Epistemic Applications of Occam's Razor in Machine Learning

Evan M. Van Jaarsveld, Raj S. Baveje, *Students, PHIL29300*

Abstract—Occam's Razor, in brief, philosophical context, states that the simplest solution is a better solution when all else is equal. The epistemic interpretation of this claim was tested in the field of machine learning using two PAC-learnable and contextually driven models, both of which were trained to predict New York Airbnb rental prices. An arbitrary complexity metric was defined for each algorithm, and this metric was then manipulated to test its impact on predictive accuracy. The experiment resulted in evidence disproving epistemic applications of Occam's Razor in machine learning, as the rule of said razor was violated by algorithm performance.

Index Terms—PAC Learning, Machine Learning, Occam's Razor, Complexity

* *This research was conducted at the behest of Dr. Bruce Rushing, a professor in the Philosophy Department of Purdue University*

* *Refer to the course roster for contact information*

1 Introduction

OCCAM'S Razor, despite its deceptive simplicity, has seemingly found a home in almost every field of inquiry imaginable. While beginning as a philosophical concept, its prescriptions have wide applicability and can even be sound life advice in certain circumstances. The philosophical razor is defined as a "problem-solving principle that recommends searching for explanations constructed with the smallest possible set of elements." [4]. Put another way, it advocates that if all else is equal between a set of solutions to a problem (in that they all solve the problem equally accurately), then the simplest solution in the set is preferable.

The razor has been welcomed into most STEM fields, including that of computer science and its vast array of subsets, of which Machine Learning (ML) is our primary focus. ML research in the modern day requires significant amounts of energy and time, making computational efficiency an important pillar of its progression. In the field of ML, complexity costs resources, so using Occam's Razor to minimize complexity while maintaining model accuracy may prove to be a sound way forward. However, while the benefits of employing Occam's Razor are possible, *proving* its effectiveness is another thing entirely.

Thus, an experiment was designed to test Occam's Razor's epistemic effectiveness in the field of ML. This required the definition of new rules and metrics, such as what constitutes "complexity" and "success" in terms of ML. Additionally, to provide a sound philosophical backing for these definitions, the logic undertaken to create the experiment will be extensively reviewed. This will take some time, as the necessary groundwork includes but is not limited to concepts of Probably Approximately Correct (PAC) learning algorithms, the Vapnik–Chervonenkis (VC) dimension of a hypothesis set, Multilayer Perceptron (MLP) Neaural Networks, and Regression Trees.

Through these understandings and the experimentation that arises from them, Occam's Razor will be proven to not be viable in the field of ML.

2 PAC Learning

PAC Learning, or Probably Approximately Correct learning, is the framework for considering questions dealing with the accuracy and efficiency of machine learning models. It bridges computational complexity with concepts like Occam's Razor and introduces tools such as generalization error and the Vapnik-Chervonenkis (VC) Dimension, which let us assess and influence model performance.

2.1 Relation to Philosophy

Probably Approximately Correct (PAC) learning is a framework for analyzing not just the accuracy of ML models, but their efficiency as well. This is done through the use of computational complexity, a concept first posited and developed for its pivotal role in the definition of the PAC framework [1]. It is this concept of complexity, and its near direct mapping to the Occam sense of complexity, which makes the razor possibly applicable to ML.

Computational Complexity Theory is a much broader subject than PAC learning, not just applying to ML models but all computer programs [7]. The theory demands that computer programs be classified according to their resource usage, and that those demanding more resources be labeled as more complex and/or difficult. In other words, "A problem is regarded as inherently difficult if its solution requires significant resources, whatever the algorithm used. The theory formalizes this intuition, by introducing mathematical models of computation to study these problems and quantifying their computational complexity" [7].

It seems apt to use this notion of "resources", the metric that determines computational complexity, to quantify the Occom-sense complexity of a program. Theoretically, this

* *All authors are students at Purdue University Main Campus, located in West Lafayette, Indiana, USA.*

would allow the analysis of multiple ML algorithms using Occam's Razor. However, there is no consensus on which resources constitute complexity [7], so the same problem of undefined metrics is encountered one step lower.

2.2 Probably and Approximately

Seeing as the vague notion of "resources" has proven untenable as a metric of Occam-sense complexity, the next viable solution lies in the calculations of 'Probably' and 'Approximately' in PAC learning, and their role in the calculation of PAC-sense complexity.

The 'Probably' portion of PAC learning is the simpler of the two, corresponding to the likelihood that a learning algorithm meets some set threshold of accuracy. [7] This threshold is determined by the 'Approximately' portion, which will be discussed in detail soon.

Keeping with the P in PAC, its significance is also the simpler to understand of the two; ML models should be consistent in their output, whatever that output may be. Even if the output is undesirable, consistent behavior is more conducive to troubleshooting than the alternative randomness.

Ideally, ML models would not just perform consistently, but also perform *well* consistently. This is where the 'Approximately' of PAC learning comes into play. Good performance is quantified by generalization error, with lower values indicating the model is more accurate in its predictions [7]. A precise mathematical definition of generalization error is beyond the scope of this paper, with the key takeaway being that generalization error is calculated on a per-instance basis [8].

Generalization error is a measure of accuracy, and as such, changes between each and every instance of a model. If model parameters are changed or if different input data is used, the generalization error will change as well. Even re-training the same model with the same parameters, just using a different data loader, will result in a different generalization error.

This understanding that generalization error changes on the slightest tweak to a model is critical. This analysis of PAC learning has been a means to an end, that end being to discover how Occam-sense complexity is represented in the field of ML. And while that end remains unknown, it has been shown that generalization error, a crucial component of PAC-sense complexity, changes wildly and unpredictably based on the smallest minutia of a model. By the transitive property of logic, this indicates that any quantification of Occam-sense complexity would be equally unpredictable. However, all is not lost, as general trends can still be spotted in the evaluation of generalization error

2.3 The Vapnik–Chervonenkis Dimension

Recall that it is the goal of any ML model to minimize its generalization error, so that it may more accurately predict whatever data it is trained on. While this precise value for generalization error is impossible to predict and can therefore only be discovered through post-training calculations, the PAC learning framework provides a way to predict general trends in the error [5].

The framework accomplishes this through the abstraction of a model's training process, depicting it not as the myriads of linear algebra that it is, but rather as the model selecting a function from a set presented to it. This function is called a hypothesis, denoted h, and it is plucked from a set called the Hypothesis Space, H. In simple terms, the VC dimension is a measure of how large H can get, and therefore how many hypotheses are available to the model. According to the PAC framework, the model's training is a process of selecting the function h with the lowest generalization error within H [5].

It is tautologically true that there exists a function h within H containing a locally minimized generalization error value, and fitting the definition of generalization error, this function h is the most accurate that a model could select. It then stands to reason that restricting H to a subset of hypotheses C could exclude the previously discovered minimum-error function, and conversely, expanding H could include a function with an even lower error than the one previously discovered. These two actions correspond to decreasing and increasing the VC dimension respectively [3].

This insight reveals that while generalization error for a model cannot be *precisely* predicted, it can be manipulated by using the inversely proportional relationship between generalization error and VC dimension.

By manipulating a model's VC dimension and by extension its generalization error, the 'Approximately' portion of PAC learning can be intentionally chnaged. This results in the PAC-sense of complexity itself changing, and due to the previously discussed mapping of complexities, it results in the direct manipulation of Occam-sense complexity.

In short, the VC dimension seems to serve as a good lever with which to manipulate Occam-sense complexity; Increasing VC increases complexity, while decreasing VC decreases complexity.

3 Regression Tree Experiment

3.1 Definition of Complexity

As discussed, a model's VC dimension can be used to manipulate its Occam-sense complexity. This notion is important as it allows us to quantify and adjust the complexity of a model in a controlled manner. The VC dimension, or Vapnik-Chervonekis dimension, is defined as a measure of the capacity of a statistical classification algorithm, where it represents the cardinality of the largest set of points that the algorithm can shatter [2]. In simpler terms, it can be thought of as how general or complex of a function the model can fit. Seeing as an increase in VC dimension equates to an increase in the available hypotheses h to a model, it stands to reason that in the context of a Regression Tree, tree depth would serve as one way to accomplish this. By increasing the depth of the tree, we are effectively increasing its VC dimension, hence increasing its complexity.

3.2 Experimental Setup

For our experiment to test our hypothesis, we utilized a dataset of New York Airbnb rental prices. The reason such a dataset was selected is due to its complexity and wide

variety of features that it contains-hence, making it a perfect candidate to test our models. We trained the Regression Tree model with a variety of different depths to observe the effect on both predictive accuracy and complexity. The tested depths varied from 2 to 32, with each specific depth corresponding to a distinct level of complexity.

The performance of the model was evaluated using two primary metrics: the coefficient of determination (R^2) and the Mean Squared Error (**MSE**). The R^2 value estimates the level of correlation between the target variable and the features; higher values indicate a better fit. On the other hand, the MSE measures the average squared difference between predicted and actual targets, where lower values indicate better performance.

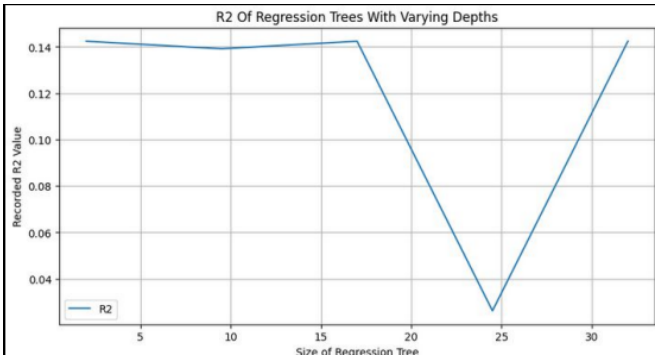
The experimental procedure involved creating a DecisionTreeRegressor from sklearn and running five regressions-each with a different level of depth. With every run of a regression, the values for R^2 and MSE were noted in order to evaluate the model.

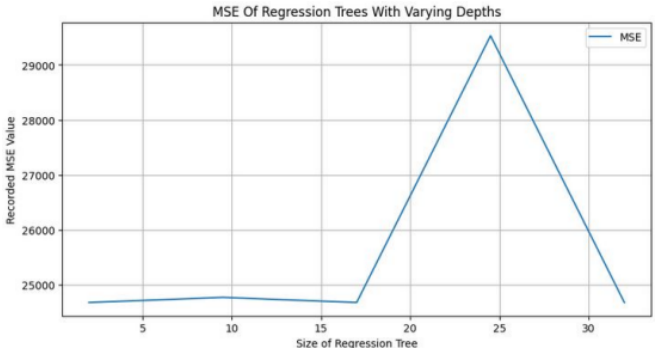
3.3 Results of Regression Trees

The results show that as the depth of the Regression Tree increased, the accuracy of the model first improved, remained constant for some time, and finally decreased with further increase in depth. This suggests that after a certain point, increased complexity does not improve performance, but, on the contrary, may degrade it due to overfitting. Overfitting occurs when a model learns the noise in the training data, rather than the actual signal, leading to poor generalization on new, unseen data [5].

Depth	R^2	MSE
2	0.14239199126	24677.2455107
4	0.13915859123	24770.284995
8	0.14239199126	24677.2455107
16	0.0262245239	29529.1021847
32	0.14239199126	24677.2455107

The table above shows the R^2 and MSE values for different depths of the Regression Tree. As we can see, the R^2 value does not consistently increase with depth, and the MSE values also show variability.





3.4 Discussion

The results show that there is no significant relationship between the VC dimension, as measured by treed depth, and model performance. This comes with the conjecture that Occam's Razor is not consistently applicable in the domain of machine learning because the lower complex models, measured by lower VC dimension, do not necessarily indicate an improved performance. The principle of Occam's Razor, which advocates for simplicity, does not hold true in this context since the relationship between the model complexity and performance is not straightforward [4].

For example, the Regression Tree with a depth of 2 had an R^2 value of approximately 0.14239 and an MSE of 24677.245. At an increased depth of 4, the R^2 value slightly decreased to 0.13915, and the MSE increased to 24770.284. This indicates that increasing the model complexity did not result in any improvement in performance for the model. Similarly, at a depth of 16, the value of R^2 decreased drastically to 0.02622, and the MSE increased to 29529.102, meaning such an increase in complexity implied a worse performance, due to overfitting. These specific values highlight the idea that there is no consistent or systematic improvement in performance for an increase in complexity, therefore supporting our claim that Occam's Razor is not applicable in every context.

Simpler models are, in general, much easier to interpret and tend to ask for less in terms of both computational power and data. However, as seen through this experiment, these simpler models do not always produce the highest levels of predictive accuracy in all situations. This is evidence for the need to balance between simplicity and complexity in machine learning models. Models that are too simple fail to capture the maximum number of important trends within the data, while models that are too complex will suffer from overfitting and generalize poorly when fit on new, unseen data.

4 Multilayer Perceptron (MLP) Experiment

4.1 Complexity

In the case of a Multilayer Perception, or ML, there is an important consideration of the number of parameters that exist in the network. Specifically, the more parameters there are, the higher the VC dimension and, theoretically, the greater a model's ability to learn complex patterns. The VC dimension in neural networks is a function of the architecture of a network, including the number of layers and neurons in each layer [2]. By changing the parameters, the complexity of the MLP can be controlled and its effect on model performance can be analyzed.

4.2 Experimental Setup

We trained MLP models with a different number of parameters through the variation in the number of layers and neurons in each layer. The dataset and evaluation metrics were the same as in the experiment with the Regression Trees. All MLPs were trained for 100 epochs, where the loss value at each 10th epoch was noted down as a significant value of their performance. The layer sizes were manipulated to come up with models of varying complexity. The performance evaluation for each model was made on a level ground using the same metrics: R-Squared (**R^2**) and Mean Squared Error (**MSE**).

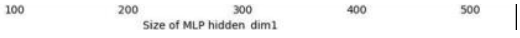
The experimental process included designing the 2-layer MLPRegression Model form the nn.module package and running five regressions for each setup. We modified the layer size specifically to create models with a different number of parameters. The number ranged from smaller configurations, where the first layer has 32 neurons and the second has 16 neurons, to larger configurations, where the first layer contained 512 neurons and the second had 256 neurons. For each run of regression analysis, we carefully noted the values of R^2 and MSE to evaluate and establish the performance of the model. With this preparation, we were able to use our algorithms to test our hypothesis behind the principle of Occam's Razor.

4.3 Results of MLPs

The accuracy of the MLP, like that of the Regression Tree, improved with increasing the number of parameters only up to a certain point. Beyond this point, the accuracy suffered from overfitting and decreased generalization performance. This indicates there may be an optimum level of complexity for the MLP, beyond which the model's performance deteriorates.

Dim1 Size	Dim2 Size	R^2	MSE
32	16	-0.144892	32943.703
64	32	0.1556605	24295.449
128	64	0.1740685	23765.770
256	128	0.1343021	24910.029
512	256	0.1505060	24443.768

The table above shows the R^2 and MSE for different configurations of MLP. As we can see, there is no continuous increase in the value of the R^2 metric while we increase the number of parameters. Furthermore, the MSE values also move away from the pattern, hinting that there is not a direct relationship between the number of parameters (complexity) and model performance.



4.4 Discussion

Further support for our claim comes from the results of our MLP experiment. There is no consistent correlation between the VC dimension (number of parameters) and model success. This further strengthens the argument against the applicability of Occam's Razor in machine learning; the principle that simpler models are inherently better does not hold true. On the other hand, the effectiveness of the machine learning models depends upon a number of factors, including the complexity of the data, model configuration, and methodology of the training process [1][3][6]. The findings indicate that there is an optimal level of complexity for the MLP, beyond which additional complexity leads to diminishing returns or even negative effects on performance.

We can analyze the data given to support our claim. We see that the MLP with 32 neurons in the first and 16 in the second layer has a R^2 value of –0.14489 and an MSE of 32943.703. Increasing the number of neurons to 64 and 32 improved the R^2 value to 0.15566 and reduced the MSE to 24295.449. However, further increasing the number of neurons to 265 and 128 resulted in an R^2 value of 0.13430 and an MSE to 24910.029, which is a decrease in the overall performance of our model. This pattern also continues with the largest configuration of 512 and 256 neurons. This data shows that there truly is a specific level of complexity that can be considered "most optimal." Beyond this level, the increase in parameters simply causes overfitting and reduces performance. This evidence strengthens our claim that

Occam's Razor is not a reliable rule of thumb in the field of machine learning since increased complexity does not always yield better results.

5 Conclusion

Our experiments with Regression Trees and Multi-Layer Perceptron (MLPs) demonstrate that there is not a strong correlation between VC dimension and model accuracy. In both experiments, we observed that there is an optimal level of complexity. Increasing the number of parameters improves model accuracy up to this certain point, but going beyond it can lead to overfitting and poorer overall performance. However, reaching this level of optimality requires increasing the complexity of a model to some extent, rather than committing to the simplest model.

This result goes against the common belief that models of lesser complexity are better at predictions-a notion supported by Occam's Razor. There are many more variables that affect model performance in machine learning, so model complexity alone cannot determine success.

While simpler models are generally easier to interpret and require fewer resources, they do not yield the highest predictive accuracy in every possible situation. More complex models, on the other hand, have the potential to discover more complicated patterns in the data, but are also more prone to overfitting and usually require more computational reasoning and resources.

Overall, this paper and study suggest that the relationship between the complexity of models and their performance is more complicated than the principle of Occam's Razor suggests. While simplicity absolutely may be beneficial in some cases, it is definitely not a principle that applies universally to every machine learning model and dataset. Thus, it is important to highlight the need for a more nuanced understanding of model complexity and its impact on performance. It is necessary to consider multiple factors when designing and evaluating machine learning models. Excessively simplistic models may not have the capabilities to explain important patterns present in data, while overly complex models may overfit and fail to perform on any new data. Therefore, the key to designing successful machine learning models is to properly balance simplicity and complexity, according to the specific context and particularities of the task at hand.

Acknowledgment

The authors wish to thank Dr. Bruce Rushing for his oversight and critiques, both on this paper and its associated presentation.

References

[1] Wikimedia Foundation. (2024, October 23). Probably approximately correct learning. Wikipedia. https://en.wikipedia.org/wiki/Probably_approximately_correct_learning

[2] Wikimedia Foundation. (2024b, November 3). Vapnik–Chervonenkis Dimension. Wikipedia. https://en.wikipedia.org/wiki/Vapnik%E2%80%93Chervonenkis_dimension

[3] GeeksforGeeks. (2023, June 12). Vapnik-Chervonenkis Dimension. https://www.geeksforgeeks.org/vapnik-chervonenkis-dimension/

[4] Wikimedia Foundation. (2024c, November 30). *Occam's Razor*. Wikipedia. https://en.wikipedia.org/wiki/Occam%27s_razor

[5] GeeksforGeeks. (2024, July 19). Understanding PAC learning: Theoretical foundations and practical applications in machine learning.

https://www.geeksforgeeks.org/understanding-pac-learning-theoretical-foundations-and-practical-applications-in-machine-learning/

[6] Wikimedia Foundation. (2024a, August 5). Shattered set. Wikipedia. https://en.wikipedia.org/wiki/Shattered_set

[7] Wikimedia Foundation. (2024b, September 25). Computational complexity theory. Wikipedia. https://en.wikipedia.org/wiki/Computational_complexity_theory

[8] Wikimedia Foundation. (2024d, October 26). Generalization error. Wikipedia. https://en.wikipedia.org/wiki/Generalization_error