

Exploring the Hyperparameters of XGBoost Through 3D Visualizations

Ole-Edvard Ørebæk, Marius Geitle

Østfold University Collage, Halden, Norway

Abstract

Optimizing the hyperparameters is one of the most important and time-consuming activities to do when training machine learning models. But the lack of guidance available to optimization algorithms means that finding values for these hyperparameters is left to black-box methods. Black-box methods can be made more efficient by incorporating an understanding of where good hyperparameter values might be located for a specific model. In this paper, we visualize hyperparameter performance-landscapes in several datasets to discover how the XGBoost algorithm behaves for many combinations of hyperparameter values across these datasets. Using this knowledge, it might be possible to design more efficient search strategies for optimizing the hyperparameters of XGBoost.

Keywords

Hyperparameters, hyperparameter optimization, visualizations, performance-landscapes


1. Introduction

Hyperparameter optimization is the task of optimizing machine learning algorithms' performance by tuning the input parameters that influence their training procedure and model architecture, referred to as hyperparameters. While essential to most machine learning problems, hyperparameter optimization is a highly non-trivial task as a given algorithm can have many hyperparameters of different datatypes, and effective values for these differ from one dataset to another [1, 2, 3]. This makes it difficult to determine effective values for specific problems and even more so universally, which in practice results in the use of black-box search algorithms [2, 1]. Black-box algorithms are designed to find solutions without exploitable knowledge regarding the problem and are often based on principles similar to brute-forcing or random guessing. While black-box algorithms for hyperparameter optimization are often quite sophisticated and can be empirically proven to return effective values, they cannot provide much, if any, insight into what makes these effective compared to others. Obtaining insight into how different hyperparameter values, individually and in combination, impact performance under different circumstances would, however, be extremely useful. With such insights, hyperparameter optimization methods can be designed to exploit preexisting knowledge in combination with black-box methods, which is likely more efficient than pure black-box algorithms.

In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021)* - Stanford University, Palo Alto, California, USA, March 22-24, 2021.

✉ oleedvao@hiof.no (O. Ørebæk); mariusge@hiof.no (M. Geitle)

ORCID 0000-0001-8468-7409 (O. Ørebæk); 0000-0001-9528-8913 (M. Geitle)

© 2021 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
 CEUR Workshop Proceedings (CEUR-WS.org)

A useful method of obtaining insight into complex problems is to visualize them, as this allows them to be comprehensively presented. In this paper, we gain insight into the behavior of the hyperparameters of the XGBoost algorithm by visualizing and comparing landscapes of prediction performance generated based on hyperparameter combinations.

The remainder of the paper is structured as follows: In Section 2, we present related work relevant to the visualization of performance-landscapes. The theory behind the XGBoost algorithm is outlined in Section 3. In Section 4, we document the methodology used for generating and visualizing samples of hyperparameter-based performance-landscapes. In Section 5, we present the findings of comparing the generated landscape-samples, and we discuss these findings in Section 6. Finally, in Section 7, we conclude the paper and discuss future work.

2. Related Work

Visualizing problem subjects can be an effective method of intuitively obtaining many types of insight, as demonstrated with many articles within machine learning research. For instance, Li et al. [4] studied the loss landscapes of artificial neural networks through a proposed visualization method based on the principle of random directions and filter-wise normalization. With this method, they provided valuable insight into the nature of artificial neural networks; specifically, how skipped connections affect the sharpness and flatness of loss landscapes and why these are necessary when training very deep networks. Smilkov et al. [5] presented in their study Tensorflow Playground, a tool for providing users an intuitive understanding of neural networks by allowing direct manipulation of the neural networks through visual representations. Other papers [6, 7], though not directly focused on visualizations, utilize some as a method of demonstrating concepts to the reader.

Despite their usefulness, research exploring visualizations directly related to hyperparameters are quite limited. The perhaps most relevant papers are the ones that present tools using visualizations as a method of aiding hyperparameter tuning/analysis processes [8, 9]. However, these papers primarily focus on designing the tools instead of using them practically to obtain insight into hyperparameters.

There are, however, several interesting papers that have investigated performance-landscapes, though not necessarily in the context of hyperparameters. Performance-landscapes are relevant to our paper because they can be visually analyzed to obtain insights into hyperparameters' effects. Performance landscapes have previously been primarily investigated in the context of neural network loss. The most prominent example of this is the earlier mentioned paper by Li et al. [4], which yielded valuable insight in this context. This study also inspired further studies proposing similar methods [10, 7]. Of these, Fort and Jastrzebski [7], among other things, demonstrated similarities in the effects of different neural network hyperparameters.

3. XGBoost

XGBoost, developed by Chen & Guestrin [11], is a gradient boosting decision tree algorithm designed for both regression and classification problems. Being state-of-the-art, the algorithm

is also regularly featured in winning solutions of, e.g., Kaggle¹ competitions. XGBoost is trained through minimizing a regularized objective function (Eq. 1) by iteratively adding base learners f_t , in the form of decision trees, to an ensemble [12, 11].

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)\right) + \Omega(f_t) \quad (1)$$

Here \hat{y}_i and y_i denote the prediction and the target, and l is a loss function that measures the difference between them. The f_t that best minimizes the loss between y_i and the previous iteration's prediction $\hat{y}_i^{(t-1)}$ is greedily added. Additionally, the complexity of the added f_t is penalized to avoid overfitting, as denoted by the regularization term Ω .

Much of XGBoost's regularization and model architecture is defined through hyperparameters. Some of the most impactful are the learning rate, the number of base learners in the ensemble, and the base learners' maximum depth. In this paper we refer to these as *learning_rate*, *n_estimators*, and *max_depth*. The *learning_rate* originates from the principle of shrinkage [12] and is a value that scales base learner weights to reduce their individual influence on the ensemble predictions. *n_estimators* and *max_depth* are quite natural regularization parameters, as they directly influence the architecture of XGBoost's produced models, and therefore significantly impact performance. Other hyperparameters include gamma, l1 and l2 regularization, and various parameters for subsampling and column sampling.

4. Visualizing Hyperparameter based Performance-landscapes

The goal with this paper was to investigate how the hyperparameters of XGBoost affect its prediction performance on different datasets and to investigate potential similarities between these effects. Our motivation was to gain a general insight into XGBoost's hyperparameters compared to simply analyzing datasets individually. To accomplish this, we compared 3D visualizations of performance-landscapes relative to each selected dataset, where each landscape was generated based on a combination of two hyperparameters.

4.1. Generating the Visualization Data

To visualize the hyperparameter performance-landscapes, we selected three of XGBoost's hyperparameters and set their respective value-ranges, as tabulated in Table 1. Generated performance-landscapes were based on the combination of two of these hyperparameters at a time. This resulted in three different performance-landscapes for a given dataset, based on the combination of *learning_rate* and *n_estimators*, *learning_rate* and *max_depth*, and *n_estimators* and *max_depth*. For the classification datasets, the used performance metric was accuracy, while for the regression datasets, Mean Absolute Error (MAE) was used.

4.1.1. Adaptive Zoom

To reduce the amount of data needed to provide highly detailed visualizations, we developed a novel algorithm, named Adaptive Zoom, for adaptively generating more data points in regions

¹<https://www.kaggle.com/>

Table 1
Selected hyperparameters and their value ranges.

Hyperparameter	Value Range
learning_rate	0.1 - 2.0
n_estimators	1 - 500
max_depth	1 - Number of Dataset Attributes

with better predictive performance, thereby keeping computation time spent on regions of low performance minimal. The main idea behind the algorithm was to iteratively "zoom" in on the region of apparent best performance. "Zooming" here refers to adaptively determining the region of known best performance, based on pre-generated points, and generating more points within the determined region. Using this algorithm, we could efficiently obtain highly detailed landscapes by only generating high numbers of points in these specific regions while leaving the remaining regions at lower details. .

The algorithm first identifies the $p\%$ best performing hyperparameter configurations in the landscape. From this set of configurations, the region of best performance, defined by hyperparameter value-ranges, is then determined by the lowest and highest value per hyperparameter. Finally, a new landscape-sample can be generated based on the determined best performing region.

The Adaptive Zoom algorithm is visually demonstrated in Fig. 1, and its pseudocode is contained in Fig. 2.

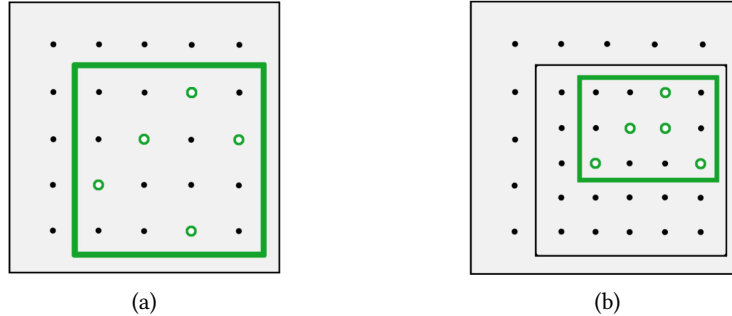


Figure 1: Visual demonstration of how the adaptive zoom algorithm works. The hollow dots represent the best performing points, and the thick outline represent the area of the grid containing these points. Fig. (a) illustrates the first iteration of adaptive zoom, while Fig. (b) represents the second iteration.

4.1.2. Interpolation

To accurately compare the performance landscapes' characteristics, we needed the same hyperparameter combinations across all ranges. However, due to using Adaptive Zoom, a varying amount of landscapes-samples, with varying ranges, were generated for each hyperparameter

```

procedure ADAPTIVEZOOM( $X, p, r \in \mathbb{N}$ )
   $S \leftarrow$  points  $X$  sorted by performance
   $p(S) \leftarrow p$  percentage best performing points
   $R \leftarrow$  empty array
  for all  $h \in p(S)$  do
     $v \leftarrow$  values of hyperparameter  $h$ 
     $v_{min} \leftarrow \text{Min}(v)$ 
     $v_{max} \leftarrow \text{Max}(v)$ 
     $h_{lin} \leftarrow \text{Linspace}(v_{min}, v_{max}, r)$ 
     $R \leftarrow \text{Append}(R, h_{lin})$ 
  end for
   $L \leftarrow$  generated landscape-sample based on  $R$ 
  return  $L$ 
end procedure

```

Figure 2: Pseudocode for the Adaptive Zoom algorithm.

combination. These samples needed to be united so that the performance, relative to each hyperparameter combination, would be represented as a single landscape. To achieve this, we utilized linear interpolation, a method of using a set of known points to generate new points of a specified resolution within the known points' range. For the implementation, we used `scipy.interpolate.griddata`² with the "linear" method, which takes a list of points and a list of their corresponding values, and returns a grid with a specified resolution of interpolated values.

4.1.3. Landscape Generation

The landscapes, with performance values obtained through two-fold cross validation, were generated with standard values of non-investigated hyperparameters at an initial resolution of 20 x 20. Further landscape-samples based on adaptive zoom were generated at an individual resolution of 50 x 50. The number of samples for each landscape was dynamically determined by running Adaptive Zoom until the returned hyperparameter ranges were no different from the previous iteration. Finally, all generated landscape-samples of each hyperparameter combination were merged into singular landscapes through linear interpolation.

4.2. Datasets

To explore effects on performance landscapes, we selected several datasets for both classification and regression problems, as tabulated in Table 2. The datasets were selected to be varied in dataset characteristics, such as size, and number of categorical and continuous attributes, to ensure that generated landscapes would be as varied as possible. This was important to ensure that obtained findings could be generalized and would represent the general relationship between the hyperparameters and performance as accurately as possible.

In terms of preprocessing, all datasets were randomly shuffled to ensure that their contained data was dispersed, categorical features with text values were one-hot encoded, and id-columns were removed.

²<https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.griddata.html>

Table 2

Datasets selected as the basis of generation and comparison of the visualizations.

Dataset	Instances	Type	Attributes	
			Categorical	Continuous
biodegradation ³	1055	Classification	24	17
contraceptive ⁴	1473	Classification	8	1
soybean-large ⁵	307	Classification	35	0
vehicle ⁶	840	Classification	0	18
wdbc ⁷	569	Classification	0	31
winequality-red ⁸	1599	Classification	0	11
auto-mpg ⁹	398	Regression	2	5
forestfires ¹⁰	517	Regression	19	10
housing ¹¹	506	Regression	2	11

5. Findings

To obtain insight into the hyperparameters' effects on performance, we analyzed the landscapes by looking at their general convexity, the compared dominance of the hyperparameters' impact on performance, the locations of optima, the numbers of local optima, and general observations of similarities between the landscapes. The motivation behind analyzing the landscapes' convexity and local optima was to investigate how applicable methods using gradient descent are to hyperparameter optimization. The dominance of the different hyperparameters was investigated to gain insight into which hyperparameters are more important to optimize to achieve the best results. Comparisons of each landscape's optima were investigated to gain insight into the predictability and consistency of their locations.

Note that the landscapes for the regression datasets are flipped to ensure visual consistency. For this reason, the MAE-values are presented as negative.

5.1. Convexity

We found that the landscapes of the learning_rate and n_estimators combination (Figure 3) were generally lacking in convexity and were instead quite flat and jagged in shape. There were, however, some exceptions to this. The landscape of the forestfires dataset had a clear convex shape in the learning_rate axis, where lower values of learning rates yielded better

³<http://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation>

⁴<https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice>

⁵[https://archive.ics.uci.edu/ml/datasets/Soybean+\(Large\)](https://archive.ics.uci.edu/ml/datasets/Soybean+(Large))

⁶<https://datahub.io/machine-learning/vehicle>

⁷[https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

⁸<https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/>

⁹<https://archive.ics.uci.edu/ml/datasets/auto+mpg>

¹⁰<https://archive.ics.uci.edu/ml/datasets/Forest+Fires>

¹¹<https://archive.ics.uci.edu/ml/machine-learning-databases/housing/>

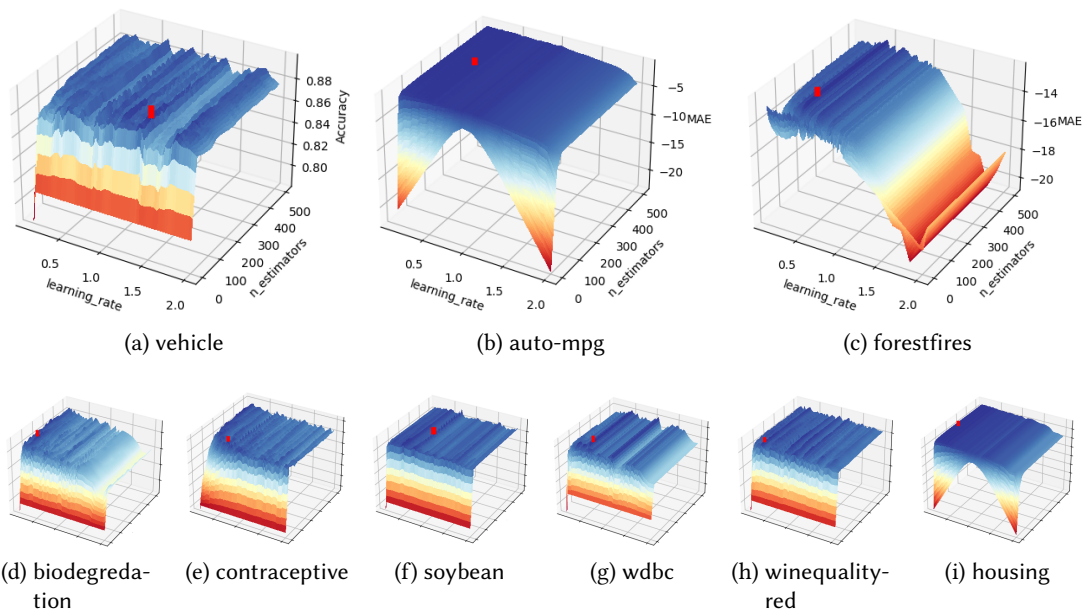


Figure 3: The interpolated performance-landscapes based on the combination of `learning_rate` and `n_estimators`.

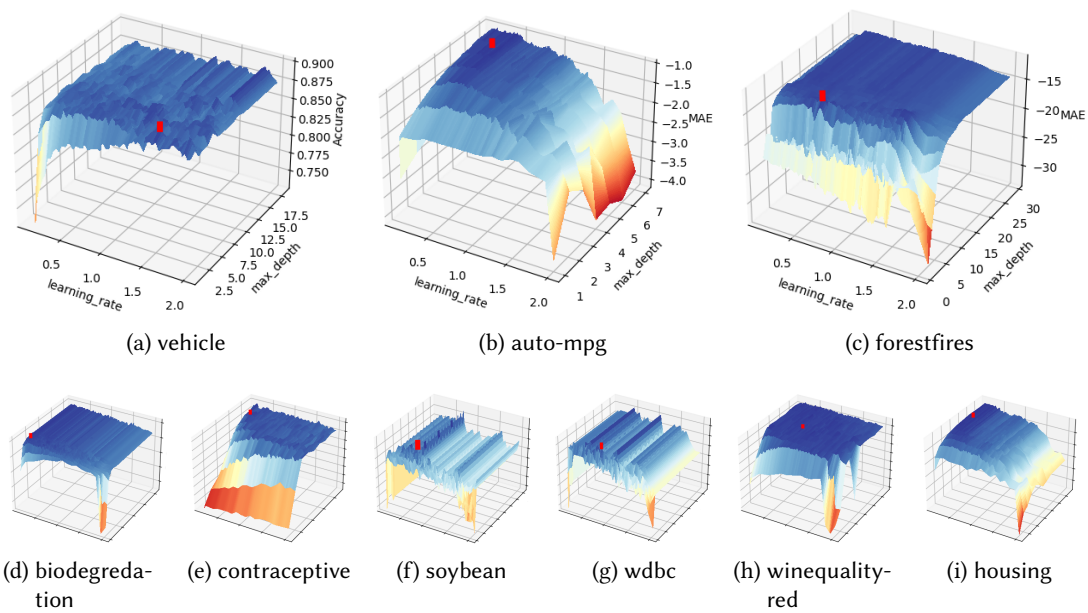


Figure 4: The interpolated performance-landscapes based on the combination of `learning_rate` and `max_depth`.

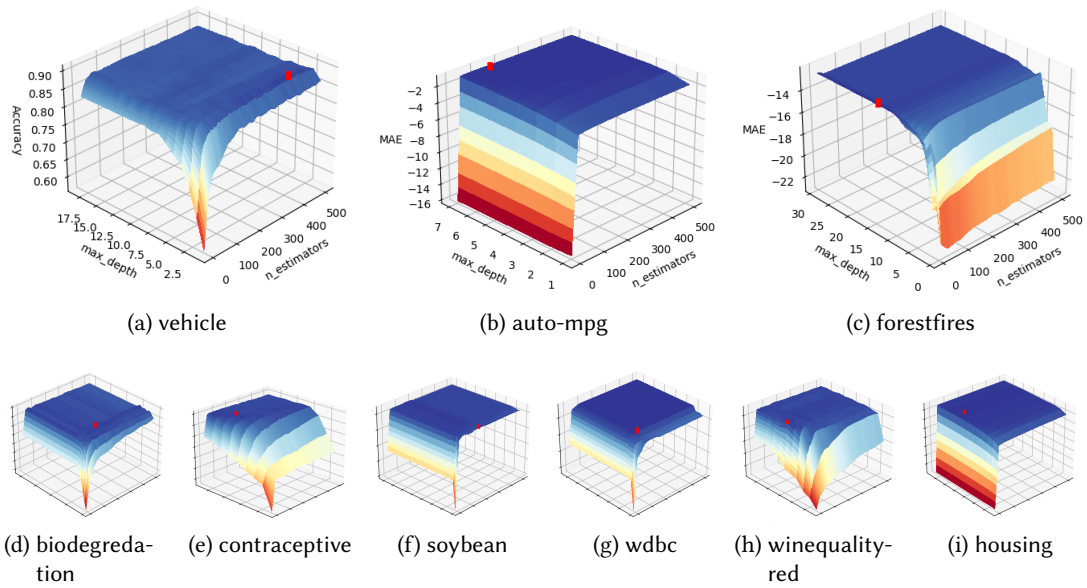


Figure 5: The interpolated performance-landscapes based on the combination of `n_estimators` and `max_depth`.

performance than higher values. The landscapes of biodegradation, wdbc, housing, and auto-mpg also seemed to have some convexity in this axis, though they were still relatively flat in overall shape.

For the landscapes of the `learning_rate` and `max_depth` combination (Figure 4) the convexity was quite varied from one dataset to another. The contraceptive dataset was relatively convex in the `max_depth` axis, which was also the case for forestfires and winequality-red. Auto-mpg and housing were somewhat convex in both the `learning_rate` and `max_depth` axis.

Most of the landscapes of the `n_estimators` and `max_depth` combination (Figure 5) were flat in overall shape. However, contraceptive, winequality-red, and perhaps forestfires, had some convexity to them.

5.2. Dominance

Based on the landscapes, it seemed that `n_estimators` had, compared to `learning_rate`, a considerably larger impact on performance. However, this only seemed the case for `n_estimators`-values lower than approximately 100, which materialized as a "wall" in the landscapes. This was apparent for all dataset-relative `learning_rate` and `n_estimators` combination landscapes (Figure 3), except for forestfires, which did not seem to contain this wall. For `n_estimators`-values over 100, we found that the combination of `learning_rate` and `n_estimators` resulted in quite jagged landscapes, with `learning_rate` appearing to be the most dominant. We also observed that jaggedness in the `n_estimators` axis was not equal for all `learning_rate` values. Only the landscapes of the auto-mpg and housing datasets were lacking in visible jaggedness.

Comparing `learning_rate` to `max_depth` (Figure 2), it seems that these hyperparameters tend

to have a nearly equal impact on performance except for with a few datasets, such as forestfires, contraceptive and winequality-red. For these, `max_depth` appeared to have a larger effect, creating a wall similar to those observed in the combinations of `learning_rate` and `n_estimators` (Figure 3). We also found that `max_depth` tended to stop impacting performance beyond certain values. These values seemed to never exceed `max_depth = 15`, but were observed for several datasets as being less.

For the combination of `n_estimators` and `max_depth` (Figure 5), we found that the most dominant hyperparameter changed from one dataset to another, with there sometimes being performance walls in the `n_estimators` axis, sometimes in the `max_depth` axis, and sometimes in both. Beyond these performance walls, the two hyperparameters seemed about equally dominant.

5.3. Optima

For all combinations and datasets except vehicle, the optima were located within the `learning_rate` value-range of 0.1 to 1.0. For vehicle, the optima were located around the `learning_rate` value of 1.5. We also observed that the optima were always located between the `n_estimators` value-range of roughly 100 to 250.

The optima for each dataset was observed to generally be located around the same `learning_rate` value within all relevant hyperparameter combinations (Figure 3 and 4), with winequality being the only exception to this. For this dataset, the optimum was an entirely different `learning_rate` value for the combination of `learning_rate` and `n_estimators` (Figure 3) compared to `learning_rate` and `max_depth` (Figure 4). Relative to this, `n_estimators` optima seemed a bit more variable. `Max_depth` optima were located within values less than 15.

5.4. Local Optima

In terms of local optima, landscapes based on the combination of `learning_rate` and `n_estimators` (Figure 3) seemed to contain many local optima. These did, however, seem to be predominantly formed from the influence of `learning_rate`.

Local optima were also observed to be plentiful for the combination of `learning_rate` and `max_depth` (Figure 4). Here, it seemed that local optima were about equally based on `learning_rate` and `max_depth`, resulting in much more chaotic landscapes. This was, however, only the case for `max_depth` values less than 15 due to the observations outlined in Section 5.2.

For the combination of `n_estimators` and `max_depth` (Figure 5), we observed that local optima seemed less common. Most of the landscapes were relatively flat and only datasets like biodegradation, contraceptive, vehicle and winequality-red had anything that could be referred to as local optima. But even for these, local optima were small in size.

5.5. Generally

For the `learning_rate` and `n_estimators` combination (Figure 3), we found that most landscapes were quite similar in characteristics, such as general shape, convexity, and local optima. There were, however, some differences between the landscapes of the classification and regression

datasets. Most notably that regression datasets generally seemed to contain less local optima, and that forestfires' landscape shape was completely unique compared to the others.

The different `learning_rate` and `max_depth` combination landscapes (Figure 4) also seemed to have reoccurring characteristics. There seemed to generally be a sharp dip in performance close to `learning_rate = 2.0` and `max_depth = 1`. The only landscape that was lacking this dip was the one generated from the vehicle dataset. We also found that biodegradation, soybean and `wdbc` had very similar landscapes for the combination of `learning_rate` and `n_estimators`, which can also be said for `winequality-red` and forestfires.

We also observed that several datasets resulted in similar landscapes for all hyperparameter combinations. The clearest example of this was housing and `auto-mpg`, which seemed nearly identical. This was also the case for soybean and `wdbc`. In terms of landscape characteristics, the two datasets that stuck out the most were contraceptive and forestfires. For contraceptive, `max_depth` seemed to have a much larger influence on performance than for other datasets, while for forestfires, `n_estimators` appeared to have no significant performance impact.

6. Discussion

Our goal with this paper was to obtain insights into how and under what conditions the hyperparameters of XGBoost affect its performance by analyzing and comparing hyperparameter-based performance landscapes for various datasets. The motivation was to use such insights to aid the efficiency of hyperparameter optimization strategies for XGBoost.

The findings showed several indications that analyzing the performance-landscape visualizations helped gain insight into effective search ranges of the investigated hyperparameters; `learning_rate`, `n_estimators`, and `max_depth`. For `learning_rate`, the fact that optima were within the value-range of 0.1 to 1.0 for all datasets except one, indicates that this range might generally be optimal when searching for values of this hyperparameter. Similarly, with `n_estimators`, the search range of 100 to 250 may be more optimal than other alternatives under 500 estimators, based on the optima generally being located within this range. We also discovered that `n_estimators` values less than 100 formed a "wall" of performance-increase, most notably in landscapes of the `learning_rate` and `n_estimators` combinations (Figure 3). This could imply that `n_estimators` less than 100 can reasonably be excluded from hyperparameter searches. Regarding `max_depth`, we observed that values greater than 15 for this hyperparameter seemed not to affect XGBoost's performance, implying that limiting the search range from 1 to 15 might be reasonable. These implications are useful for standardizing searches regarding these hyperparameters, which makes their optimization more efficient by reducing computation time.

By analyzing and comparing the general characteristics between the performance-landscapes, we also found indications that the general behavior of XGBoost's hyperparameters is somewhat predictable. Specifically, we found that landscapes based on the same hyperparameter combination had general similarities across the datasets, and that certain datasets resulted in almost identical landscapes, optima included, across all combinations. This was observed to be the case for datasets both similar and dissimilar in general characteristics. Based on this, it is possible that performance-landscapes based on XGBoost's hyperparameters can largely be

predicted from aspects of the datasets, though any such specific aspects were not obvious from the findings. However, if such landscape predictions are possible and reliable, it could mean that effective search methods and hyperparameter values for XGBoost can potentially be made somewhat deterministic. Based on this assumption, such predictions would be a prime target to take advantage of when designing future hyperparameter optimization methods relevant to XGBoost. Worth noting, however, is that this is still unlikely to completely remove the need for black-box methods, due to the large amount of local optima and general flatness observed in the landscapes. Nevertheless, a combination of the two would likely be beneficial.

Compared to earlier studies, our visualization method seemed effective for obtaining hyperparameter insight despite its simple premise and implementation. For instance, compared to previously suggested tools using visualizations for hyperparameter tuning processes [8, 9], our method does not need to be tied to a specific tuning process but instead focuses on several datasets to obtain more general insight into the hyperparameters. It is also apparent that exploring performance landscapes, similarly to earlier papers investigating neural networks [4, 7, 10], is also useful for investigating hyperparameters of other algorithms/methods.

7. Conclusion and Future Work

In this paper, we attempted to gain insight into how XGBoost’s hyperparameters affect performance by analyzing hyperparameter-based performance-landscape visualizations. The findings indicate that the method of visualizing performance-landscapes, based on combinations of two hyperparameters at a time, is a useful tool for gaining insight into, e.g., effective ranges of XGBoost’s hyperparameters. This was derived from, e.g., how optima were generally located between 0.1 and 1.0 `learning_rate`, and 100 and 250 `n_estimators`. Also, how `max_depth` was never observed to have an effect on performance for values greater than 15, and how `n_estimators` typically had a wall of performance within the value range of 1 to 100. We also found indications that the visualization method is effective for gaining insight into the general and specific behavior of XGBoost’s hyperparameters with different datasets. This was derived from how we observed that most datasets had common landscape-characteristics and how some dataset-landscapes were nearly identical regardless of similarity/dissimilarity of dataset-characteristics. Thus, we conclude that 3D visualizing hyperparameter-based performance-landscapes is an effective tool for obtaining various types of insight into the behavior of XGBoost’s hyperparameters. And that these insights can possibly be used to design more efficient hyperparameter optimization strategies for this algorithm.

While the findings and their implications are promising, several points of investigation should be explored in future work. For instance, larger datasets should be investigated to ensure that the findings are generalizable. During the process of generating the visualizations, we also noticed that the interpolation method would produce artifacts for certain landscapes. Minor aspects of the visualizations might therefore be inaccurate and should be fixed. The utilized visualization method is limited in that it can only visualize two hyperparameters at a time. However, it might be possible to use hyperparameter-based vectors to generate the visualizations, similar to how Li et al. [4] did with neural network weights, which would solve this limitation. As discussed in Section 6, we observed that certain datasets, of similar and

dissimilar characteristics, produced strikingly similar landscapes. The cause of this is likely a worthwhile point of further research.

References

- [1] F. Hutter, J. Lücke, L. Schmidt-Thieme, Beyond manual tuning of hyperparameters, *KI-Künstliche Intelligenz* 29 (2015) 329–337.
- [2] M. Feurer, F. Hutter, Hyperparameter optimization, in: *Automated Machine Learning*, Springer, Cham, 2019, pp. 3–33.
- [3] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, *Journal of machine learning research* 13 (2012) 281–305.
- [4] H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, Visualizing the loss landscape of neural nets, in: *Advances in Neural Information Processing Systems*, 2018, pp. 6389–6399.
- [5] D. Smilkov, S. Carter, D. Sculley, F. B. Viégas, M. Wattenberg, Direct-manipulation visualization of deep networks, *arXiv preprint arXiv:1708.03788* (2017).
- [6] N. Frosst, N. Papernot, G. Hinton, Analyzing and improving representations with the soft nearest neighbor loss, *arXiv preprint arXiv:1902.01889* (2019).
- [7] S. Fort, S. Jastrzebski, Large scale structure of neural network loss landscapes, in: *Advances in Neural Information Processing Systems*, 2019, pp. 6706–6714.
- [8] T. Li, G. Convertino, W. Wang, H. Most, T. Zajonc, Y.-H. Tsai, Hypertuner: Visual analytics for hyperparameter tuning by professionals, in: *Proceedings of the Machine Learning from User Interaction for Visualization and Analytics Workshop at IEEE VIS*, 2018.
- [9] H. Park, J. Kim, M. Kim, J.-H. Kim, J. Choo, J.-W. Ha, N. Sung, Visualhypertuner: Visual analytics for user-driven hyperparameter tuning of deep neural networks, in: *Demo at SysML Conference*, 2019.
- [10] S. Fort, A. Scherlis, The goldilocks zone: Towards better understanding of neural network loss landscapes, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019, pp. 3574–3581.
- [11] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
- [12] J. H. Friedman, Greedy function approximation: a gradient boosting machine, *Annals of statistics* (2001) 1189–1232.