Assignment 2: Regularization and Model Evaluation

Alexandre St-Aubin, Jonathan Campana, & Jake Gameroff

Comp 551: Applied Machine Learning

September 30, 2024

Abstract

This paper explores how non-linear basis functions can be used to help linear regression models fit non-linear data. A data set of 100 points is generated from a non-linear function that incorporates sinusoidal components and Gaussian noise. The original features transformed using Gaussian basis functions, allowing the linear model to capture non-linear relationships. We compare models using varying numbers of basis functions and investigate how much the model overfits or underfits its given training data.

We also explore the L1 (Lasso) and L2 (Ridge) regularization using gradient descent. The regularization terms are found to affect the model weights differently: L1 yields sparsity, while L2 favours uniformly shrunk weights. We then use cross-validation to select optimal model parameters, such as the number of basis functions or the regularization strength. This work provides insights as to how model complexity might be balanced against generalization in non-linear regression problems.

Summary of Main Results

In this assignment we use Gaussian basis functions to enhance linear regression models, allowing them to fit non-linear, noisy data. For the particular dataset generated for this assignment, we found that the model performs best with 10 basis functions, minimizing validation error and avoiding overfitting. When using regularization, the optimal regularization strengths were $\lambda = 0.01$ (for L1) and $\lambda = 0.1$ (for L2), and the learning rate which was optimal for gradient descent was $\alpha = 0.05$. When implementing cross validation to find the MSE and bias-variance decomposition, it was found that the optimal $\lambda = 0.01668$ (for L1) and $\lambda = 1.9307$ (for L2). We also found that L1 regularization encouraged sparsity by driving certain weights to zero, while L2 regularization reduced all weights uniformly without eliminating any.

1 Linear Regression with Non-Linear Basis Functions

1.1 Data Generation

To simulate real-world noisy data, we generated a dataset of 100 data points sampled from the non-linear function $y(x) = \sin(\sqrt{x}) + \cos(x) + \sin(x) + \varepsilon$, where x is sampled uniformly in the range $[0, 20]$, and ε is Gaussian noise with mean 0 and variance 1, sampled independently for each data point. The generated data points are visualized in Figure 1, alongside the underlying function without noise for comparison.

1.2 Non-Linear Basis Functions

To model the non-linear relationship in the data, we transformed the original features using Gaussian basis functions and fit a linear regression model to the transformed features. The Gaussian basis functions are defined as follows:

$$\phi(x, \mu, \sigma) = \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right)$$

where μ is the mean of the Gaussian function, and σ is the standard deviation. We applied Gaussian basis functions of increasing complexity by varying the number of basis functions D, starting from 0 up to 100 Gaussians. The means μ were spaced evenly within the range of the generated data points, i.e. $\mu \in [0, 20]$.

This transformation allows the linear model to capture non-linear patterns in the data by mapping the input features into a higher-dimensional space where the relationship between the features and the target variable becomes more linear. We plot 10 basis functions in Figure 2 for reference.

1.3 Model Fitting

We fit a series of linear regression models using Gaussian basis functions, varying the number of basis functions D from 0 to 100 in steps of 10. For each model, the basis functions were applied to transform the input features into a higher-dimensional space, as described in the previous section.

We plotted the fitted curves for each model, alongside the original noisy data and the true underlying function, across different σ in Figures 6 5 3 4. We notice that the smaller σ is, the less bases are needed for the model to overfits. For example, with $\sigma = 0.1$ and 100 basis functions, the model is able to fit through every data point, which does not happen with larger values of σ. With $\sigma = 2$, on the other hand, the model does not seem to overfit, even with 100 bases.

We also experimented with other basis functions, namely polynomial, sigmoid, tanh, and radial. Plots of each basis function are illustrated in Figure 8. We report the results in the next section.

1.4 Model Selection

To select the optimal model complexity, we split the data into a training set and a validation set. For each linear regression model with varying numbers of Gaussian basis functions, we computed the sum of squared errors (SSE) on both the training set and the validation set. The values are shown in Table 1. We plotted the SSE for both the validation and training sets as a function of the number of bases used in Figure 7. To select the right model, we use the validation set SSE, and compare it to the training set SSE to evaluate overfitting. The optimal number of Gaussian basis functions is easily seen to be 10, as it minimizes the validation SSE, with an average value of 22.29 across seeds. This optimal model provides a balance between bias and variance.

As the number of Gaussian basis functions increases, the model's behavior changes in a clear pattern. With too few basis functions, the model underfits the data, as it lacks the capacity to capture the non-linear patterns in the true function. This is evident from both the high training and validation SSEs. As we add more basis functions, the model becomes more expressive and starts to better capture the underlying non-linear structure, leading to lower SSE values on both the training and validation sets. However, as the number of basis functions increases beyond 10, the model starts to overfit the training data, capturing noise rather than the true signal. This results in a significant decrease in training SSE, but the validation SSE begins to rise again, indicating poorer generalization performance.

For the **polynomial, tanh, sigmoid,** and **radial** basis functions, the optimal number of bases consistently fell within the range of 10 to 15. Despite this similarity in the optimal range, the behavior of the fitted function varies significantly across these different bases, as illustrated in Figure 8. Each basis function exhibits a distinct progression as the number of bases increases.

2 Bias-Variance Tradeoff with Multiple Fits

2.1 Setup

With the model selection complete, we now turn to visualizing the fitted models alongside the true underlying function, averaging the results to account for variability. This will help us understand how model complexity affects bias and variance.

For a given number D of basis functions, we repeat the process in Section 1 a total of 11 times. Specifically, for each number $D \in \{0, 10, 20, \ldots, 100\}$, we repeat the following 10 times:

1. Generate our dataset as in Subsection 1.1.
2. Fit a linear regression model to the data using D basis functions.

For each D, we plot all 11 fits along with their average. Figure 9 shows the results over all $D \in \{0, 10, 20, \ldots 100\}$.

2.2 Bias-Variance Tradeoff Analysis

The bias-variance tradeoff is the balance between a model's ability to fit the training data (bias) and its sensitivity to noise in the data (variance). A model with high bias may underfit the data by being too simplistic, while a model with high variance may overfit the data by being overly complex, capturing noise instead of general patterns.

This balance is reflected in the number of basis functions we employ in our model. We will evaluate the model's performance according to the quantity of basis functions D. Because the model is too simplistic to describe the underlying function $y(x)$, it performs poorly when $D = 0$. This indicates that the model underfits the data and is unable to accurately fit to the underlying function. The main issue is the strong bias, which causes extreme underfitting, although the model's simplicity yields minimal variance (the green lines overlap considerably).

However, the model also performs poorly when $D = 100$, but for a different reason: an excessive number of basis functions causes the model to become very complex, so it begins to overfit the data noise. Because each model fit depends heavily on the training data, this results in large variance. The individual models are highly sensitive to changes in the data, which results in poor performance when the model is applied to new data, even though the average fit matches the underlying function rather well. In this case, the model has significant variance but little bias.

The model performs best at intermediate values of D, particularly between $D = 30$ and $D = 50$. In this range, the model learns the structure of the true function without fitting the noise.

Figure 10 shows the relationship between the number of basis functions used in the model and the corresponding training and test errors. The training error measures how well the model fits the training data, and the test error evaluates the model's performance on unseen data.

The training error continuously drops as the number of basis functions rises. This is to be expected since a more complex model can fit its training data more easily. Even while the training error keeps decreasing, the test error starts to rise after a certain point (about 20 basis functions). The model is fitting the noise in the training data instead of capturing the underlying distribution, which is a classic sign of overfitting. When the test error is minimal, which happens when using around 10–20 basis functions, the model performs optimally.

3 Regularization with Cross-Validation

3.1 Adding Regularization

To add L1 (Lasso) and L2 (Ridge) regularization to our linear model, the linear regression class was modified to take as parameter the type of regularization we would like to apply. L1 regularization was done with a gradient descent since lasso regression does not have a closed form due to the fact that the absolute value function does not have a derivative at 0. The decomposition of the derivative is the following:

$$\frac{dj}{dw} = \frac{1}{m}\left(\left(\sum_{i=1}^{m} y^{(i)} - \hat{y}^{(i)}\right) + \lambda\right) \quad \text{if w} > 0 \tag{1}$$

$$\frac{dj}{dw} = \frac{1}{m}\left(\left(\sum_{i=1}^{m} y^{(i)} - \hat{y}^{(i)}\right) - \lambda\right) \quad \text{if w} \leq 0 \tag{2}$$

For L2 regularization, we simply add an additional regularization term $\lambda \mathbf{I}$ to the closed form.

3.2 Cross-Validation

Cross-validation was implemented with 10-fold cross validation. The model was split in 10 fold, where a model was trained on 9 folds and validated on the remaining fold. This was done 10 times for each fold to be used as the validation fold. At each iteration, the training error and validation error was recorded, and was averaged out with every fold in the cross validation. This was repeated for a set of λs, for both L1 and L2. This can be seen in figure 11 and 12. It can be observed that the train MSE was at a plateau for a few lambdas, but would start to drastically increase when approaching 0.1. The validation MSE saw a small when increasing the lambda on the log scale to 0.01, but started to increase like the train MSE at 0.1. The train MSE for L2 regularization saw a steady increase, and started to increase at a more rapid rate when lambda was approaching 10. The validation error started high, and decreased to approximately 1.2, before increasing once again when lambda approached 10. This is expected since

when we begin to increase lambda by large amounts, the weights of the model decrease, and thus the model becomes worse at predicting data.

3.3 Bias-Variance Decomposition

The bias-variance decomposition was implemented with 10-fold cross validation, where the model were averaged over 50 datasets. In the L1 plot in 13, we can see that for small lambda, the bias squared is less than the variance and is increasing as a function of lambda whereas the variance is decreasing as a function of lambda. They are almost equal for lambda approximately equal to 0.01, and for larger lambda, the bias squared becomes greater than the variance and increases very rapidly. Thus the optimal lambda for L1 would be approximately 0.01668. A similar relationship is seen can be seen in the L2 bias-variance decomposition plot in 14. The variance is greater than the bias squared until they equal it each other at approximately 1.9, before the bias squared begins to increase more quickly and the variance continues to decrease. Thus the optimal lambda is approximately 1.9307 where, the variance and bias are approximately equal while still being relatively low. With these plots we are able to see the expected bias-variance trade off, and make our choice of lambdas with where bias and variance are both being minimized.

4 Effect of L1 and L2 Regularization on Loss

4.1 Data Generation & Setup

We begin by generating synthetic data based on the linear relationship $y = -4x + 10 + 2\varepsilon$, where ε is Gaussian noise with a mean of 0 and variance of 1. A total of 50 data points are generated, with x uniformly distributed between 0 and 10. This dataset forms the foundation for the subsequent analysis of L1 and L2 regularization.

We use gradient descent to investigate the effects of Lasso (L1) and Ridge (L2) regularization. In particular, we are interested in how changing regularization strengths (values of λ) affect the model fits. We would like to understand how each type of regularization penalizes the weights and influences model performance (specifically overfitting and underfitting).

4.2 Plotting the Loss Function and Visualizing Gradient Descent

For L1 and L2, we plot the loss contours over varying regularization strengths. We also overlay the trajectory of the gradient descent optimizer on the contour plots. This helps us examine how the optimization path changes as λ increases. In summary, we find that L1 encourages sparsity, and L2 gradually decreases large weights.

4.3 Analysis of Results

We can observe the effects of L1 and L2 regularization on the optimization process through the plots in Figure 15, which illustrate how varying the regularization parameter λ influences the model weights. The contrast between the two techniques can be seen from their impact on weight tuning.

L1 regularization is particularly effective at promoting sparsity by driving certain weights to zero. Notice the left-hand column of Figure 15: as λ increases, the red optimization paths follow sharp angles along the axes, showing the point where the algorithm forces weights to zero. This effect becomes even more pronounced with larger values of λ because it encourages the model to eliminate less significant weights.

In contrast, L2 regularization operates differently. Instead of pushing weights to zero, it penalizes larger weights more uniformly. The right-hand column of Figure 15 shows smoother optimization paths under L2 regularization, even for higher values of λ. The weights are gradually reduced, but none are completely driven to zero. This indicates that L2 regularization favours shrinking all weights proportionally, as opposed to selectively zeroing them out.

The influence of λ on both regularization methods is also important. For small λ values (as seen in the top row of Figure 15), the regularization affect is relatively weak, allowing the weights to remain larger and the paths more spread out. As λ increases (bottom row), the regularization strength intensifies, leading to more constrained paths. L1 regularization sharply cuts the weights to zero, while L2 regularization results in smoother reduction without completely eliminating any weights.

Conclusion

In this assignment, we explored the effectiveness of linear regression models using non-linear basis functions, particularly Gaussian bases, in capturing complex, non-linear relationships in data. Through model selection, we found that using around 10 Gaussian basis functions gives an optimal balance between bias and variance, making the model generalize to unseen data.

Also, exploring the bias-variance tradeoff across multiple fits revealed how increasing model complexity can lead to overfitting, with the model becoming sensitive to noise. We used regularization methods like L1 (Lasso) and L2 (Ridge) to help mitigate this overfitting by penalizing large weights. L1 regularization was shown to promote sparsity, driving certain weights to zero, while L2 regularization more evenly shrinks the model weights, preventing overfitting without driving weights to zero.

Overall, the incorporation of non-linear basis functions combined with regularization techniques provides a robust approach for handling both underfitting and overfitting in linear regression models, allowing for more flexible and accurate modeling of non-linear data.

Creativity

1. In section 1.3, we fitted the models with normal bases of varying σ and analysed the impact it had, rather than using only $\sigma = 1$.
2. In sections 1.3 and 1.4, we fitted the model with the following additional basis functions: polynomial, tanh, sigmoid, and radial. We reported on the results.
3. We implemented K-fold cross validation for the Bias-Variance decomposition in sections 3.2 and 3.3.

Contributions

- Alex: I did task 1. I experimented with the different σ values and basis functions in task 1.
- Jake: Implementation for tasks 2 and 4; write up for abstract and tasks 2 and 4; helped Jon with task 3 cross-validation implementation.
- Jonathan: I did task 3, worked with Jake in making the Gaussian bases function. I also did the k-fold in the bias-variance decomposition.

Appendices

A Linear Regression with Non-Linear Basis Functions

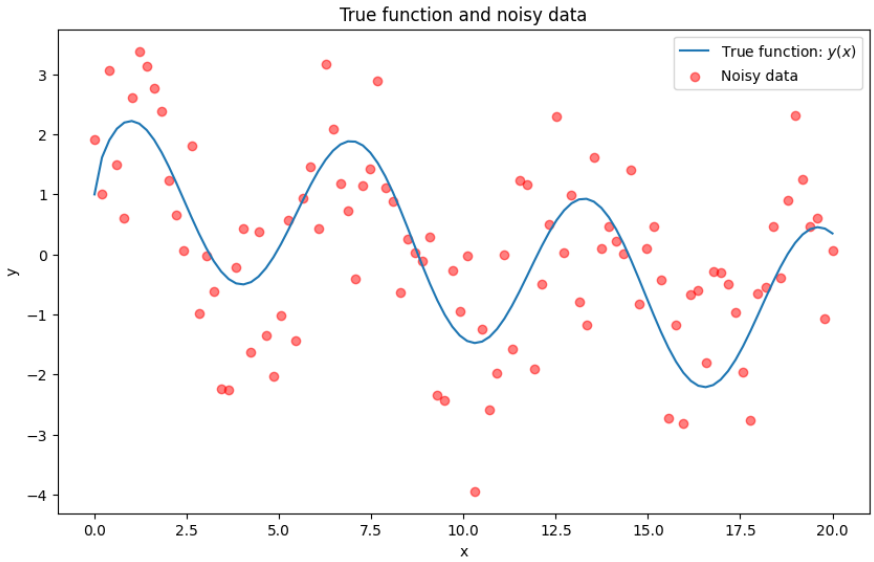


Figure 1: Plot of the true function $y(x)$ along with points defined as $y(x) + \varepsilon$, where ε is randomly sampled from a normal distribution with $\mu = 0$, $\sigma = 1$.

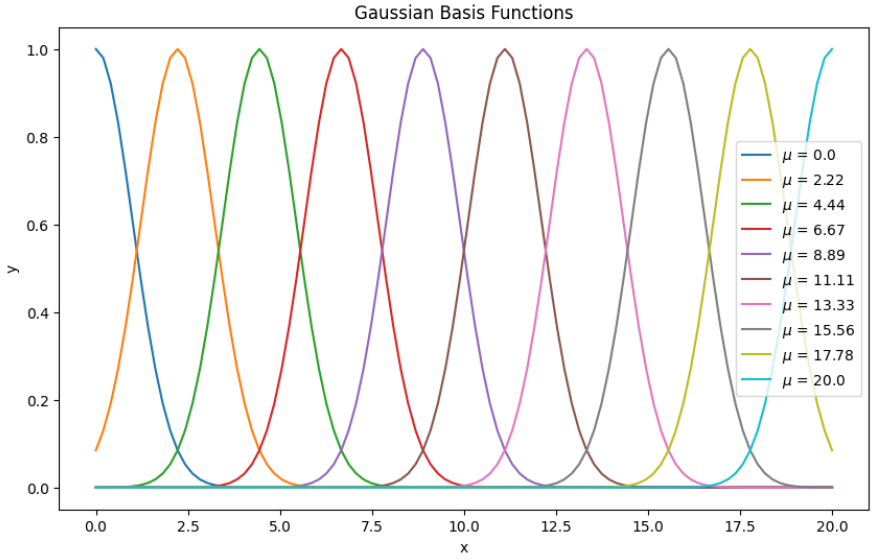


Figure 2: Plot of 10 Gaussian basis functions with $\sigma = 1$.

B Bias-Variance Tradeoff with Multiple Fits

C Regularization with Cross-Validation

D Effect of L1 and L2 Regularization on Loss

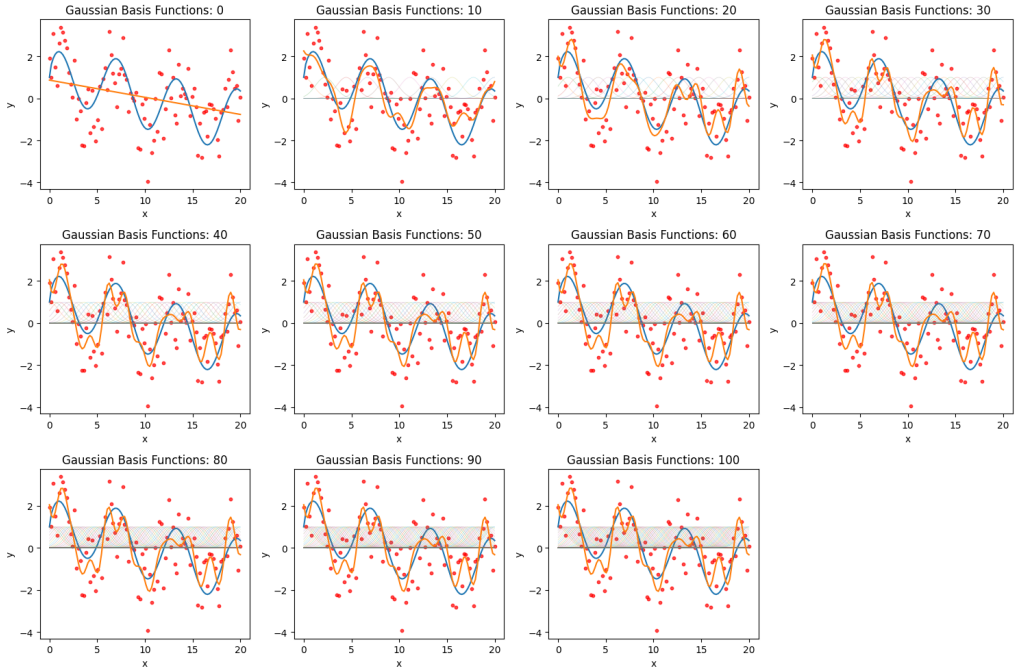


Figure 3: Comparison of the fitted models across different numbers of bases, $\sigma = 1$.

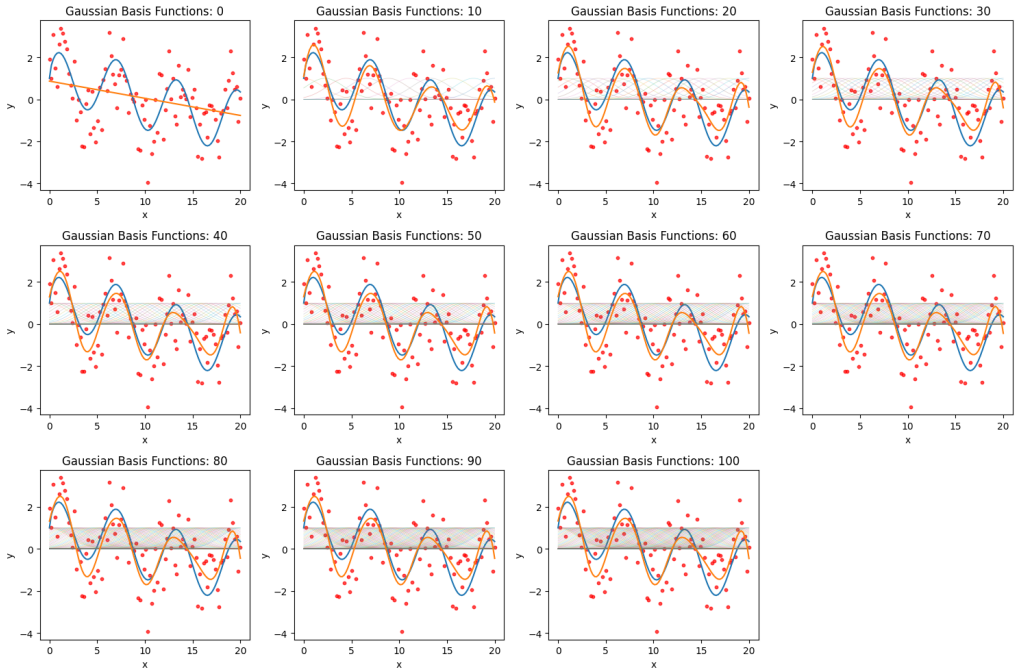


Figure 4: Comparison of the fitted models across different numbers of bases, $\sigma = 2$.

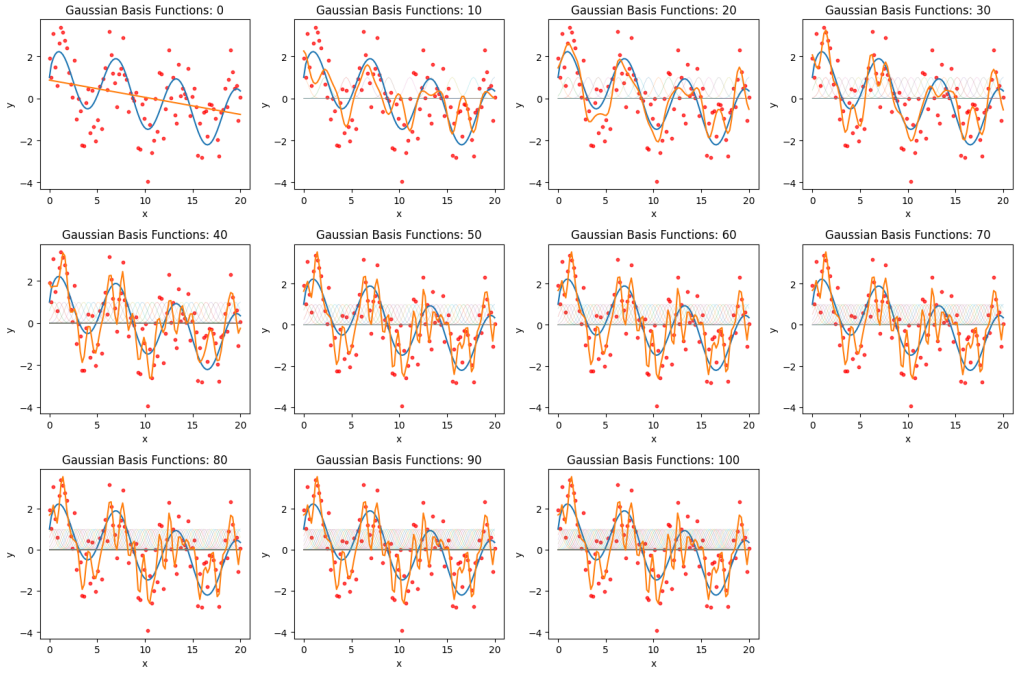


Figure 5: Comparison of the fitted models across different numbers of bases, $\sigma = 0.5$.

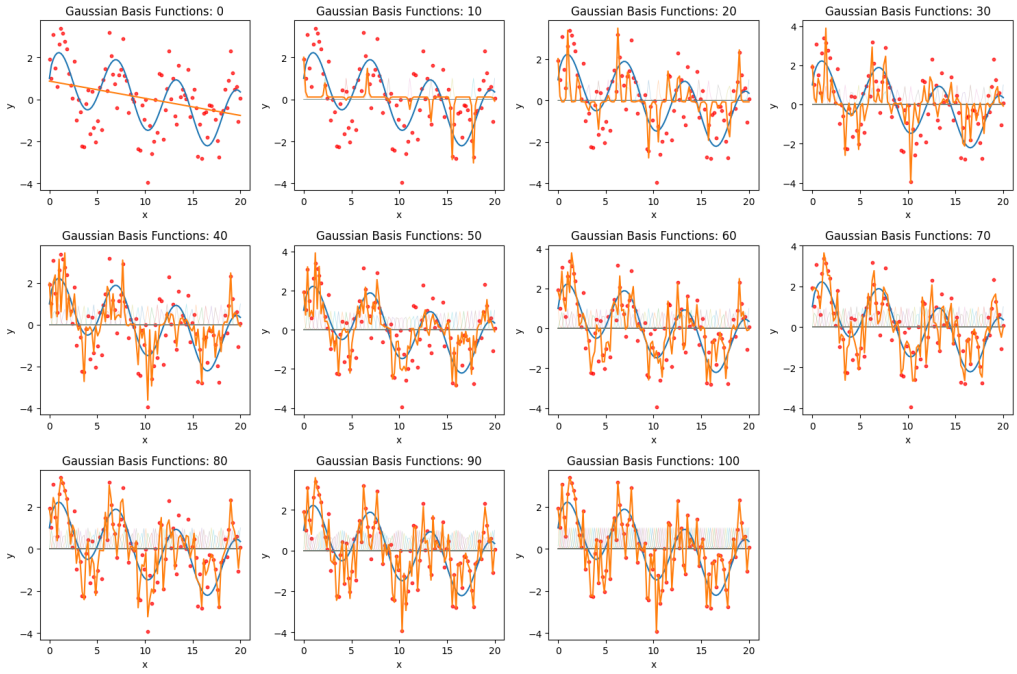


Figure 6: Comparison of the fitted models across different numbers of bases, $\sigma = 0.1$.

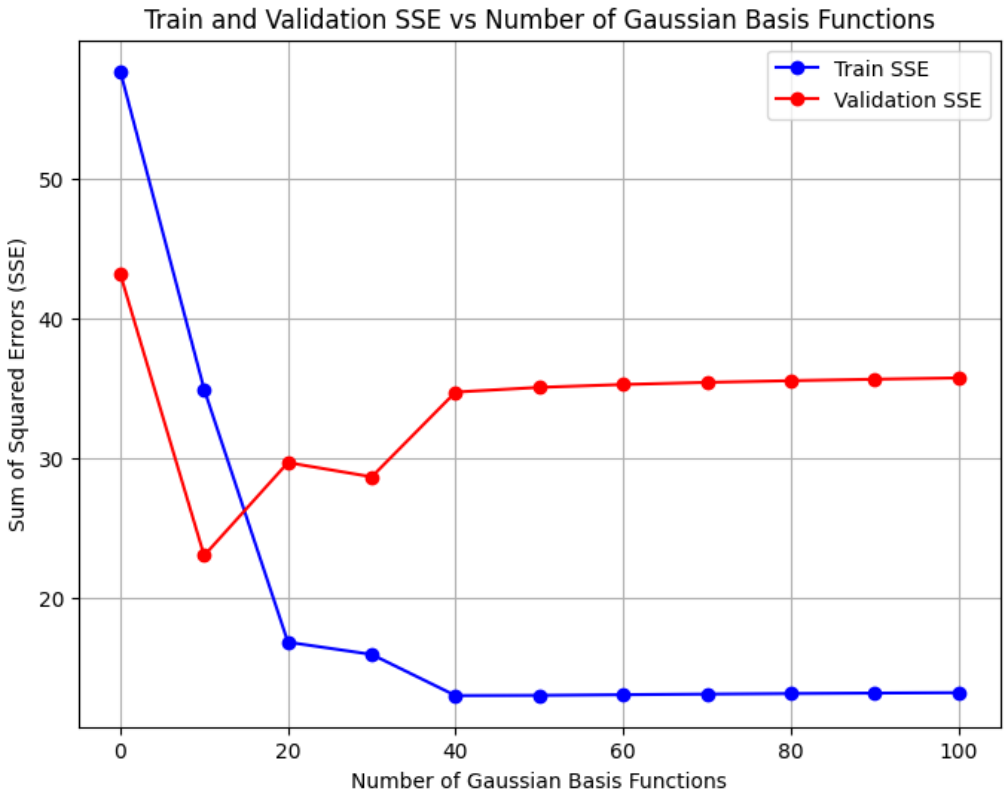
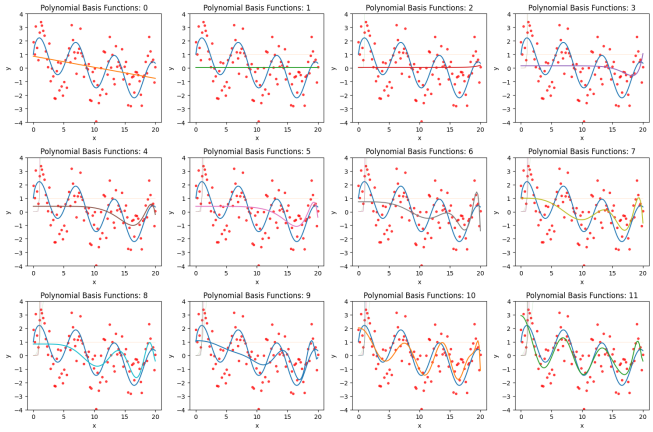


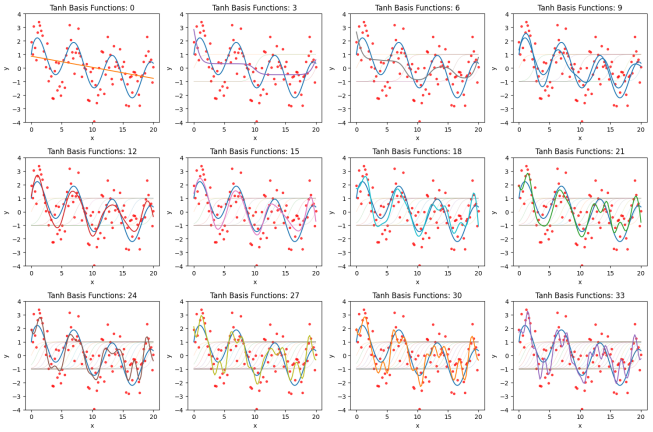
Figure 7: Validation and training set SSE as a function of the number of bases used.

Number of Bases	Train SSE	Validation SSE
0	57.6831	43.1723
10	34.9343	23.0279
20	16.8284	29.6798
30	15.9494	28.6611
40	13.0038	34.7276
50	13.0221	35.0661
60	13.0685	35.2707
70	13.1117	35.4215
80	13.1500	35.5434
90	13.1834	35.6478
100	13.2125	35.7409

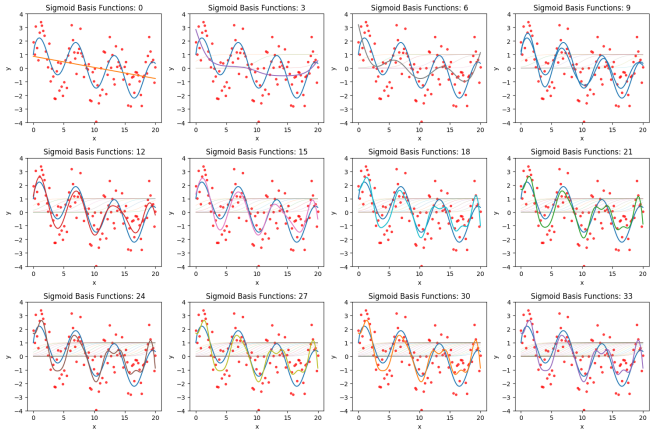
Table 1: Train and Validation SSEs for Models with Varying Numbers of Gaussian Basis Functions, with $\sigma = 1$ fixed. The values are plotted in Figure 7.



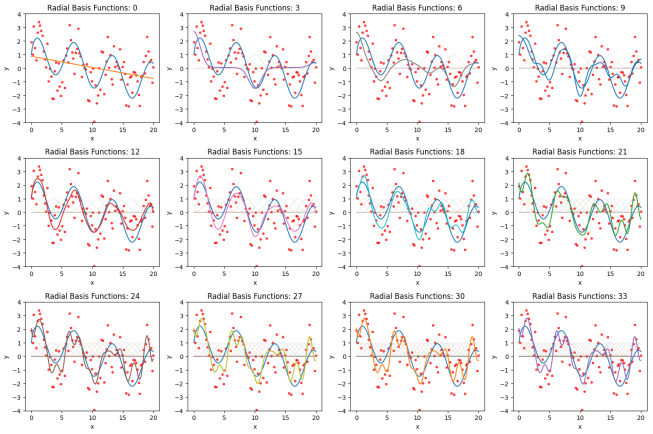
(a) Polynomial Basis Functions



(b) tanh Basis Functions



(c) Sigmoid Basis Functions



(d) Radial Basis Functions

Figure 8: Various Basis Functions for Linear Regression

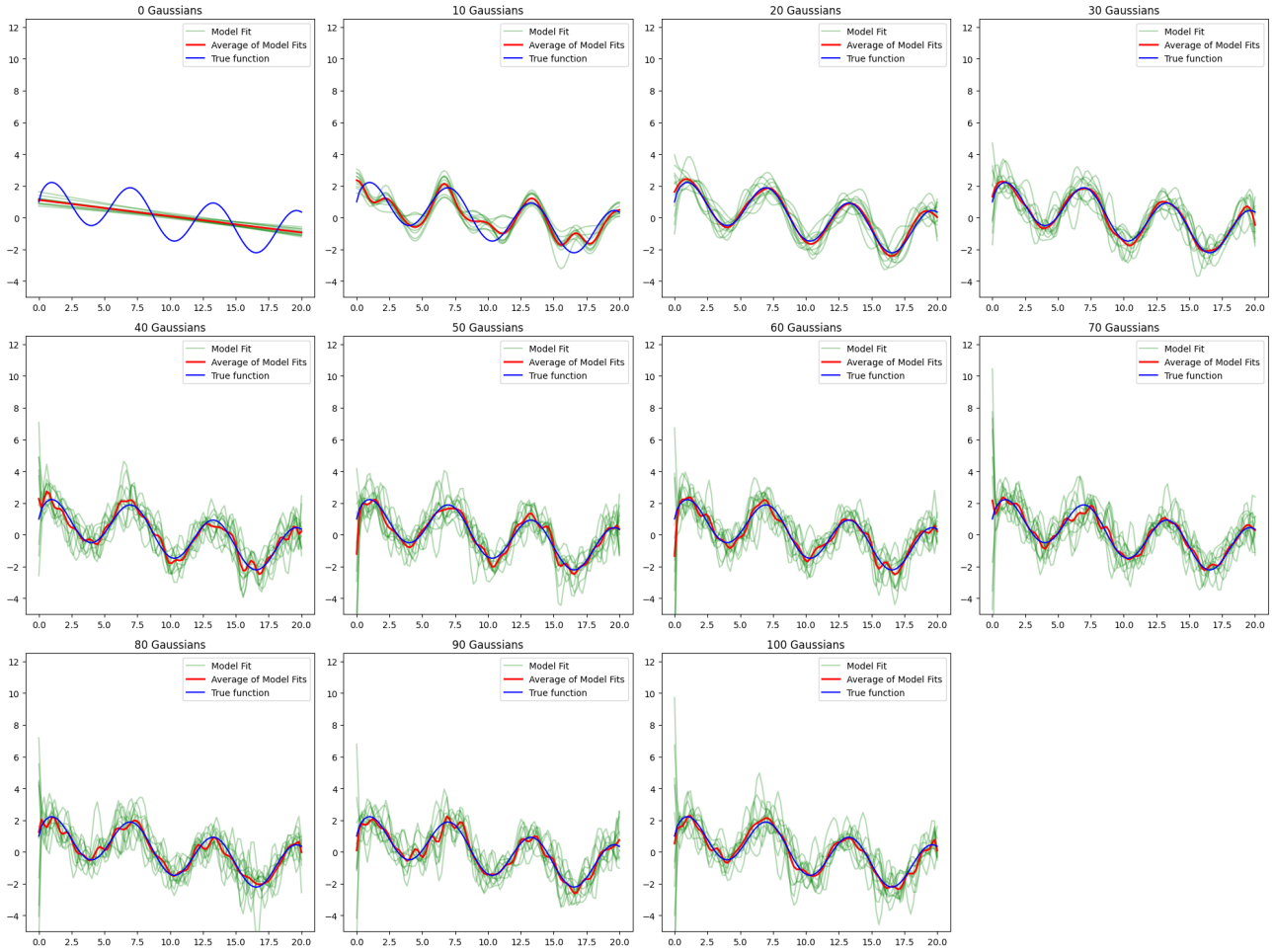


Figure 9: Model fits for varying numbers of Gaussian basis functions. Note that all the axes are uniform across the plots to enable clear visual comparison.

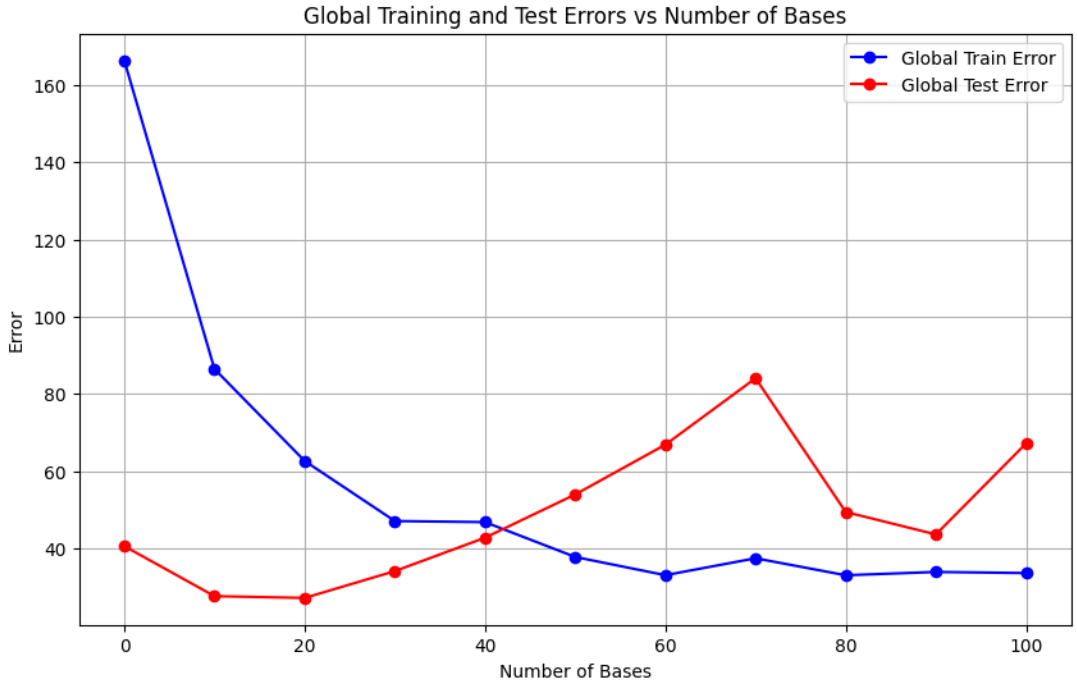


Figure 10: Global Training and Test Errors vs Number of Bases. The blue curve represents the training error, which decreases as model complexity increases, while the red curve represents the test error, which decreases initially but increases after a certain number of bases due to overfitting.

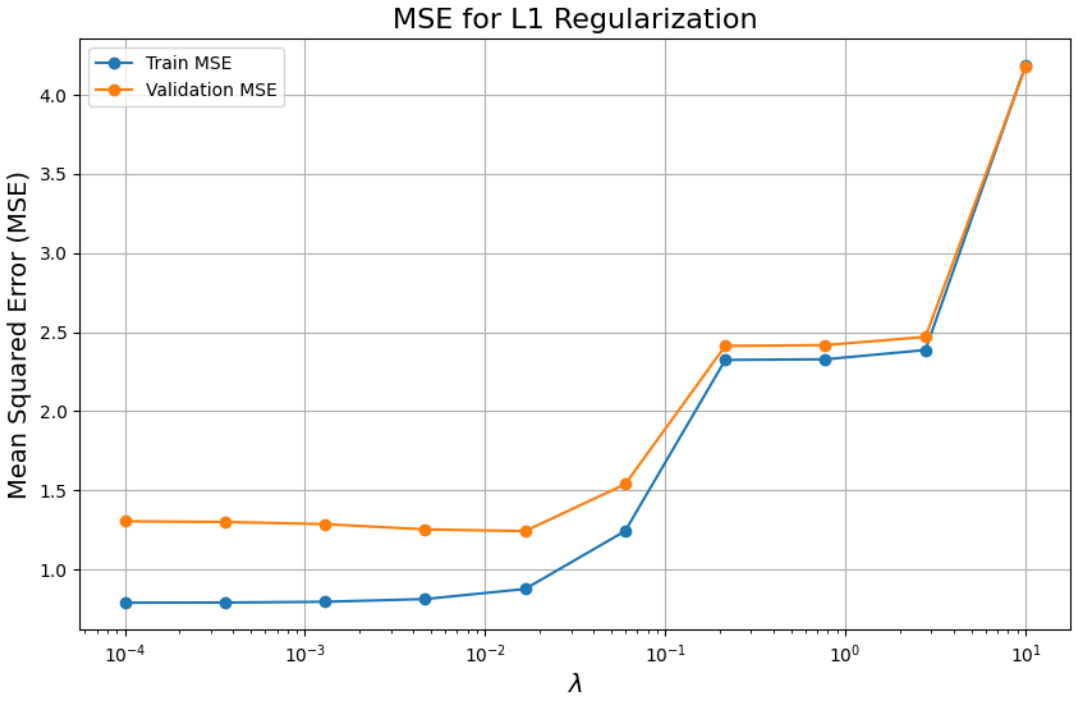


Figure 11: MSE for L1

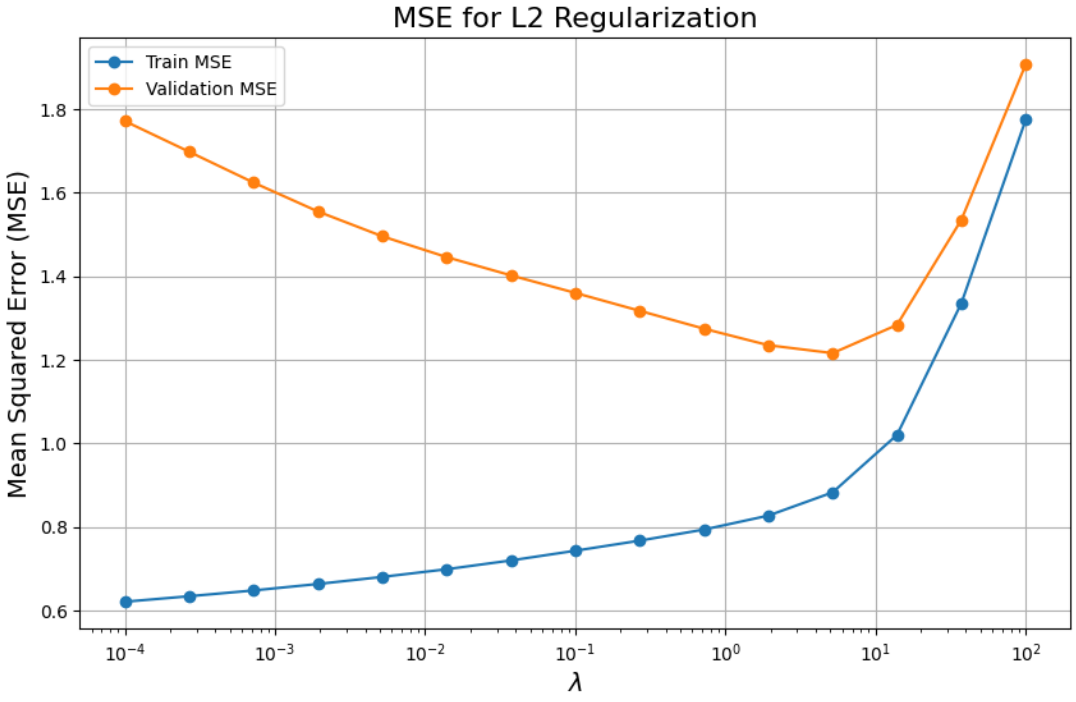


Figure 12: MSE for L2

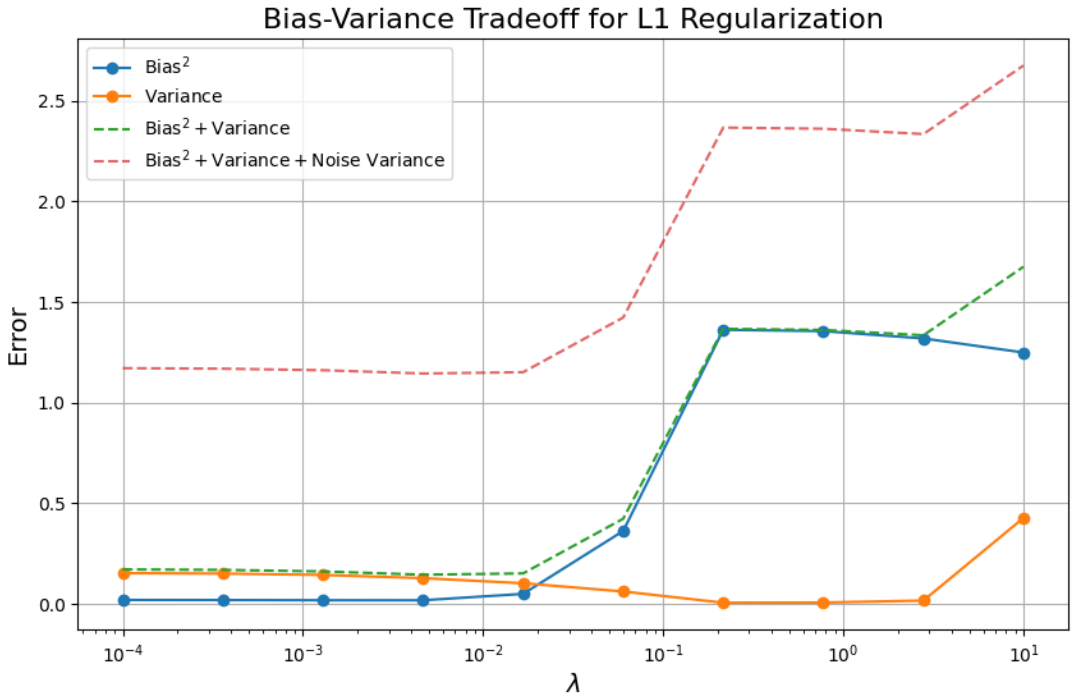


Figure 13: Bias-Variance Decomposition for L1

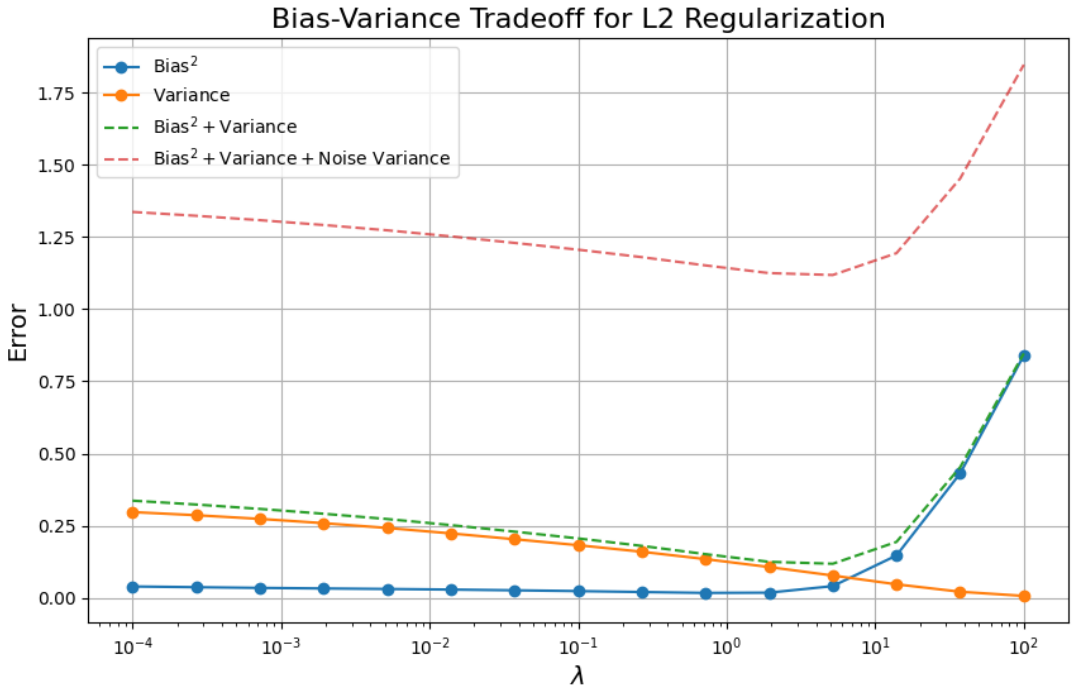


Figure 14: Bias-Variance Decomposition for L2

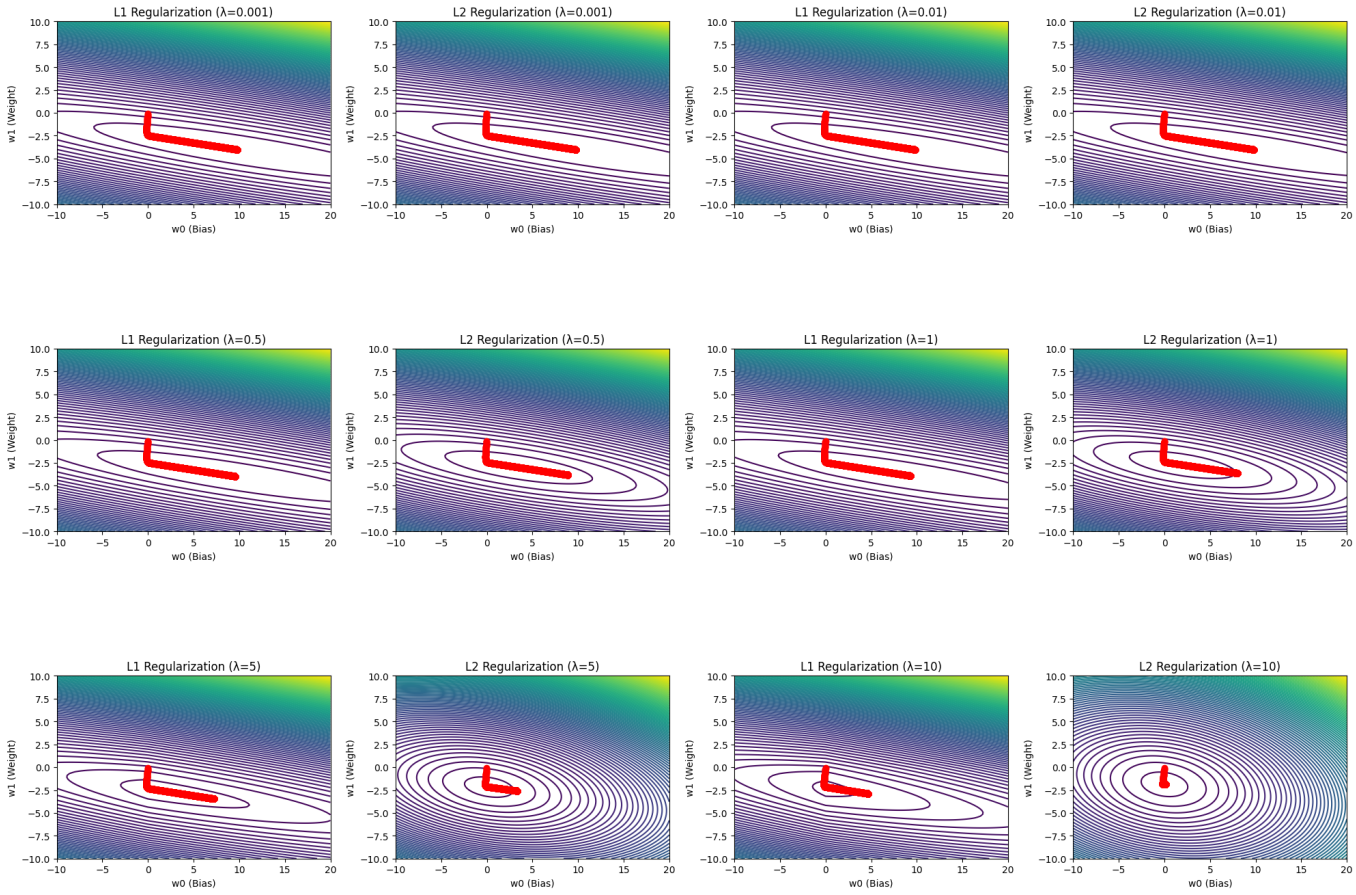


Figure 15: Comparison of L1 and L2 regularization across different values of λ.