Homework 2: Bayesian Estimation & Linear Regression

UC Irvine CS275P: Machine Learning with Generative Models

Homework due at 11:59pm on April 22, 2025

Full Name: Viveka Agrawal

UCINetID (e.g., ucinetid@uci.edu): vivekaa@uci.edu

Question 0: (5 points)

Canvas includes detailed guidelines on how homework solutions should be formatted for submission to Gradescope. To receive full credit on this question, be sure that you read and follow these guidelines carefully! In particular, please take care that:

- There are two Gradescope assignments for Homework 1. For the first assignment, submit a PDF containing your answers for questions 1-2. For the second assignment, submit your Python code for both questions, which will be checked by an autograder.
- For the PDF submission, you must fill in this PDF template with your answers. All pages must remain in their original order when uploading your assignment to Gradescope. *Do not rearrange, add, or remove any pages from the provided PDF template.*
- For multiple-choice questions, ensure that you *completely fill in the bubble next to your selected answer*, and provide an explanation or justification to support your answer.
- Enter your final answers in the designated answer boxes, and write your explanations and/or math justifications in the space provided below each question. *Always show the work needed to produce your answers, not just the final result.* Math may be handwritten, but if possible, please enter explanatory sentences as typed text.
- For both questions, use the provided Python file. After adding your code to the template, submit that file separately to the second Gradescope assignment. In addition to uploading your Python code, be sure you provide answers and explanations in the designated areas of the PDF.
- If any question asks you to create a plot, insert an image showing the plot in the PDF.
- Check that your answers (especially scanned math) are readable within Gradescope, and that content has not been cut-off by the page margins.

Question 1: (25 points for PDF answers, 15 points for Python code submission)

In the next two problems, we study different approaches to linear regression using a one-dimensional dataset collected from a simulated motorcycle accident. The input variable, x, is the time in milliseconds since impact. The output variable, t, is the recorded head acceleration. We have divided the full dataset into 40 training examples (variables `Xtrain` and `Ytrain`), and 53 test examples (variables `Xtest` and `Ytest`). To reduce numerical problems, all training features should be scaled to lie in the interval $[-1, +1]$ before fitting. Note that the same scaling must then be applied to all test data. We have provided a demonstration script to get you started.

We compare linear regression models with various families of non-linear basis functions. For a model of order L, we define $M = L+1$ basis functions: the constant function $\phi_0(x) = 1$, and L functions $\phi_j(x), j = 1, \ldots, L$, that vary with x. In your solutions, you do not need to report the numerical values $\hat{w}$ of estimated regression coefficients. We will assess correctness using the plots you create based on these estimates, and your submitted code.

In the standard linear regression model, the observations t_n follow a Gaussian distribution centered around a linear function w of a fixed set of basis functions:

$$p(t_n \mid x_n, w, \beta) = \text{Normal}(t_n \mid w^T \phi(x_n), \beta^{-1}).$$

Here, β is the inverse variance or precision. As derived in the textbook, given N training observations the maximum likelihood weights $\hat{w}$ satisfy the normal equations

$$(\Phi^T \Phi)\hat{w} = \Phi^T t,$$

where t is a N-dimensional vector of training responses t_n, and Φ is an $N \times M$ matrix of corresponding features, such that $\Phi_{ij} = \phi_j(x_i)$.

For all parts below, you should implement your own numpy code for computing regression parameter estimates, rather than using standard packages for linear models.

a) Use the provided function `basis_poly` *in* basis_utils.py *to define polynomial basis functions $\phi_j(x) = x^j$. Determine the maximum likelihood weight estimate $\hat{w}$ for a model of order $L = 5$. (You do not need to show anything in this part.)*

Hint: *To compute $x = A^{-1}b$ in Python, rather than explicitly calling the* `np.linalg.inv` *command, use the following command to improve numerical stability:*

```
>>> x = np.linalg.lstsq(A, b, rcond=-1)[0]
```

b) *Compute maximum likelihood (ML) estimates $\hat{w}$ of the regression parameters for models of order $L = 0, 1, 2, \ldots, 19$. Consider the following squared error loss function:*

$$L(x, t \mid \hat{w}) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (t_n - \hat{w}^T \phi(x_n))^2} \tag{1}$$

Evaluate and plot $L(x, t \mid \hat{w})$ as a function of the model order, L, for the 40 training examples. Also do this (on the same axes) for the 53 test examples.

Squared-error loss of Equation (1) versus L for training and test data:

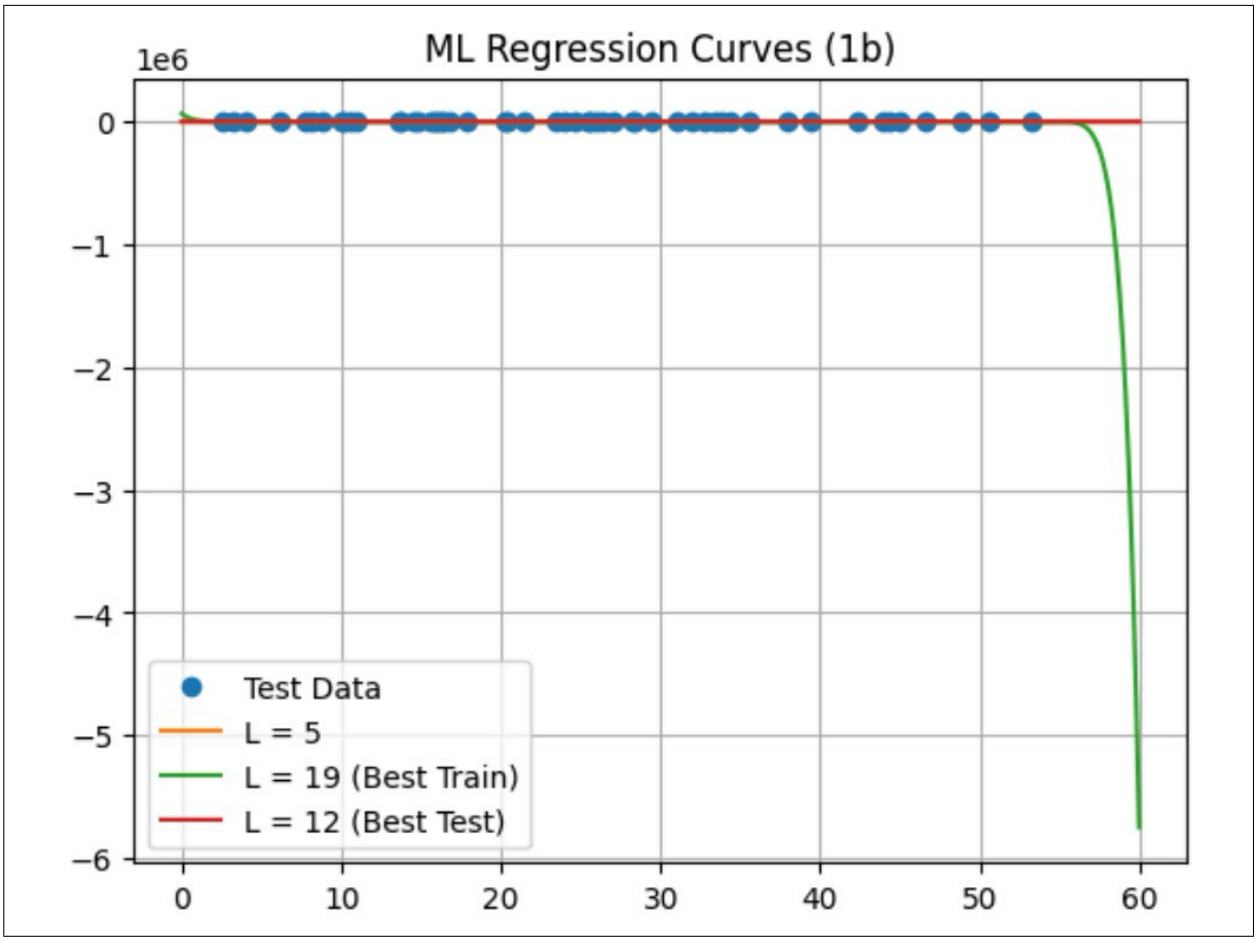
Which model has the smallest training error, and which has the smallest test error?

Model with smallest train error: $L =$ 19

Model with smallest test error: $L =$ 12

On a single set of axes, versus a dense grid of x values (see code), plot the test data and the mean predictions $\hat{w}^T\phi(x)$ for three models estimated via maximum likelihood: a model of order $L = 5$, the model with best train error, and the model with best test error.

Plot of test data and ML regression curves for three model orders L:

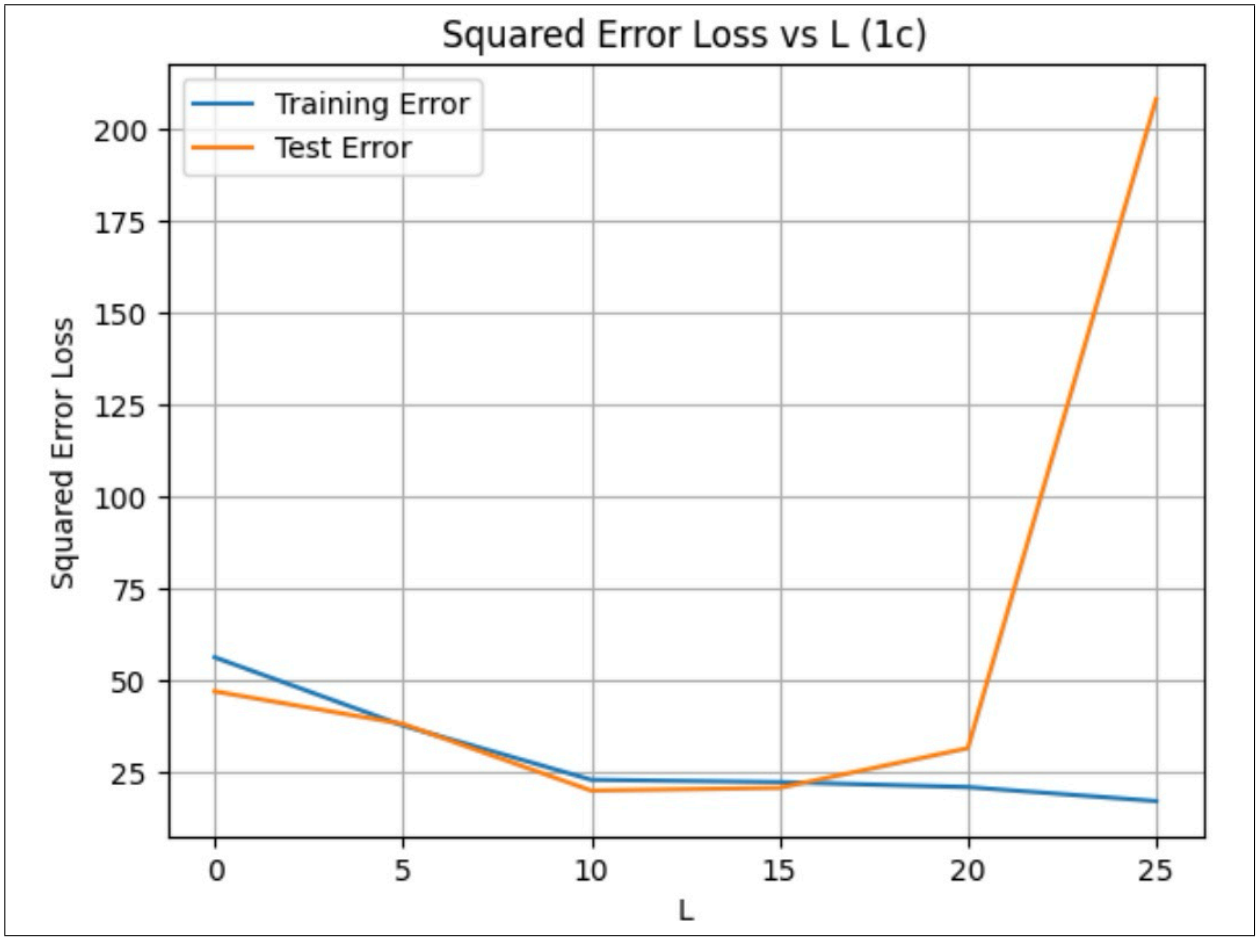


c) We now consider alternative, radial basis functions of the form

$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2\sigma^2}\right), \qquad j = 1, \ldots, L.$$

For any model order L, we space the basis function centers μ_j evenly between -1 and 1, and set the bandwidth $\sigma = (\mu_2 - \mu_1)$ to the distance between basis centers. See the provided function `basis_radial` *in* basis_utils.py. *Repeat parts (a-b) for radial basis function models of order $L = 0, 5, 10, 15, 20, 25$.*

Squared-error loss of Equation (1) versus L for training and test data:

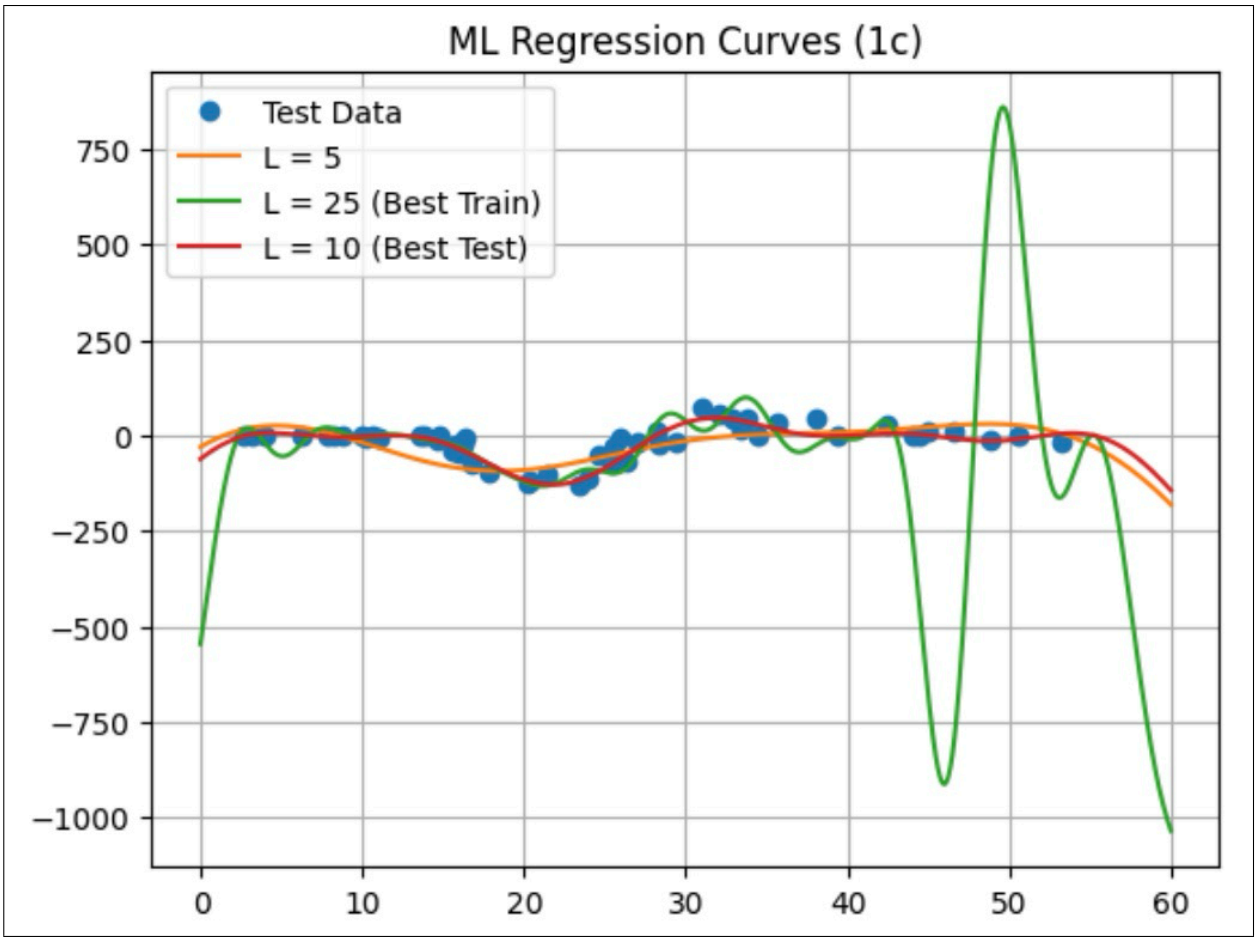


Which model has the smallest training error, and which has the smallest test error?

Model with smallest train error: $L =$ 25

Model with smallest test error: $L =$ 10

Plot of test data and ML regression curves for three model orders L:



d) *In constructing the training and test sets, we excluded one point from the original dataset: $x = 57.6$, $t = 10.7$. What is the error in the prediction of this point for the polynomial model which minimized the loss on the other test points? What is the error in the prediction of this point for the radial basis function model which minimized the loss on the other test points? Discuss differences between this datapoint and the other test data.*

Loss for polynomial model: 10555.490

Loss for radial basis function model: 3508.300

Brief discussion of how this excluded point differs from other test data:

The excluded point is near the upper end of the range used for training and testing. This point may be extrapolated unlike the other test data which is primarily clustered at lower x values. Polynomial models may produce high predictions at boundaries. Radial basis models may underestimate at boundaries which can lead to high error.

Question 2: (40 points for PDF answers, 15 points for Python code submission)

In the previous question, you may have noticed that unregularized ML estimates can become unstable for large model orders. We now consider an alternative, Bayesian approach in which the regression coefficients are assigned a Gaussian prior

$$p(w) = \text{Normal}(w \mid 0, \alpha^{-1} I_M)$$

where $\alpha > 0$ is an inverse-variance hyperparameter. As derived in the textbook, the posterior mean m_N and covariance S_N of the weight vector, under a Gaussian prior, equal

$$m_N = (\Phi^T \Phi + \frac{\alpha}{\beta} I_M)^{-1} \Phi^T t, \qquad S_N = (\beta \Phi^T \Phi + \alpha I_M)^{-1},$$

where I_M is an $M \times M$ identity matrix.

a) Give an expression for the MAP estimate of w under the Gaussian prior above, and the linear observation model of question 1. (A proof is not necessary, you may use results from lecture.) For a polynomial basis of order $L = 50$ and $N = 40$ training examples, is there a unique MAP estimate? Is there a unique ML estimate? Explain your reasoning.

MAP estimate of $w =$ $(\Phi^T \Phi + \alpha \beta^{-1} I)^{-1} \Phi^T t$

Is there a unique MAP estimate?

● ***Yes*** ○ ***No***

Posterior: $p(w|t,X) \propto p(t|w) p(w)$

Likelihood: $p(t|w) = N(t|\Phi w, \beta^{-1} I)$

Prior on w: $p(w) = N(w|0, \alpha^{-1} I)$

Is there a unique ML estimate?

○ ***Yes*** ● ***No***

Posterior: $p(w|t,X) \propto \exp\left(-\frac{\beta}{2}\|t - \Phi w\|^2\right) \exp\left(-\frac{\alpha}{2}\|w\|^2\right)$

Brief explanation for uniqueness of MAP and ML estimates:

Take log: $\frac{\beta}{2}\|t - \Phi w\|^2 + \frac{\alpha}{2}\|w\|^2$

ML estimate is not unique because $\Phi^T \Phi$ may be singular for large values of L which can lead to non-uniqueness and multiple solutions for w.

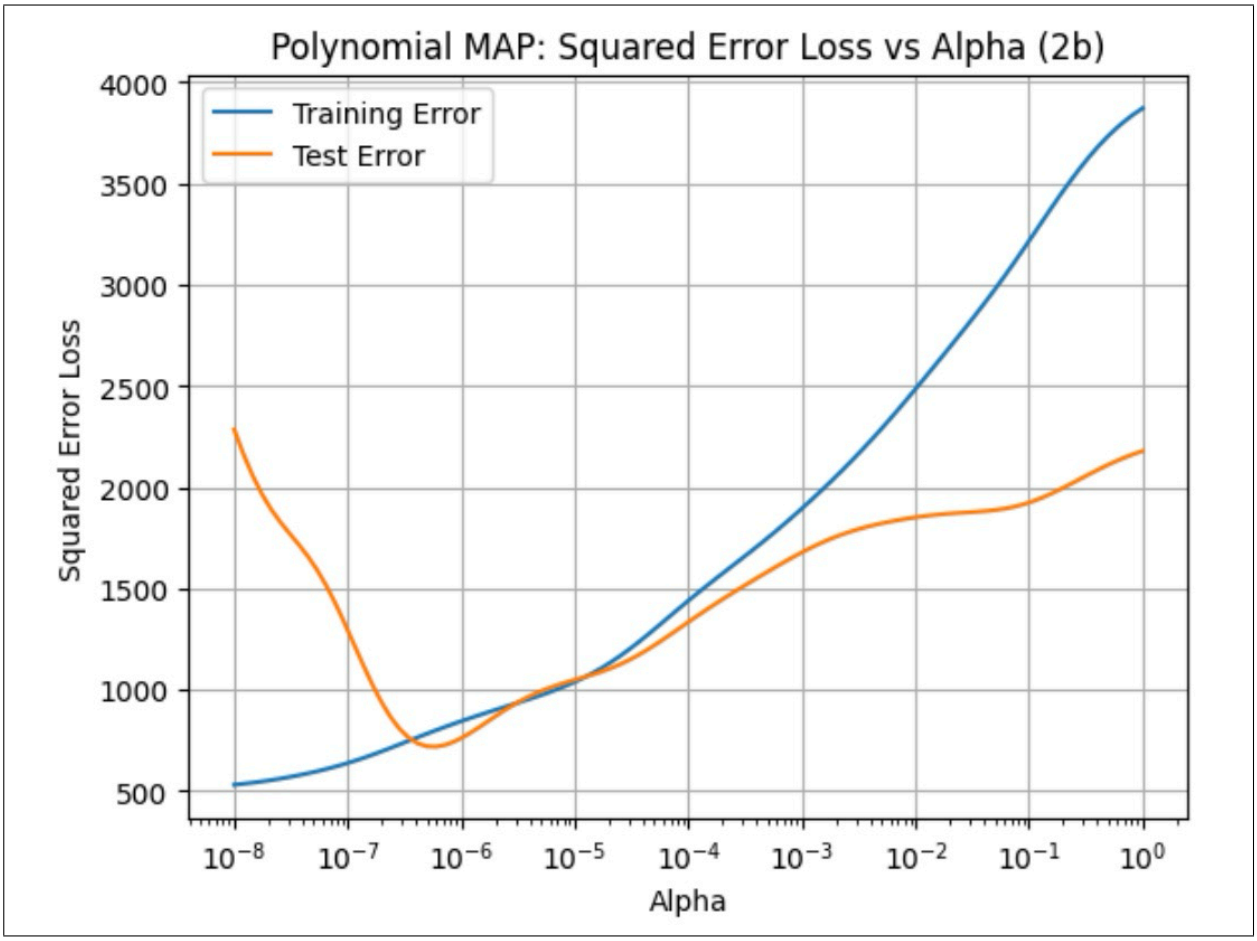
MAP estimate is unique because the regularization term α allows the matrix $\Phi^T \Phi + \alpha \beta^{-1} I$ to be invertible even for large values of L.

b) Fix $\beta = 0.0025$, and consider 100 candidate values for the regularization parameter α, logarithmically spaced between 10^{-8} and $10^{0} = 1.0$:

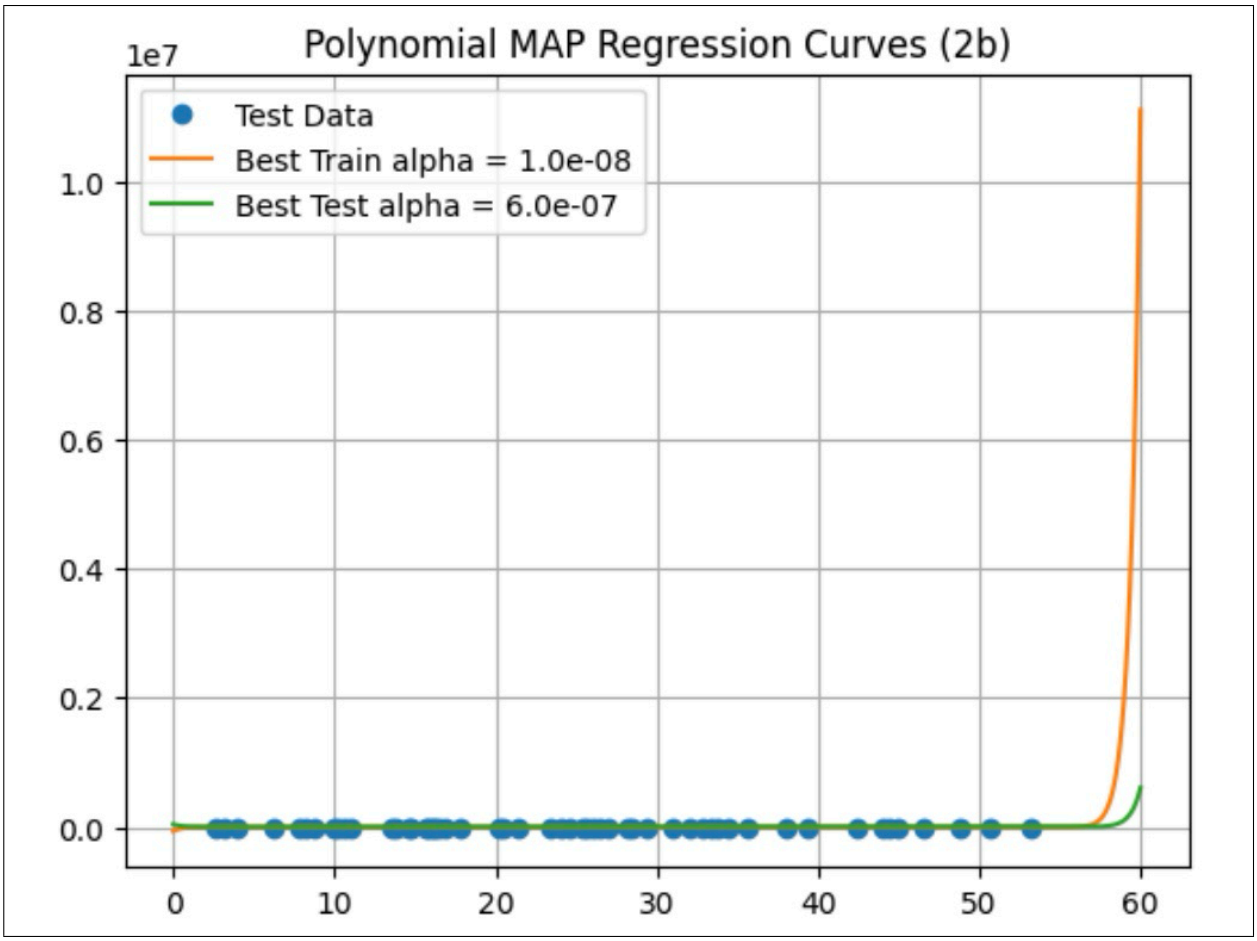
```
>>> alpha = np.logspace(-8,0,100)
```

Determine MAP estimates $\hat{w}$ for each of these priors, and a polynomial basis of order $L = 50$. Using the `pyplot.semilogx` *command, plot the error metric of Equation (1) versus α for both the training and test datasets. Then plot the mean prediction $y(x) = \hat{w}^T\phi(x)$ for the models which minimize the training and test error.*

Squared-error loss of Equation (1) versus α for training and test data:

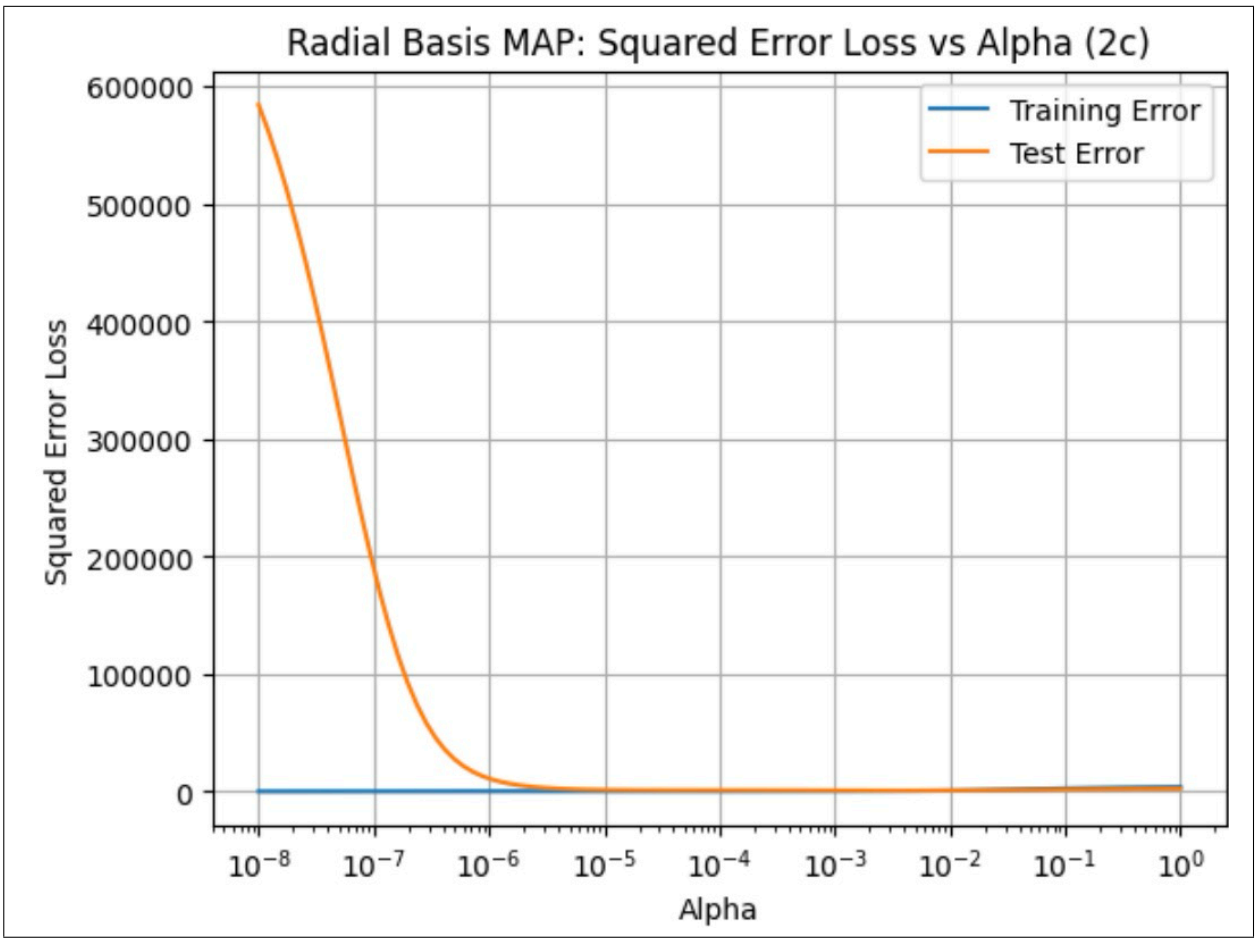


Plot of test data and MAP regression curves for two regularization parameters α:

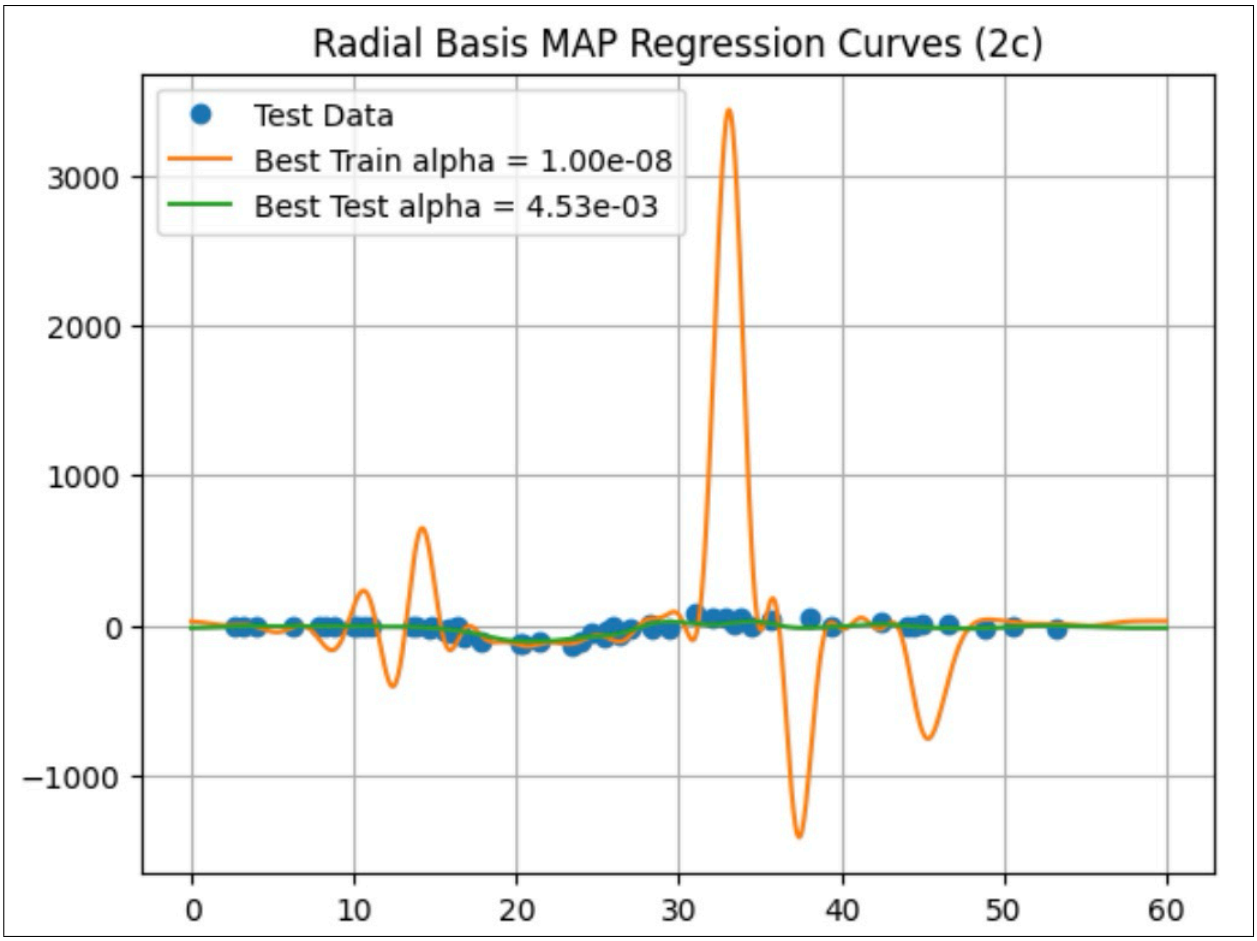


c) Repeat part (b) for a radial basis function model of order $L = 50$.

Squared-error loss of Equation (1) versus α for training and test data:



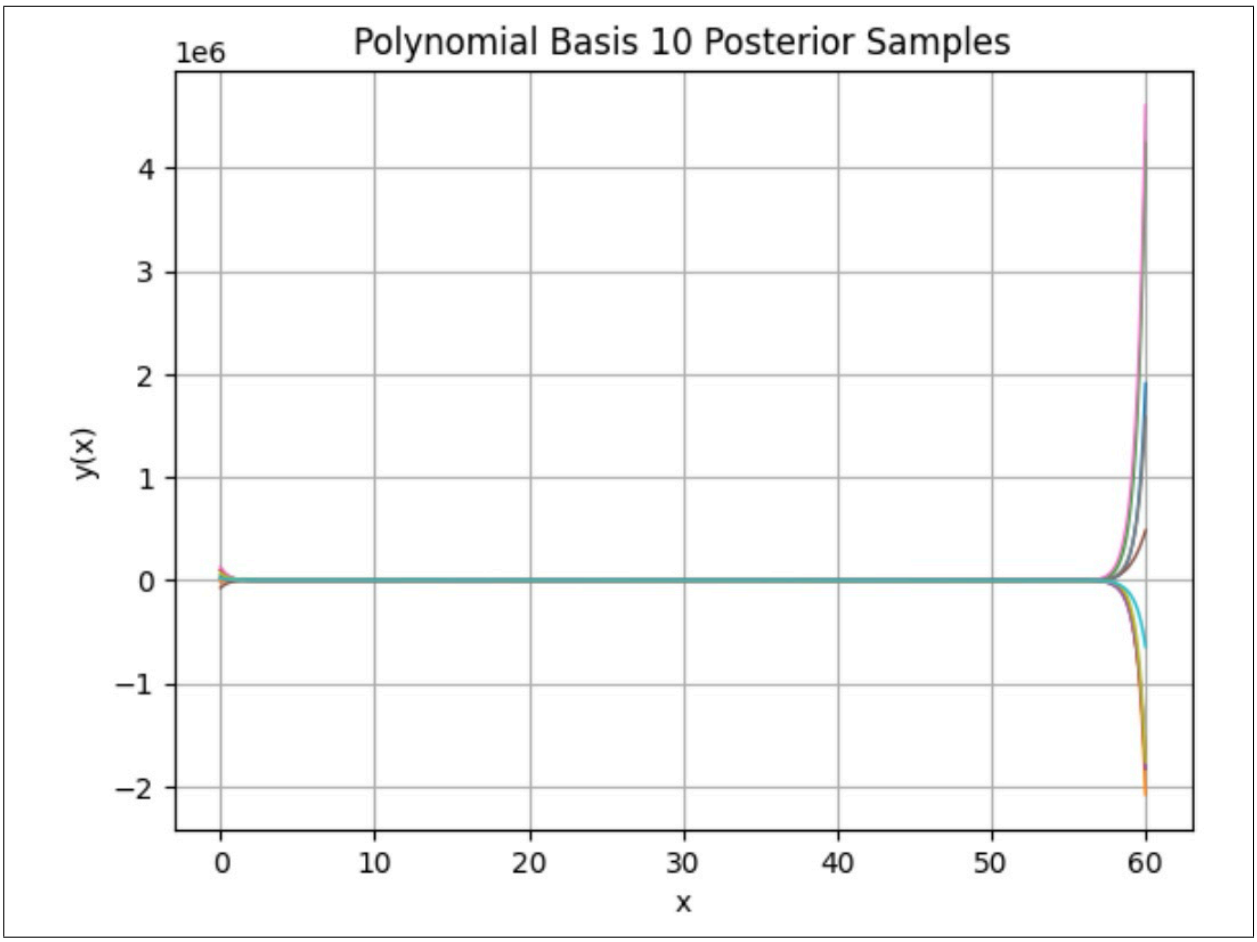
Plot of test data and MAP regression curves for two regularization parameters α:



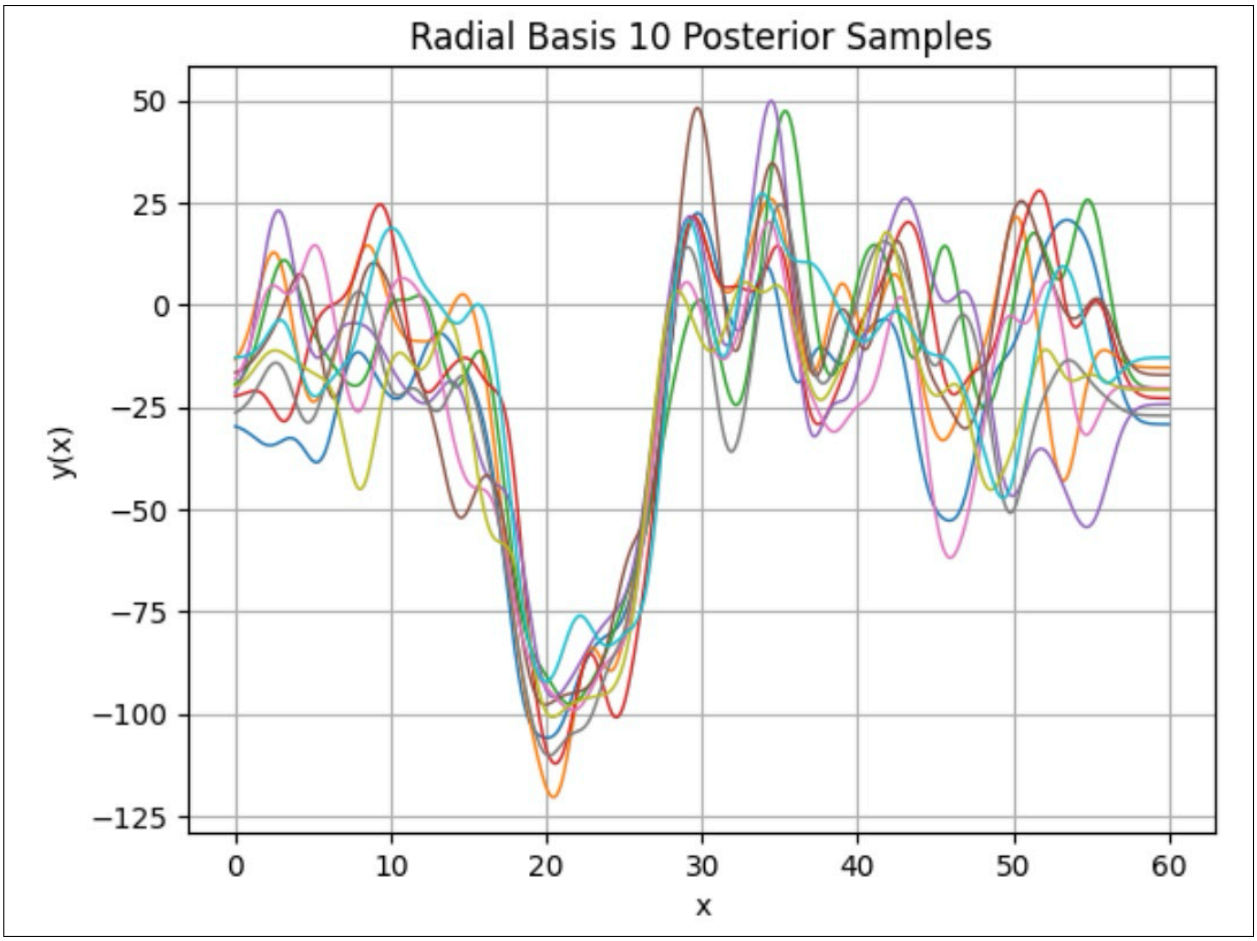
d) Consider the pair of models from parts (b,c) which minimize the test error for their corresponding basis families. We can simulate from the posterior of the predictive function $y(x)$ by first sampling weights $\tilde{w} \sim \text{Normal}(w \mid m_N, s_N)$ from their posterior distribution given the training data, and then setting $y(x) = \tilde{w}^T \phi(x)$. Draw and plot 10 samples from the polynomial and radial basis function posteriors on prediction functions, for the grid of x values defined in the example code.

Hint: *The numpy* `np.random.multivariate_normal` *command samples from a multivariate normal distribution.*

Plot of 10 curves sampled from posterior of polynomial regression model:



Plot of 10 curves sampled from posterior of radial basis regression model:



e) *Again consider the pair of models from part 2(d). What is the error in their corresponding predictions of the held-out test point from problem 1(d)? Are the relative magnitudes of these errors predictable from the posterior distributions plotted in part 2(d)?*

Squared Loss for polynomial model: 52793560.695

Squared Loss for radial basis function model: 748.163

Brief discussion of how these errors relate to the posterior distributions:

The polynomial model posterior samples showed more variance and overfitting at the boundaries of the data range at which the excluded point is at, leading to a large error in predicting the excluded point. The radial model showed more stable posterior curves even near the boundaries and therefore resulted in a smaller loss at the excluded point. As a result, greater spread indicates less confidence and larger potential prediction error.