Samuel Atkins
February 28, 2019
1002951754

ROB313 Assignment 2

Assignment Objectives:

Learning Objectives:

1. Understand how to select an effective set of basis functions and regularization parameter.
2. Understand how the computational cost and memory savings when using the dual approach.
3. Understand the difference between a stationary and non-stationary kernel.
4. Understand various weight derivations.

Functional Objectives:

1. Using SVD, construct a GLM and select a set of basis functions and regularization parameter.
2. Construct a kernelized GLM using Cholesky factorization.
3. Construct an RBF model using a Gaussian kernel. Consider a variety of θ and regularization values.
4. Derive the minimizer of the least-squares loss of a given linear regression model.
5. Derive a strategy to estimate $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}^T$ given an objective function.

Code Structure Explanation:

For this project I have implemented three different files for the three different coding implementations required.

Question 1: svd_glm.py

The first file, svd_glm.py, implements a GLM using SVD. In this file, the optimal regularization parameter is found by finding the lambda that minimizes the least squares error. Finding the optimal regularization parameter is contained in a function called minLambda(). After this parameter is selected, the ϕ matrix is constructed using the training data. This matrix is then used to calculate the weights. After calculating the weights, the test predictions are determined. The error of these predictions is then computed, and the predictions are plotted using the performancePlot(. . .) function.

Question 2: kernelized_cholesky_glm.py

The second file, kernelized_cholesky_GLM.py, is much shorter than the first file. All code executes in the kernelized_cholesky_glm() function. This function computes the K matrix and the α vector using the training and validation data combined. After finding the α vector, the testing predictions are recovered and the RMSE is computed. The kernel is then plotted.

Question 3: gaussian_glm.py

The final file, gaussian_glm.py, implements a gaussian GLM for regression and classification. There are two main functions in this file, gaussian_glm_regression(regressInd) and gaussian_glm_classification(). The regression function computes the optimal parameters for regression by calling findParameters(regressInd) where the regressInd represents which dataset is being used. After finding the optimal parameters, the test predictions are easily calculated using the validation and training dataset for training. These predictions are then plotted. Similarly, the classification function finds the optimal parameters by calling findParameters(3) where 3 represents the iris classification dataset. After finding the optimal parameters, the predictions are recovered and plotted.

Question 1: SVD GLM Implementation

A 5th order polynomial and a sin feature with a frequency of 110.125 were implemented as the basis functions for the GLM. These features proved to be very successful at modeling the dataset. The RMSE with these features is 0.0423. Figure 1 below is a graph of the predictions, y_test values, and y_train values as a function of the inputs:

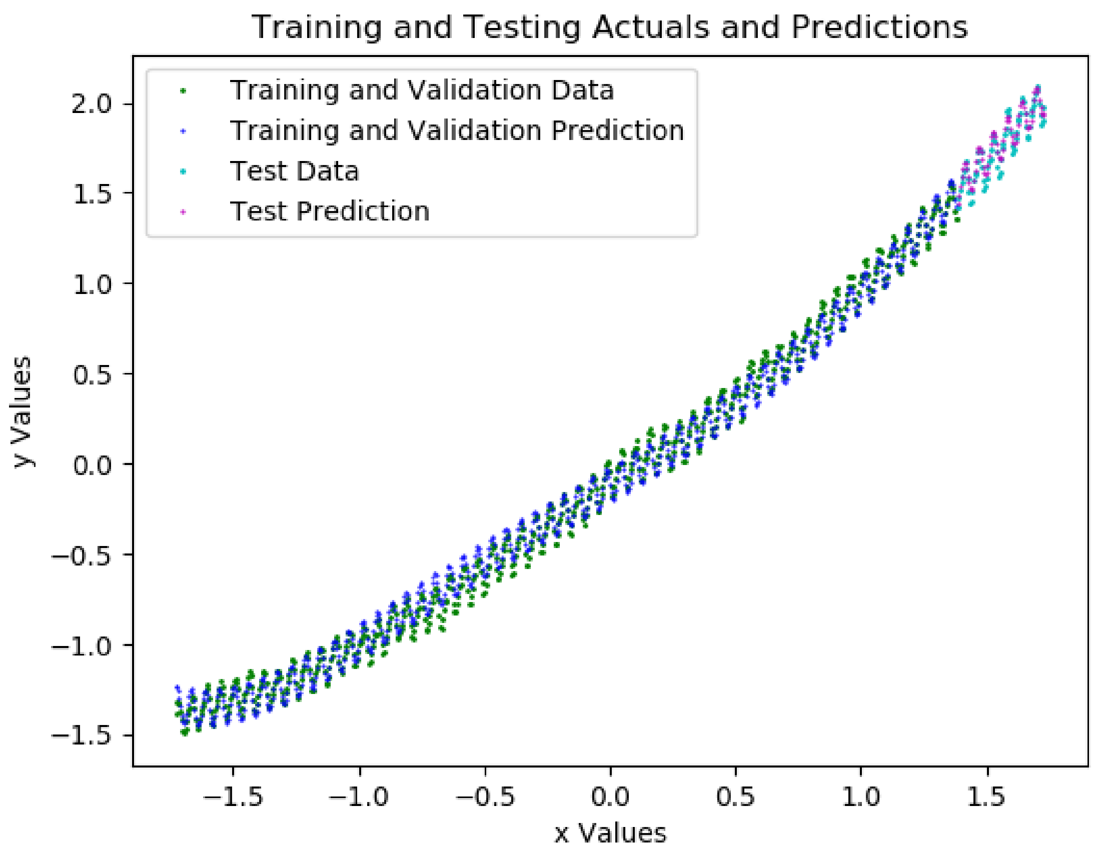


Figure 1: Results of SVD GLM

As one can see from the above figure, the predictions very closely track the actual values. Therefore, the chosen basis functions effectively characterize the mauna_loa dataset.

Question 2: Kernelized GLM

The computational costs and memory requirements of each method are summarized in Table 1 below:

Table 1: Computational and memory costs for the primal approach versus the dual representation

Method:	Time to Run:	Memory:
Dual Representation:	0.01981	7897088
Primal Approach:	0.002006	4096

The computational and the memory costs of the dual representation are greater than the primary approach. This is because the kernel is computationally expensive to compute. Figure 2 below is a plot of k(0, z) and k(1, z + 1) with z ∈ [−0.1, 0.1]:

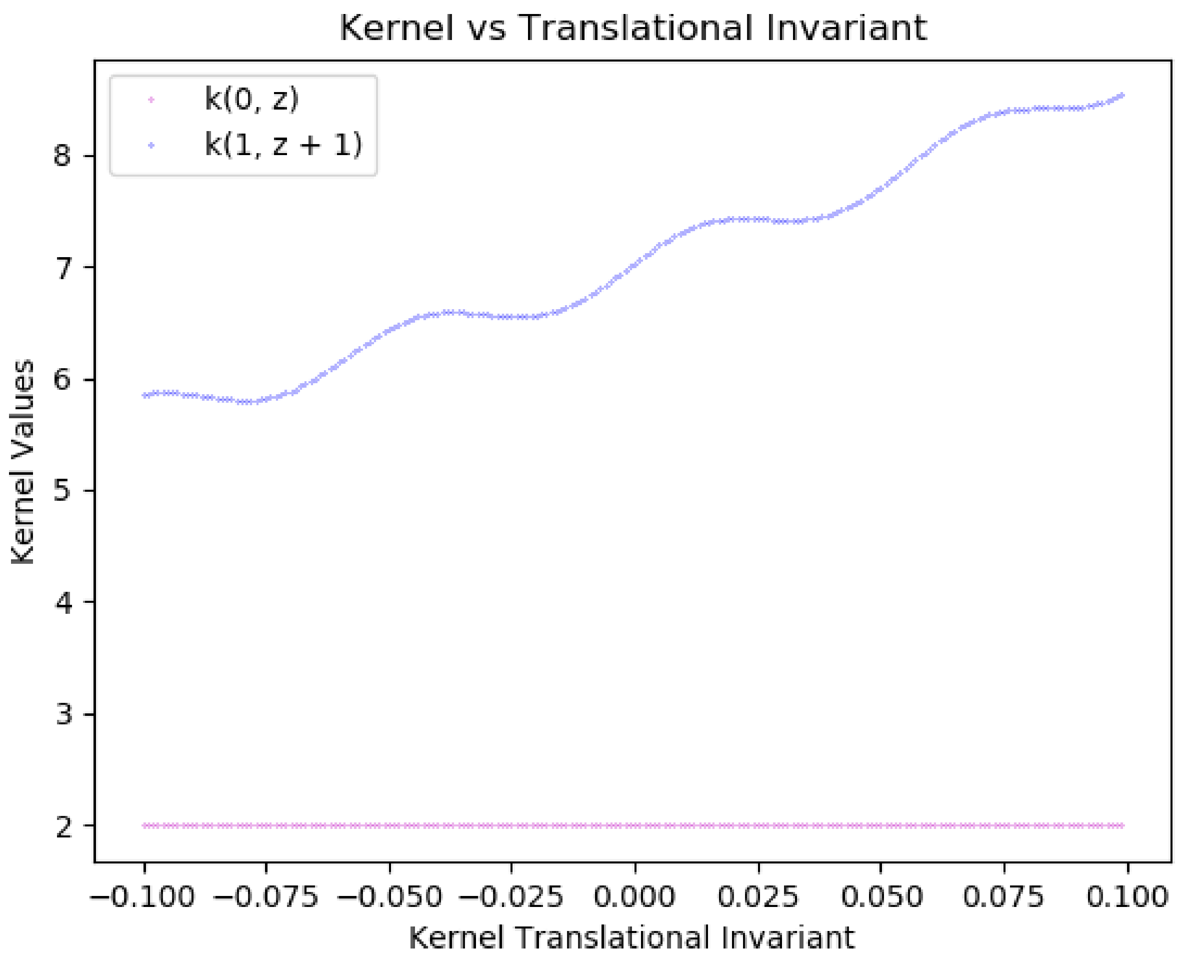


Figure 2: k(0, z) and k(1, z + 1) over z ∈ [−0.1, 0.1]:

From the figure above, we can see that k(0, z) and k(1, z + 1) are different. This implies that the kernel is non-stationary because a stationary kernel would remain the same given a unit translation. It makes sense that our kernel is non-stationary because it is not an RBF.

Question 3: Gaussian RBF GLM for Regression and Classification

Table 1 below delineates the optimal lambda and theta parameters calculated for each dataset. Table 2 also includes the test RMSE for each regression dataset and the accuracy for the classification dataset:

Table 2: Results for Gaussian RBF GLM

Dataset Type:	Dataset:	Test RMSE:	Accuracy:	Optimal Lambda:	Optimal Theta:
Regression	mauna_loa	0.1498	-	0.001	1
Regression	rosenbrock	0.1481	-	0.001	2
Classification	iris	-	100%	0.001	1

Question 4: Minimizer of Linear Regression Model Derivation

$$\underset{\mathbf{w}\in\mathbb{R}^D}{\arg\min}\ ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \mathbf{w}^T\mathbf{\Gamma}\mathbf{w}$$

Figure 3: Tikhonov regularization training problem

In this question the minimizer of a least-squares loss linear regression model with the training problem shown in Figure 3 above was solved in Figure 4 below:

$$\mathcal{L}(\underline{w}) = \underset{\underline{w}\in\mathbb{R}^D}{\arg\min}\ ||y - \underline{X}\underline{w}||_2^2 + \underline{w}^T\underline{\Gamma}\underline{w}$$

$$\frac{\partial\mathcal{L}(\underline{w})}{\partial\underline{w}} = -2(y - \underline{X}w)^T(\underline{X}) + 2\underline{w}^T\underline{\Gamma} = 0$$

$$\Longrightarrow \underline{w}^T(\underline{X}^T\underline{X} + \underline{\Gamma}) = y^T\underline{X}$$

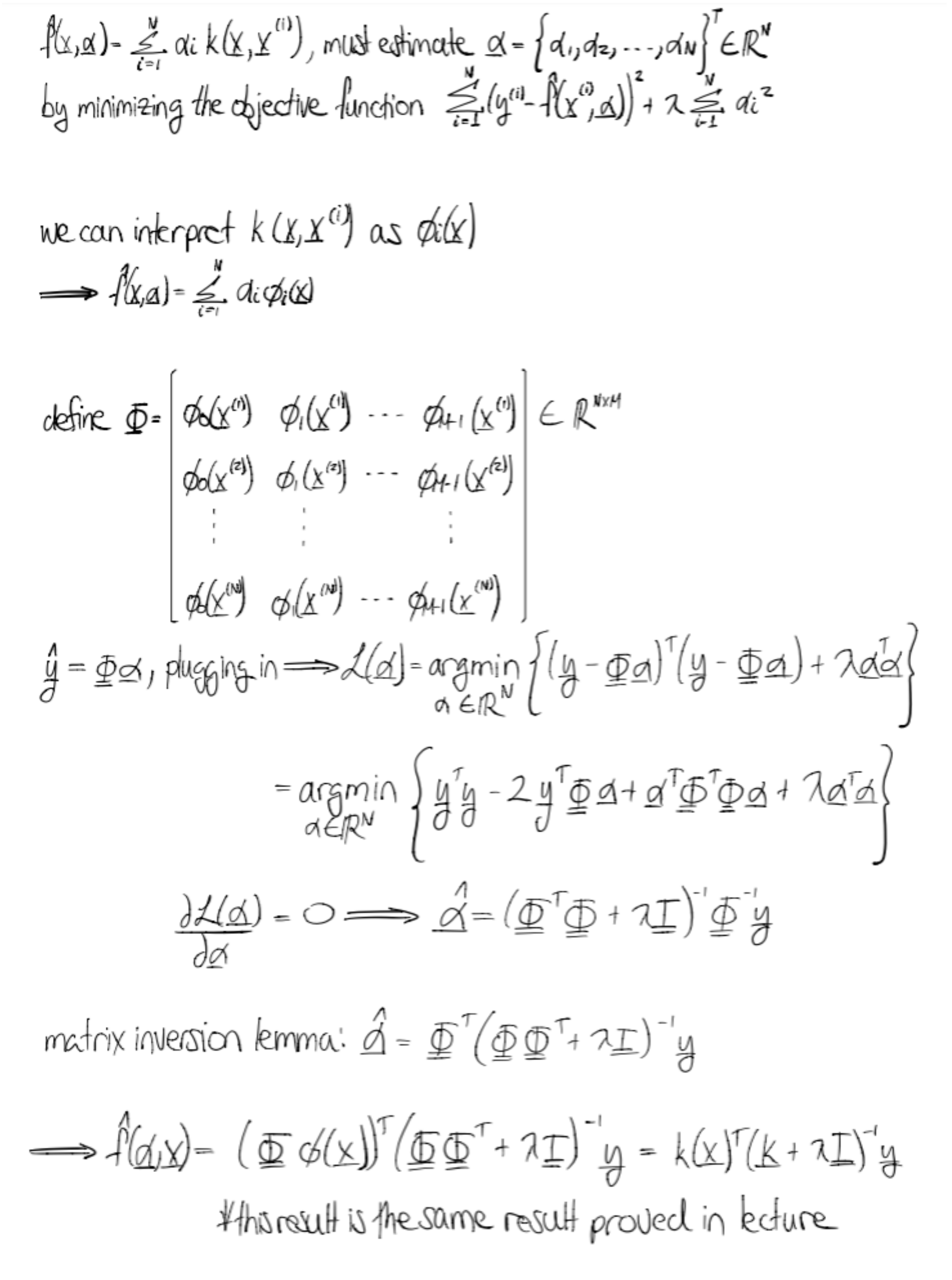
$$\underline{w}^T = y^T\underline{X}(\underline{X}^T\underline{X} + \underline{\Gamma})^{-1}$$

$$\underline{w} = (\underline{X}^T\underline{X} + \underline{\Gamma})^{-1}\underline{X}^T y$$

Figure 4: Derivation of least-squares loss minimizer for Tikhonov regularization training problem

Question 5: Estimating $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}^T$

Figure 5 below illustrates the derivation of a computational strategy to estimate α:



$f(\underline{x},\underline{\alpha}) = \sum_{i=1}^{N} \alpha_i k(\underline{x},\underline{x}^{(i)})$, must estimate $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}^T \in \mathbb{R}^N$

by minimizing the objective function $\sum_{i=1}^{N} (y^{(i)} - f(\underline{x}^{(i)},\underline{\alpha}))^2 + \lambda \sum_{i=1}^{N} \alpha_i^2$

we can interpret $k(\underline{x},\underline{x}^{(i)})$ as $\phi_i(\underline{x})$

$\Longrightarrow f(\underline{x},\underline{\alpha}) = \sum_{i=1}^{N} \alpha_i \phi_i(\underline{x})$

define $\underline{\Phi} = \begin{bmatrix} \phi_0(\underline{x}^{(1)}) & \phi_1(\underline{x}^{(1)}) & \cdots & \phi_{M-1}(\underline{x}^{(1)}) \\ \phi_0(\underline{x}^{(2)}) & \phi_1(\underline{x}^{(2)}) & \cdots & \phi_{M-1}(\underline{x}^{(2)}) \\ \vdots & \vdots & & \vdots \\ \phi_0(\underline{x}^{(N)}) & \phi_1(\underline{x}^{(N)}) & \cdots & \phi_{M-1}(\underline{x}^{(N)}) \end{bmatrix} \in \mathbb{R}^{N \times M}$

$\hat{\underline{y}} = \underline{\Phi}\underline{\alpha}$, plugging in $\Longrightarrow \mathcal{L}(\underline{\alpha}) = \underset{\alpha \in \mathbb{R}^N}{\operatorname{argmin}} \left\{ (\underline{y} - \underline{\Phi}\underline{\alpha})^T(\underline{y} - \underline{\Phi}\underline{\alpha}) + \lambda \underline{\alpha}^T\underline{\alpha} \right\}$

$$= \underset{\alpha \in \mathbb{R}^N}{\operatorname{argmin}} \left\{ \underline{y}^T\underline{y} - 2\underline{y}^T\underline{\Phi}\underline{\alpha} + \underline{\alpha}^T\underline{\Phi}^T\underline{\Phi}\underline{\alpha} + \lambda\underline{\alpha}^T\underline{\alpha} \right\}$$

$$\frac{\partial \mathcal{L}(\underline{\alpha})}{\partial \underline{\alpha}} = 0 \Longrightarrow \hat{\underline{\alpha}} = (\underline{\Phi}^T\underline{\Phi} + \lambda\underline{I})^{-1}\underline{\Phi}^{-1}\underline{y}$$

matrix inversion lemma: $\hat{\underline{\alpha}} = \underline{\Phi}^T(\underline{\Phi}\underline{\Phi}^T + \lambda\underline{I})^{-1}\underline{y}$

$\Longrightarrow f(\underline{\alpha},\underline{x}) = (\underline{\Phi}\,\phi(\underline{x}))^T(\underline{\Phi}\underline{\Phi}^T + \lambda\underline{I})^{-1}\underline{y} = k(\underline{x})^T(\underline{K} + \lambda\underline{I})^{-1}\underline{y}$

*this result is the same result proved in lecture

Figure 5: Derivation of computational strategy to estimate α

From the derivation shown in Figure 5 above, we can see that the results are the same. The results are the same because the kernel defined in this problem is the same thing as $\Phi\ \Phi^T$. After transforming from the kernelized representation to the representation in terms of Φ, we recover the same result.