Assignment 2 Write Up: Regularization and Model Evaluation

Students Rob Li, Ian Sonoda McFarland, Finnley Howald
Professors Reihaneh Rabbany, Isabeau Prémont-Schwarz

1 Abstract

In this assignment, we implemented a linear regression on fabricated data and examined the effects of non-linear basis functions on non-linear data, overfitting, and regularization. We demonstrated the change that comes with a varying number of basis functions, highlighting the bias-variance trade-off and how overfitting arises with increasing model complexity. Furthermore, We applied L1 and L2 regularization techniques, analyzed their effects on sparsity and weight penalization, and used 10-fold cross-validation to select optimal regularization strengths. Our findings show that model selection and regularization are essential to balance bias and variance and to ensure generalizability of the model.

2 Results

2.1 Task 1: Linear Regression with Non-Linear Basis Functions

The goal of this task was to generate 100 data points sampled from the non-linear function y(x) = sin(sqrt(x)) + cos(x) + epsilon and fit a linear regression model using Gaussian basis functions of the form $\phi(x, \mu, \sigma) = \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right)$.

2.1.1 Varying the Number of Basis Functions

Given a value for D, we equally spaced out D Gaussian bases over [0,20]. When varying the number of basis functions D, the correlation between the number of bases and model complexity becomes evident. We observe that, as the number of basis functions increases, the model becomes more complex and overfits the training data as seen in the figures below. We plotted this for D = 0, 10. 20. . . . , 100.

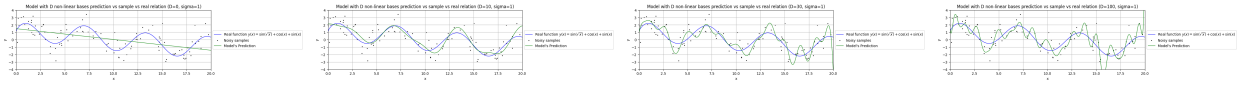
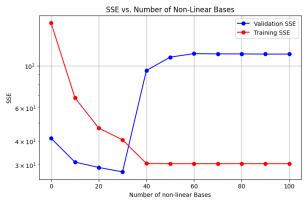


Figure 1: Fit on training data based on non-linear bases

2.1.2 Model Selection

For each model with D Gaussian basis functions, we calculated the sum of squared errors (SSE) on both the training and validation sets (figures below).



	Non-Linear Bases (D)	Training SSE	Validation SSE
0	0	182.965834	25.829549
1	10	97.098997	9.991391
2	20	78.184331	14.103161
3	30	60.219777	15.142119
4	40	51.382194	8.676281
5	50	48.840544	11.942828
6	60	48.982614	11.405799
7	70	49.049339	11.223329
8	80	49.085614	11.100729
9	90	49.105958	11.033031
10	100	49.114210	11.019827

Figure 2: SSE Table, and SSE vs. Number of Non-Linear Bases

To select the optimal model the rule of thumb is to select the simplest model within one standard deviation of the model with the lowest validation SSE. As seen in the graphs above, the lowest validation SSE is when there are 30 Gaussian bases. The simplest model one standard deviation within that point is when there are 30 Gaussian bases. Thus, the optimal model is when there are 30 Gaussian bases. As seen in the figure, as the number of Gaussian bases increases, the validation SSE increases, but the training SSE decreases. This suggests that as the number of Gaussian basis functions increases, the model overfits the training data but becomes less generalized for unseen data, hence the lower validation SSE and higher training SSE.

On the other hand, for lower numbers of Gaussian basis functions, we observed that the training SSE is higher. This indicates that as the number of basis functions decreases, the model underfits the training data.

2.2 Task 2: Bias-Variance Trade-off with Multiple Fits

In this task, we repeated the process from Task 1, 10 times to observe how bias and variance are affected by the number of Gaussian bases.

2.2.1 Explaining Bias and Variance Based on the Plots

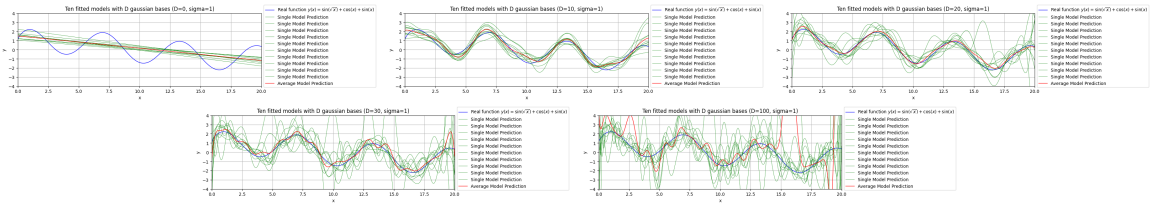


Figure 3: Ten fitted models with different number of Gaussian bases (D = 0, 10, 20, 30, 100)

Non-Linear Bases (D)	Training SSE	Validation SSE
0	168.677651	34.982049
10	86.236102	23.289356
20	68.449256	31.523504
30	57.999724	51.393447
40	50.178429	4398.829721
50	49.797013	4014.831960
60	49.537544	4844.594186
70	49.469503	4911.780227
80	49.460937	4971.716836
90	49.456849	4981.401283
100	49.454582	4985.196291

Figure 4: SSE Table, and SSE vs. Number of Non-Linear Bases

Plotting the 10 linear regression models with the different numbers of Gaussian bases 10 times and the training and test SSE values, we observed visually and quantititaively the trend between the number of Gaussian bases, bias, and varaince.

When the number of Gaussian bases was 0, visually we observed, that the curves had a low average difference and they did not fit the true distribution of the data accurately. Thus, the variance was low and the bias was high, and so the model underfits the training data (see figure 3).

The SSE table and figure back this claim further as we see there is a small but non-optimal training and variance SSE, suggesting that there is a better fit for the training data with a more complex model.

When the number of Gaussian bases was 10 and 20, we observed visually a trade-off between the variance and bias. The average difference between the curves increased, while the difference between the models and the true fit decreased yet still appeared to fit it well, this indicates that the bias and variance seem to be at a nice balance at this range (see figure 3). The SSE table and figure, also show that these models have the smallest validation error and low training SSE, further suggesting that the model is well fit and generalizes unseen data optimally.

Lastly, when the number of Gaussian bases was 30 or greater, the models overfit the training data since, visually, we see that the average difference between the curves is much higher. Quantitatively, we observed that these models have a very high validation SSE, suggesting that they are overfitting the training data and do not generalize unseen data well (figure 4).

2.3 Task 3: Regularization with Cross-Validation

Penalization Parameter	Training SSE	Validation SSE
0.000100	9.342586	3.179833
0.000228	9.356032	3.178179
0.000518	9.387532	3.174671
0.001179	9.463741	3.168013
0.002683	9.658984	3.157072
0.006105	10.191993	3.143373
0.013895	11.681900	3.163112
0.031623	15.816197	3.374281
0.071969	25.601720	4.020479
0.163789	39.040290	4.772550
0.372759	43.602230	4.877917
0.848343	43.779647	4.866210
1.930698	43.776320	4.865855
4.393971	43.775219	4.867043
10.000000	43.792668	4.865561

Penalization Parameter	Training SSE	Validation SSE
0.100000	5.718097	4.776391
0.193070	6.245073	4.160859
0.372759	6.845277	3.733134
0.719686	7.586285	3.427766
1.389495	8.607941	3.216603
2.682696	10.193934	3.106192
5.179475	12.775735	3.125986
10.000000	16.718527	3.295289
19.306977	21.942514	3.587831
37.275937	27.740082	3.932378
71.968567	33.088367	4.249989
138.949549	37.242843	4.492850
268.269580	40.041729	4.654039
517.947468	41.740409	4.750868
1000.000000	42.703260	4.805418

Figure 5: L1 and L2 errors summary

For this task, we applied L1 and L2 regularization to the linear regression model and used 10-fold cross-validation to determine the optimal regularization strength. L1 Regularization was implemented using gradient descent, and L2 regularization was implemented using the analytical solution.

2.3.1 Selection of the optimal lambda for both L1 and L2 regularization

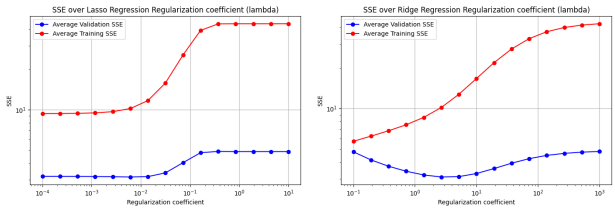


Figure 6: SSE vs Regularization coefficient, lasso and ridge regression

Based on our 10-fold cross validation plot for average train and validation error, we found the best hyperparameter values to for each regularization implementation. For L1 Lasso regularization, we found that 0.006105 gave us the lowest average validation error, after testing hyperparameters from 0.0001 to 10. For L2 Ridge regularization, we found that 2.682696 gave us the lowest average validation error, after testing hyperparameter values from 0.1 to 1000.

2.3.2 Analysis of the Results

For higher values of lambda, we observed smaller increases in error, suggesting that beyond a certain regularization parameter, the lost function does not result in significant changes in final prediction all other parameters being fixed. The next task in this section was give the bias-variance breakdown of the generalization error. We were also able to do this under the 10-fold logic. We will detail how we did this under section 3: Originality.

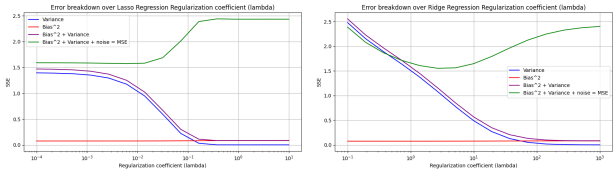


Figure 7: Error breakdown for lasso and ridge regression

Looking at our error breakdown for both L1 and L2, we saw increasing the regularizaiton hyperparameter significantly decreased the variance which we expect as the more we punish the weights, the less the model should overfit. Beyond a certain point, the overall error increased which we also expected. However, this increase is not driven by bias^2 as we might expect. Rather, this increase was driven by more "noise". Bias does increase slightly, but we do not see the full bias-variance trade-off that we expected. It is likely that our model is already elaborate enough to model a originally simple real function to see a significant bias component of our error.
We also had an issue where at low lambda values for the L2 model, our variance would be larger than the general error which shouldn't happen in theory. Our best explanation for this is the accumulation of numerical discrepancies for very small lambda values.

2.4 Task 4: Effect of L1 and L2 Regularization on Loss

Plots of effects of L1 and L2 Regularization on Loss, and the gradient descent path:

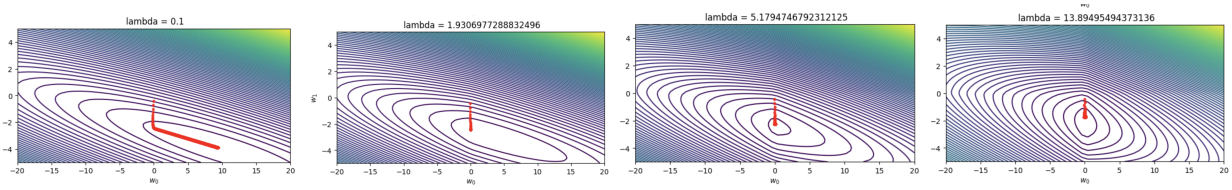


Figure 8: Effects of L1 Regularization on loss and gradient descent paths

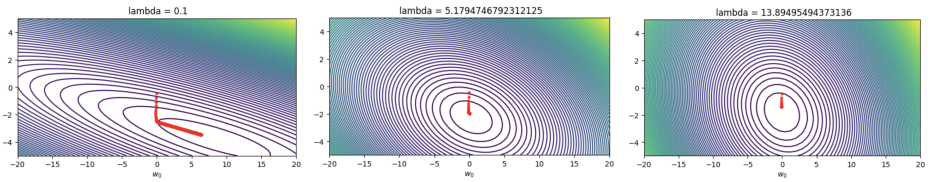


Figure 9: Effects of L2 Regularization on loss and gradient descent paths

2.4.1 How L1 regularization encourages sparsity (setting some weights to zero).

In our L1 regularization plots, it is clear that this regularization strategy encourages sparsity. As λ increases, and the loss plots become sharper, the loss function becomes flatter, and penalizes non-zero weights in a more intense fashion. At a lambda value of 0.1, our weights end up far away from (0,0), at a value of (10, -3.7). Compare this to a regularization strength of 2, where both w0 is already completely set to 0, and the optimal weight value depends entirely on w1. This is a stark contrast from the lower regularization strength. As such, it is clear that L1 regularization encourages sparsity.

2.4.2 How L2 regularization penalizes large weights but does not promote sparsity as strongly as L

In the plots for L2 Regularization, the effects of an increasing lambda in penalizing large weights is very apparent. At a lambda value of 0.1, our optimal weight values at the end of gradient descent are around (6, -3.6). Already, with this regularization strength, we see how L2 Regularization penalizes large weights. Compared to L1 regularization with a λ value of 0.1, the weights are already smaller. This is magnified in the following graphs with a lambda of 5.18 and 13.9, where the values of w0 and w1 creep closer and closer to 0. However, as compared to L1 regularization, our values of w0 and w1 never actually reach exactly zero, as happened for w0 in L1 regularization.

2.4.3 How different values of λ affect the optimization paths and loss landscapes

It is clear that as we increase the value of λ, that the loss landscapes of both L1 and L2 loss become closer and closer to just the L1 and L2 losses. For example, with a lambda value of 0.1, the loss landscape its almost exactly the loss landscape of our original function and synthetic data. However, as we increase the value of this lambda, the landscape changes, as we can see with a lambda value of 50 or 100, where the entire landscape is basically just the landscape of L1 or L2 loss. As for optimization paths, firstly, we see that the optimal value for gradient descent inches closer and closer to (0,0), as the regularization takes it effect. In bother cases, regularization lowers the value of the weights, and in the case of L1 regularization, it also promotes sparsity by setting weights to 0 (in our case, w0). It also takes a lower amount of iterations to converge to our optimal value. The path that the optimizer needs to take in order to get to our optimal value also becomes more linear, as it does not need to move to the right as the value of λ increases.

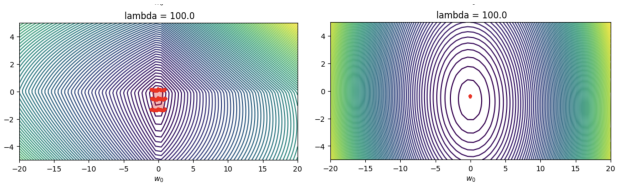


Figure 10: L1 and L2 regularization at a regularization strength of 100

There is a caveat, however, as when our regularization strength reaches 100, our models are completely dominated by the regularization component. In L1 regularization, this causes our gradient descent to diverge, and for L2 regularization, this causes the model to not have any change.

3 Originality/Creativity

Our creative input in this project was finding the Bias-Variance-Error Decomposition under the l-fold logic. For each hyperparameter value, we train upon K different samples of size N, for each sample we also train $l = 10$ additional models for each fold. It is worth noting that under l-fold cross validation, each sample point in the set is in the validation set once. So for each of the K samples and for each of the N sample points in that sample, we have our sample target $y(x)$, one model prediction $\hat{y}(x)$, and the value of x under the underlying function $y_{\text{real}}(x)$.
Lets fix some hyperparameter value. For any given sample k, its model $\hat{y_k}$, and for any specific sample point x_n, the

generalization breakdown is given by:

$$\text{MSE} = \text{Variance} + \text{Bias}^2 + \text{Noise}$$

$$\mathbb{E}\left[\hat{y_k}(x_n) - y_{\text{real}}(x_n)\right] = \mathbb{E}\left[\left[\hat{y_k}(x_n) - \mathbb{E}_D[\hat{y_k}(x)]\right]^2\right] + \mathbb{E}\left[y_{\text{real}}(x_n) - \mathbb{E}_D[\hat{y_k}(x_n)]\right]^2 + \mathbb{E}[\text{noise}]$$

For any fixed x, we define the average model's prediction:

$$\bar{y}(x_n) = \mathbb{E}_D[\hat{y_k}(x_n)] = \frac{1}{K}\sum_{k=1}^{K}\hat{y_k}(x_n)$$

This was indeed how we calculated $\bar{y}(x_n)$ for each sample point.
Now we can write:

$$\mathbb{E}\left[\hat{y_k}(x_n) - y_{\text{real}}(x_n)\right] = \mathbb{E}\left[\left[\hat{y_k}(x_n) - \bar{y}(x_n)\right]^2\right] + \mathbb{E}\left[y_{\text{real}}(x_n) - \bar{y}(x_n)\right]^2 + \mathbb{E}[\text{noise}]$$

We now we realize the random variables are $\hat{y_k}$ which varies over the K samples, and x_n which has N values. We can now see how we can find expected values in the breakdown.

$$\frac{1}{N}\sum_{n=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\left[\hat{y_k}(x_n) - y_{\text{real}}(x_n)\right] = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\left[\left[\hat{y_k}(x_n) - \bar{y}(x_n)\right]^2\right] + \frac{1}{N}\sum_{n=1}^{N}\left[y_{\text{real}}(x_n) - \bar{y}(x_n)\right]^2 + \mathbb{E}[\text{noise}]$$

The above equation gives how we handled the error breakdown for each hyperparameter value. In our case, $N = 20$, and $K = 50$.

4 Discussion and Conclusion

Throughout this assignment, we explored the intricacies of model complexity, overfitting, and the importance of regularization in linear regression.

In Task 1, we observed how increasing the number of Gaussian basis functions allowed the model to capture more complex patterns in the data, at the expense of overfitting. This is evidenced by the growing discrepancy between training and validation SSEs at different numbers of bases. The results showed that a moderate number of basis functions (around 30) was a good balance between bias and variance as it minimized the validation error without overfitting the training data.

In Task 2, we analyzed models fitted on multiple data samples and observed the bias-variance trade-off. Models with few basis functions exhibited high bias and low variance, while more complex models showed higher variance and a tendency to overfit. We found that the optimal model was found to be in between these extremes, where bias and variance were reasonably balanced.

Task 3 introduced L1 (lasso) and L2 (ridge) regularization techniques, which penalized large coefficients and controlled the complexity of the model. Using cross-validation, we determined the optimal regularization strengths. We found that L1 promoted sparsity by setting some weights to zero and L2 shrunk all weights proportionally without promoting sparsity. This was essential in preventing overfitting while ensuring good generalization to unseen data.

Finally, Task 4 provided a deeper understanding of how different regularization strengths influence optimization paths and loss landscapes. Both L1 and L2 regularization encouraged more stable models with smaller coefficients. However, at very high regularization strengths, the models became overly simplified and dominated by the regularization terms rather than the data itself.

In conclusion, this assignment showed the role of model selection and regularization in machine learning. We found that using regularization techniques such as lasso and ridge regression helped balance the complexity while ensuring models are neither too simplistic nor too complex, promoting generalizability. In closing, the bias-variance trade-off is a fundamental concept that must be carefully managed, and regularization offers a powerful tool to navigate this balance effectively.

5 Statement of Contributions

Rob Li: Tasks 1-3, Overleaf
Ian Sonoda McFarland: Task 3-4, Overleaf
Finnley Howald: Task 1, 4, Overleaf