Bias-Variance Tradeoff Analysis in various Hypothesis Classes (BITS F464 Project)

Pranav Siby, Siddharth Chaitra Vivek, Amogh Balepur, Ashit Jain
(2022A7PS0356G, 2022A7PS0569G, 2022A7PS1228G, 2022A7PS0606G)

April 15, 2024

Abstract

The Bias-Variance Tradeoff is a fundamental concept in machine learning that governs the performance of predictive models. In this project report, we examine the effect of varying the complexity of hypothesis classes on model performance. We inspect three different classes of models: Decision Trees, Support Vector Machines with the kernel method, and Multilayer Perceptron neural networks.

Problem Statement

1. Generate a synthetic dataset using a function from the hypothesis class $\mathcal{H}_0$.

2. Then, manipulate the complexity by designing different hypothesis classes such that

$$H_{-n} \subseteq H_{-n+1} \subseteq \ldots \subseteq H_0 \subseteq H_1 \subseteq \ldots \subseteq H_n.$$

3. Examine the bias-variance trade-off curve look for various models and draw conclusions.

The challenge of balancing model complexity with generalization performance is a difficult one. The bias-variance trade-off encapsulates this challenge, representing the delicate equilibrium between underfitting and overfitting.
The aim of this project is to explore the bias-variance trade-off across different hypothesis classes in machine learning. By generating synthetic datasets and manipulating the complexity of various hypothesis classes, we aim to analyze how bias and variance change as the complexity of the models varies. Specifically, we will examine the behavior of decision trees, neural networks and kernel methods.
This research seeks to determine the optimal level of model complexity that minimizes bias and variance while maximizing predictive performance. By systematically varying the complexity of models across different hypothesis classes, we aim to identify the point at which the bias-variance trade-off is optimized.
The bias-variance trade-off is a fundamental concept in machine learning that helps us understand the trade-offs between underfitting (high bias) and overfitting (high variance) models. Bias refers to the error introduced by approximating a real-world problem with a simplified model, while variance refers to the model's sensitivity to fluctuations in the training data. A high-bias model may oversimplify the data, leading to poor predictive performance, while a high-variance model may capture noise in the training data and fail to generalize to unseen data.
Through examining the bias-variance tradeoff graph for different complexities for a particular hypothesis class,we can get where it is the most optimal, giving us the most suitable model for each hypothesis class.

Methodology

We examine the effects of the Bias-Variance Tradeoff in 3 main Hypothesis Classes: Decision Trees, Kernel Methods (Support vector Machines) and Neural networks. The task at hand is a classification problem where we are trying to categorize points into 2 or more classes based on input features. This is visualized in the form of boundaries drawn on a 2D graph (higher degree feature spaces also represented in 2 dimensions). In each subsection a synthetic dataset was generated by choosing our classification boundary prior, and then generating points to lie within these boundaries and assigning classes. Finally, noise was added to the labelled data. Then hypothesis classes of lower and higher complexities were generated and the respective models fit on these classes. The results are visualized and conclusions are drawn in the next

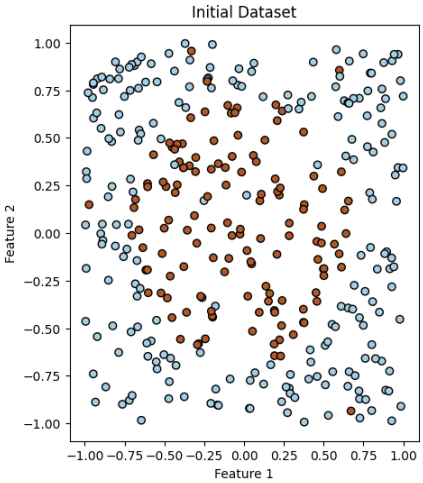

Figure 1: *Visualization of the dataset for Decision Tree experiments*

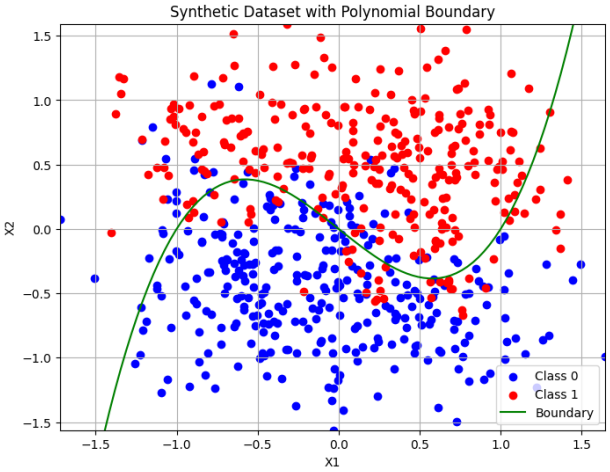

Figure 2: *Visualization of the dataset for kernel methods experiments*

section. This process is explained in more detail in each subsection.

Decision Trees

We employed a decision tree with a maximum depth of 4, denoted as hypothesis h_0, for our analysis. To create a synthetic dataset that aligns perfectly with this decision tree, we defined a rectangular region where all points inside the rectangle were assigned to class 1 and those outside to class 0. This ensured that the decision tree could classify the points accurately based on their coordinates.
To generate the synthetic dataset, we randomly created 500 points and assigned them labels based on their location relative to the rectangle. We introduced noise to the dataset to mimic real-world scenarios where data might not be perfectly separable.

For each decision tree with heights ranging from 1 to 9, we calculated the bias squared, variance, training error, and test error. Boundaries generated by the trees are also plotted to show the various cases of underfitting and overfitting

Kernel Methods

Kernel methods are a class of algorithms in machine learning primarily used for pattern analysis, classification, and regression tasks. They are particularly useful when dealing with nonlinear relationships in data. The basic idea behind kernel methods is to implicitly map input data into a higher-dimensional space where the data might be more easily separable, even if they are not linearly separable in the original space The fundamental goal here is to show the effect of increasing and decreasing complexity on the train and test error.

The dataset was generated by assuming the ground truth boundary to be the curve

$$y = x^3 - x$$

Then 600 data points were randomly distributed into classes on either side of the boundary with noise added giving us the dataset as shown in Figure 2. The noise is ϵ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ .This dataset is used for all experiments with Kernel Methods.
On this dataset models of different complexities were fit, where complexity/flexibility of the model can be modified in several ways:

1. Since the ground truth is a 3rd degree polynomial, we would expect a 3rd degree polynomial kernel to perform the best. Hence polynomials kernels of different degrees (both higher a lower value) will result in differing complexities of the hypothesis classes and are suitable for our problem statement
2. Another way to modify complexity is through regularization. By changing the regularization parameter we can control the extent to which models fit. Hence models were also fit with differing regularization parameter values. This is done for different types of Kernels are well, namely: Linear, Polynomial of degree 3, Gaussian and Sigmoid kernels
3. Lastly completely different Kernels were used,namely the following: Linear, Cosine, Laplacian, Radial basis, Polynomial and Sigmoid While this doesn't give us much insight into the bias-variance tradeoff it shows how much effectiveness of kernels varies (this is also very dependent on what the ground truth is)

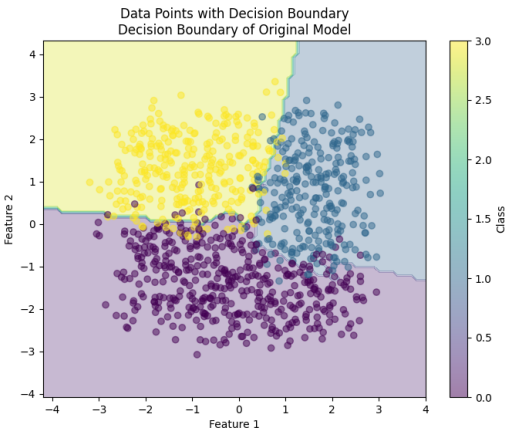

Figure 3: *Visualization of the ground truth decision boundary for the neural network experiment*

The results of the following experiments are discussed in the next section.

Neural Networks

To test the bias-variance trade-off, we generated a synthetic dataset by sampling 1000 feature vectors of size 2 from a uniform distribution. Then, the data was labelled with (at most) 4 classes according to a randomly initialized neural network $f(X)$. We chose a function $f(X)$ belonging to the hypothesis class $\mathcal{H}_0$ which is the set of neural networks with 2 hidden layers, input dimension = 2 and output dimension = 4. Now, to simulate real-world data, we let the response $Y = f(X) + \epsilon$ by adding a random Gaussian noise to each labelled data point, where ϵ is random noise, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. ($\sigma = 0.5$ here). We then split the labelled data into train and test portions. The ground truth decision boundary was then plotted. The dataset may be changed by editing the randomizer seeds in Neural_Networks.py.

Then, we created 9 neural networks having the same structure but different number of perceptrons in the hidden layers. Each network had 2 hidden layers, ranging from 4 to 36 perceptrons each. The networks were then trained on the same train data and tested on the same test data. We plot each result in a separate graph.

Experimental Results and Validation

Decision Trees

From the analysis of the graphs depicting the bias-variance trade-off for decision trees of varying heights, several key observations emerge. Firstly, it is

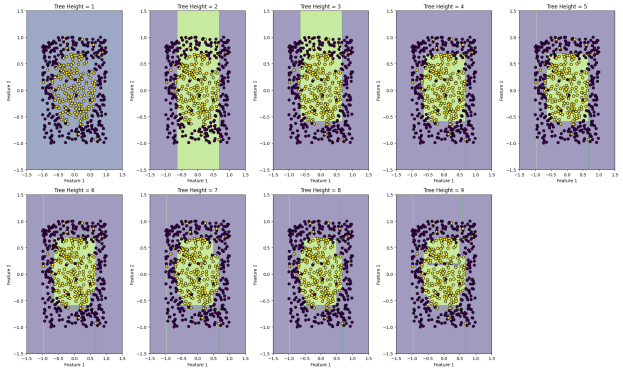

Figure 4: *Decision Tree fit for trees of varying complexity*

evident that the variance of the model increases as the depth of the tree increases. This phenomenon occurs because the model tends to overfit the training data, creating patterns that do not actually exist in the underlying data distribution. This overfitting leads to a higher variance in the predictions, as the model becomes more sensitive to small fluctuations in the training data.

For trees with a height of 9, the presence of small, intricate lines in the decision boundary plot indicates the model's attempt to partition the data to accommodate outliers or noise. This behavior is a characteristic of overfitting, where the model tries to fit the training data too closely, capturing noise along with the actual patterns in the data.

The trend observed in the train error further confirms the overfitting phenomenon. As the model complexity increases (i.e., higher tree heights), the train error consistently decreases. This reduction occurs because the model is able to fit the training data more closely, making more "correct" predictions according to the training data. However, the test error, which represents the model's performance on unseen data, initially decreases but then starts to increase after a certain point (around height = 4 in this case). This increase in test error indicates that the additional complexity beyond a certain point does not generalize well to new data, leading to poorer performance on unseen instances.

Lastly, the bias decreases initially as the model complexity increases, which is expected as the model becomes more capable of capturing the underlying patterns in the data. However, the bias seems to remain relatively constant after a certain point. This stabilization occurs because the model has almost perfectly identified all the points in the dataset, resulting in minimal bias in the predictions.
Overall, these observations highlight the delicate balance between bias and variance in machine

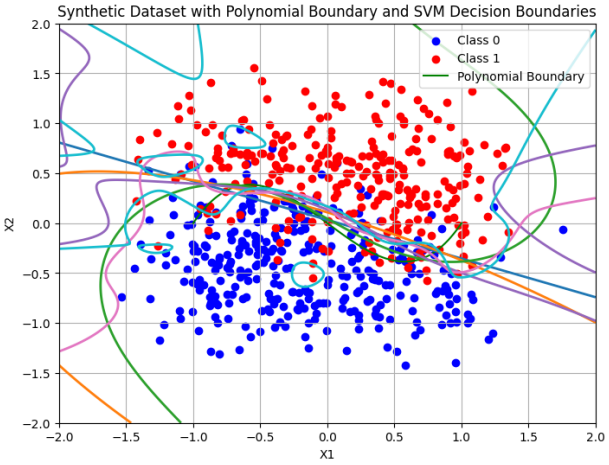

Figure 5: *SVM Boundaries for degrees 5,7,10*

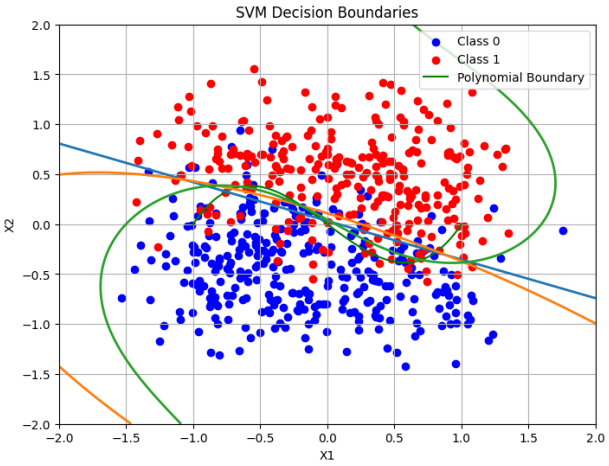

Figure 6: *SVM Boundaries for degrees 1,2,3*

learning models and underscore the importance of selecting an appropriate model complexity to achieve the best generalization performance.

Kernel Methods

As mentioned in the previous section 3 main experiments are performed:

The first experiment compared different degree polynomial kernels, as the generated boundary lines show, the boundary becomes very overfit as the complexity increases, the same is observed in the error vs complexity graph as the test error decreases and then increases as complexity increases. This is very reminiscent of the U shape test error curve which we encounter in theory, this is visible in figure 7.

The second experiment compared Test and Train Error for different values of regularization

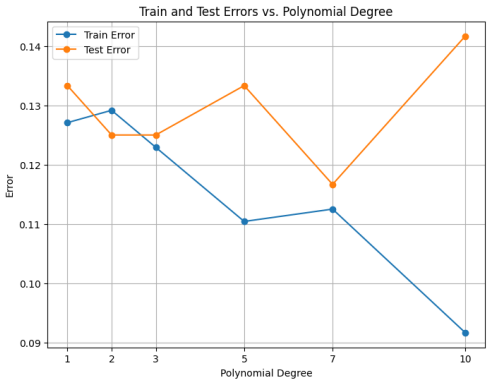

Figure 7: *Train and Test error curves for Kernel Methods*

parameters, here too we can observe as gradual decrease in Train error as model complexity increases and a somewhat stagnation of test error. While the results don't perfectly fit our expectation from theoretical study, they do still follow the general trend. This is shown in figure 12.

Finally, the third experiment doesn't have much significance apart from showing the variation of Train and Test error depending on the type of kernel used given a certain ground truth. The plots for Train and Test error versus the different methods is shown. This is shown in Figure 13

Neural Networks

After the training of each network, we plotted the decision boundaries of the trained network and the data. After each model was trained, we used the test data to measure how well each model did. We quantified this by plotting train and test error against model complexity (number of perceptrons in each hidden layer).

We had hoped to see an increase in test error with increasing complexity but with this formulation of the neural network, we found a general decrease in both train and test error with increasing complexity, even for many perceptrons per layer. This could be due to the simplicity of the dataset, or the shallowness of our hypothesis neural network.

Conclusion and Future Work

Decision Trees

We were able to clearly see the effects of too much variance on the fit of the model. Fig 4 shows how, upon adding more splits to the tree, small and narrow regions were created signifying overfitting. When

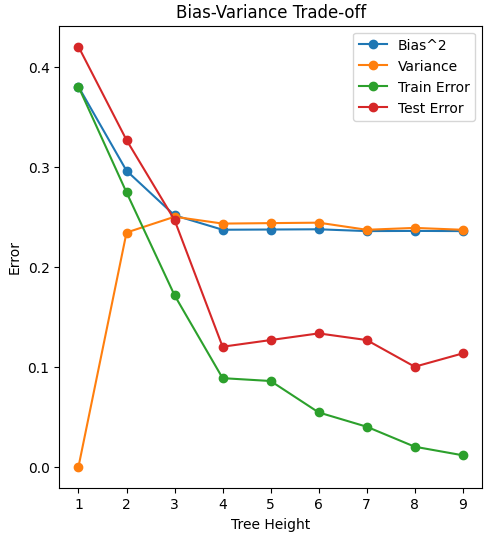

Figure 8: *Train and Test Errors for decision trees*

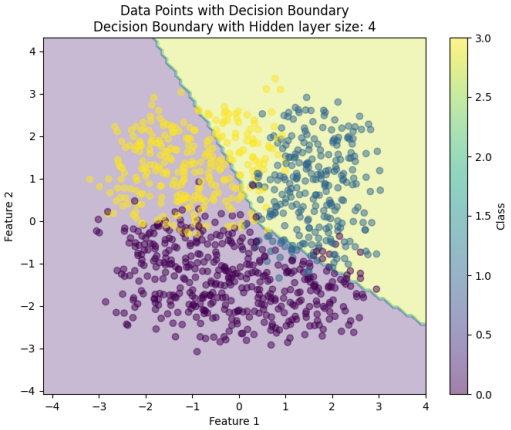

Figure 9: *Low complexity trained model dataset in the neural network experiment*

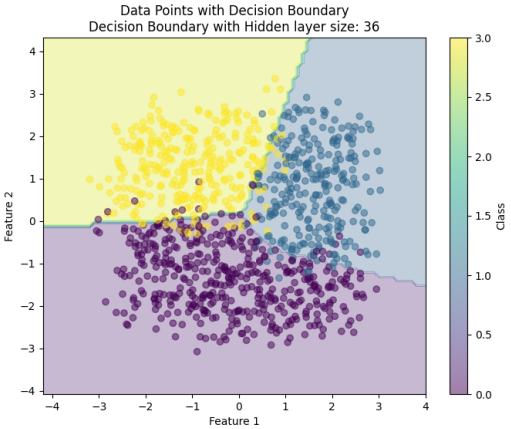

Figure 10: *High complexity trained model dataset in the neural network experiment*

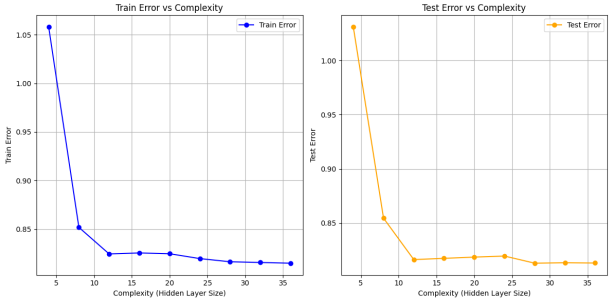

Figure 11: *Train and Test Error vs Complexity in the trained neural networks*

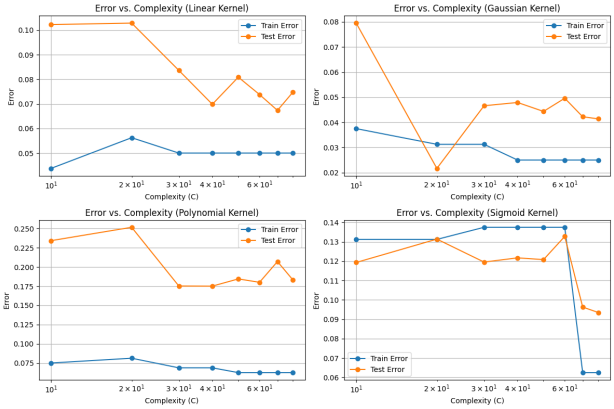

Figure 12: *Plots for varying regularization factors in Kernel Methods*

there were too few splits, few points were classified correctly. From the figures,

Our simulated data was quite simple and roughly bounded within a single rectangular region. We could further amplify the effects of overfitting for demonstration purposes by simulating data from a more intricate data set, in which case the under- and over-fitting will be even more evident. Adding more classes could also be helpful for demonstration.
One major conclusion which we can reach is that decision trees of 4 or 5 splits was most suitable for the synthesized dataset.

Kernel Methods

We clearly see from Figure 5 that the high-complexity kernels (high-degree polynomials) overfit wildly and Figure 6 illustrates how the lower complexity kernels fit the data decently well. In Figure 5, we see that the higher-degree polynomial kernels create illogical decision boundaries in areas outside the boundaries of the data points. Further, we can try this experiment for different ground truths, such as bounded ones.
One major conclusion which we can reach is that while comparing various degree polynomial kernels, 3rd or 4th degree polynomial kernel was most suitable for the synthesized dataset.

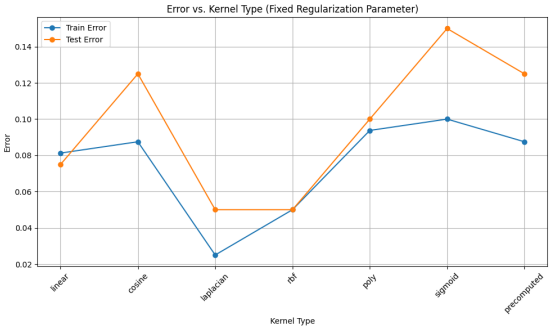

Figure 13: *Plots for different types of Kernels*

Neural Networks

From our experiment, we see that the neural networks with many neurons were able to fit the data better. (Figure 3, 7, 8) From Figure 9, we see that models with more than 12 neurons per hidden layer were able to match the ground truth decision boundary quite accurately, as made evident by the decision boundary in Figure 8 closely matching that in Figure 3. Our hypothesis neural network was too shallow to capture complex relationships and hence even after adding a large number of perceptrons per layer, the decision boundary was not overfit too much. Allowing a more diverse range of structures could better showcase high-variance results.
Similarly here too we can see that around 12-16 neurons per hidden layer is the ideal amount for the given dataset