing different approaches (primal and dual representation models), as well as applying different kernels. We also use a greedy algorithm in order to select which basis functions to add to the model, that would give us the most value.

2)

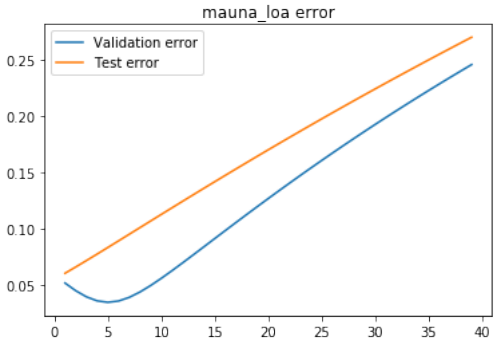


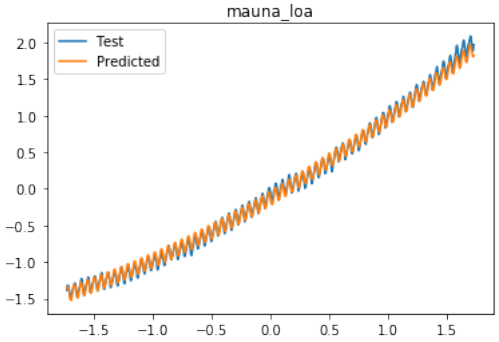
We start by plotting the mauna_loa set in order to visualize the occurring trends. As we can see from the plot, we could potentially fit an mx+b linear function on it in order to get a simple prediction. We also realize the oscillations that occur, thus adding a sin and/or a cosine basis function will help model those. In addition, we can partially see that the model doesn't increase linearly, therefore we predict an x^2 or e^x factor will be present.

basisFunction(X) takes a vector X of N training points with D dimensions (1 dimension in this case) and returns an N by M matrix where M is the number of basis functions used. Being able to generalize this function allowed me to test out multiple possible basis functions easily.

We implemented a simple l2 – loss function using SVD in order to solve for the weights with different λ values, which is minimized when it's equal to 4. We are able to get an error of 0.0342 by using the following basis functions {1, x, x^2, cos(110.19x), sin(110.19x)}. The value of 110.19 was calculated by the frequency of oscillations in the model

Finally, we can see how the predicted model closely resembles the test model for the calculated weights.

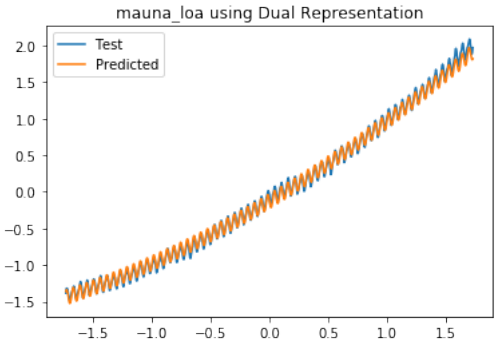




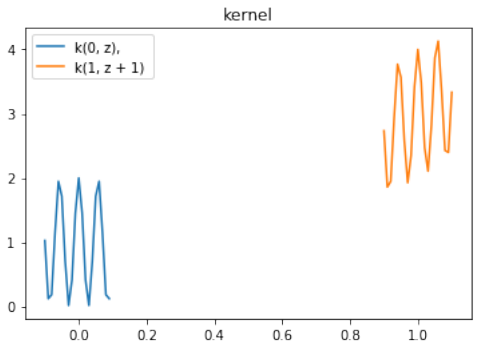
3) Comment on the computational cost and memory requirements of the dual approach versus the primal approach.

Construction an equivalent basis function in a dual representation form, we get the following model

kernel(x,z) = {1 + np.dot(x,z) + np.dot(x,z)**2 + np.dot(np.cos(110.19*x),np.cos(110.19*z)) + np.dot(np.sin(110.19*x),np.sin(110.19*z))}

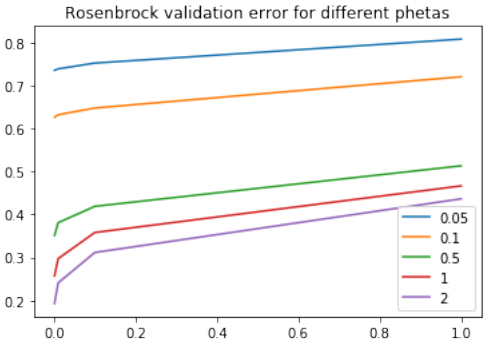


We then compute Gram + kI and apply the Cholesky factorization on it. We then proceed to solve for alpha and compute the error for the validation set. Plotting the validation curve lets us pick a regularization parameter of 4. We therefore get the following predicted curve. We notice that the RMSE is 0.0342, which is the same we got on question 2.
The computational cost for dual representation is O(N^3) with a space of O(N^2). For a primal approach is O(ND) for space and O(2N(D+1)^2 + 11N(D+1)^3) for computational cost. In this example, it would be quicker to use the primal approach since D<<N



Plotting the kernel below, we can see that that it is not translation invariant. The shape of the plotted kernel slightly varies when 1 is added to the each argument of the kernel.

4)



In order to find the values in the table, I plotted the different validation error for the combinations of lambda and thetas given. We emply Cholesky factorization using the RBF Gausssian kernels to optimize the hyperparameters.
In order to do so, we construct the Gram matrix and add the lambda hyperparameter to it. We are then able to decompose the Matrix and solve for the weights efficiently.

Dataset	theta	lambda	Validation	Test
Mauna_Lao	1	0.001	0.12447867031377788	0.14977338771821658
Rosenbrock	2	0.001	0.1932395869740603	0.14812442755014374
IRIS	2	1	0.8709677419354839	1

5)

At every iteration, we select a basis function based on the point with the largest error. We use that point as the gaussian basis average and recompute the new weights with the added basis. We are able to stop the computing new weights when the value of the MDL function starts increasing again. We are thus able to use a sparse model with the following error.

Dataset	theta	Sparsity	Training Error	Test Error
Rosenbrock	0.01	21	376.20	953.30
Rosenbrock	0.1	18	361.41	666.01
Rosenbrock	1	8	369.56	443.36