How does scaling a learning curve influence the curve fitting process?

Chaan van den Oudenhoven
Supervisor(s): Tom Viering , Taylan Turan, Cheng Yan

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 26, 2025

Name of the student: Chaan van den Oudenhoven Final project course: CSE3000 Research Project
Thesis committee: Tom Viering , Taylan Turan, Cheng Yan, Arie van Deursen

Abstract

Learning curves show the learning rate of a classifier by plotting the dataset size used to train the classifier versus the error rate. By extrapolating these curves it is possible to predict how well the classifier will perform when trained on dataset sizes that are currently not available. This can be useful when trying to determine which classifier to select when dealing with a classification problem. Obtaining these learning curves is usually done by fitting a parametric model to the learning data. This paper analyzes the potential of fitting the curve in a different space scaling the fitting data. This is done by analyzing the accuracy of the fit and the frequency of the fit succeeding. Our main findings are that log scaling produces better MSEs than not scaling, while exponential scaling is inconclusive.

1 Introduction

Modern-day machine learning models are capable of doing various things, with a common use case being solving a classification problem [1][2]. However, to solve these problems, oftentimes significant amounts of data are required to sufficiently train the models. What people oftentimes define as sufficient in this context is that the performance of the model should not exceed a certain error rate (in the case of classification models). The error rate, also known as the classification error [3], is the frequency of the classifier getting a classification incorrect, represented as a ratio. Finding out exactly how much data you need to reach a certain error rate can be somewhat tricky, yet very important due to the potential monetary or computational cost associated with gathering large amounts of data. For this reason, it is often times of interest to try to predict this point based on the error rate of lower quantities of data.

Mohr [4] describes learning curves as the following: "Learning curves describe a system's performance on a task as a function over some resource to solve that task." The way these curves are usually generated is by varying the amount of data used to train a classifier and then measuring the error rate by evaluating on either the training data, validation, or test set. Curve fitting however is the process of taking some parametric model and trying to fit it to the learning curve data. For this process, a parametric model is defined, and the curve-fitting algorithm aims to find the optimal parameters such that the loss function is minimized on either the data used to do the curve on some subset of the data points. This curve can then be used to predict the error rate on data quantities that are currently unavailable. This is called interpolation and extrapolation.

This curve fitting does, however, have a number of flaws. It can get stuck in a local minimum [5] or potentially fail to fit all together. This can potentially make the extrapolation process less accurate. Furthermore, Mohr [6] explains a number of pitfalls that researchers should look out for when trying to evaluate different curve-fitting techniques. Namely, one of the mentioned pitfalls regards trying to scale the data before attempting to curve fit. It explains that the MSE in a certain space is distinctly different from fitting in another space. Regardless, we decided to investigate this pitfall more thoroughly to discover the impact this procedure might have and answer the question: "How does scaling a learning curve influence the curve fitting process?"

2 Related Works

Research regarding curve fitting after scaling is severely limited. Regardless, we still discuss some related works regarding the general field of this research.

2.1 Learning Curves

Extensive work has been done within the field of learning curves. Mohr [6] for example provides a comprehensive overview of the general shape of learning curves. He debunks the misconception that learning curves must always be well-behaved. Well-behaved learning curves are ones that are both non-increasing and smooth[7] in terms of error rate. However, some learning curves are ill-behaved and thus do not always show improvement as the size of the dataset used to train the classifier increases. One example that he mentions has to do with something he refers to as the "peaking phenomenon", where a classifier improves up until a certain point and then becomes worse.

Furthermore, there have been various studies regarding ways to improve the general curve fitting process. For example, one study [8] proposes a method that involves using artificial intelligence in the form of a neural network to curve fit. They manage to fit onto, what they deem, difficult curves with decent accuracies.

Another study [9] aims to improve the fitting process on specifically NURBS curves. They propose using the geometric properties of the curves to generate weights that can be used in the fitting process. They find that their method allows for better fitting even on noisy data.

Lastly, there is also a study [10] that introduces the CurvPy library. This library aims to make curve fitting more accessible to potential users while still retaining enough complexity for it to be useful.

2.2 LCDB Database

Mohr [11] also introduces the LCDB database. It contains, as of writing this, more than 4000 unique learning curves spread over various classifiers from Scikit-learn, with more being added in the future. This database attempts to ensure that the learning curves accurately reflect the true error rates of the classifiers by creating curves using multiple datasets and splitting these using 5 outer splits which each have 5 inner splits, resulting in 25 unique learning curves for each unique pair of classifier and dataset. This paper contains a couple of notable findings. First of all, the majority of the learning curves they analyzed were well-behaved. Furthermore, they also find that there is no overall curve model, but rather that it depends on the size of the training dataset. All of the experiments we performed make use of the learning curves provided by this paper.

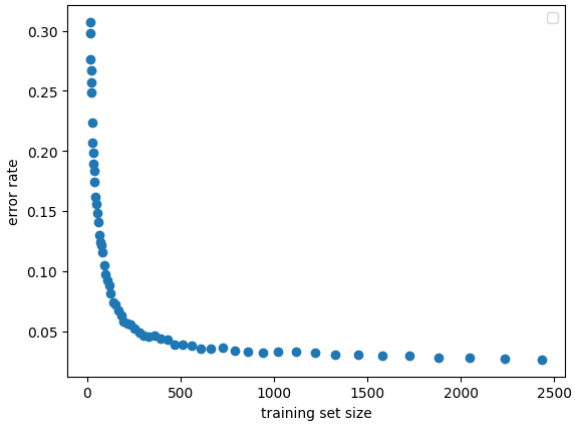

Figure 1: Example of a learning curve, taken from the LCDB database [11]

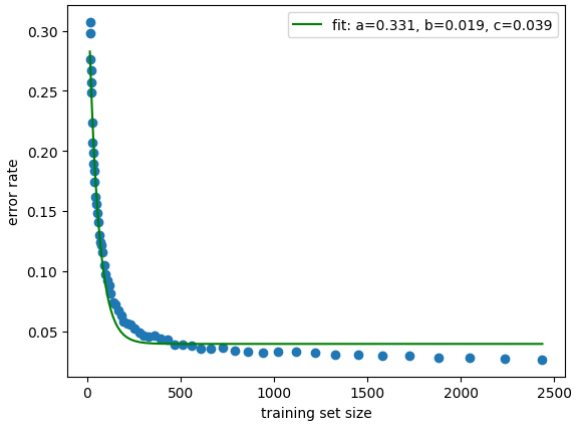

Figure 2: Example of a learning curve along with a curve produced by fitting a parametric model on it, taken from the LCDB database [11]

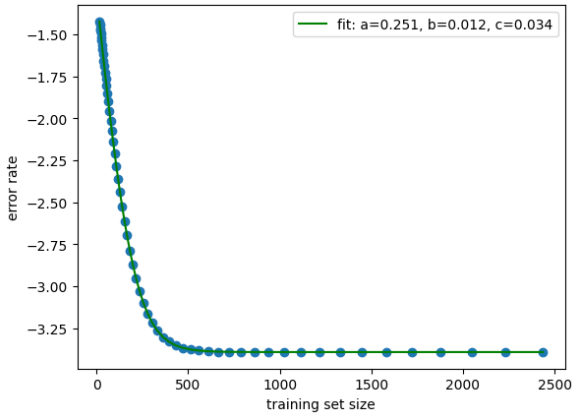

Figure 3: Example of a learning curve along with a curve produced by fitting a parametric model on it in a space in which we scaled the Y axis with the natural logarithm, taken from the LCDB database [11]

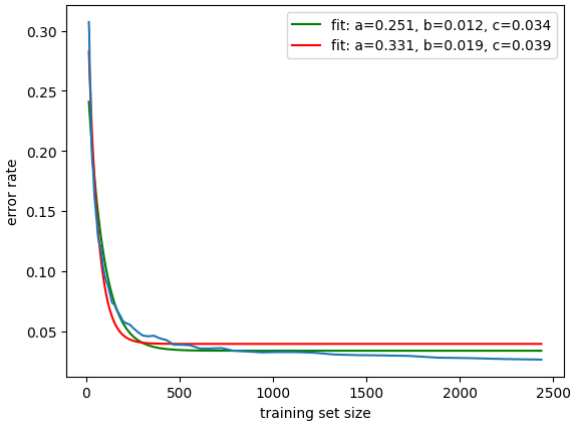

Figure 4: Example of 2 learning curves. The green curve is the curve found in the log space after being scaled back to the normal space, while the red curve was fit in the normal space, taken from the LCDB database [11]

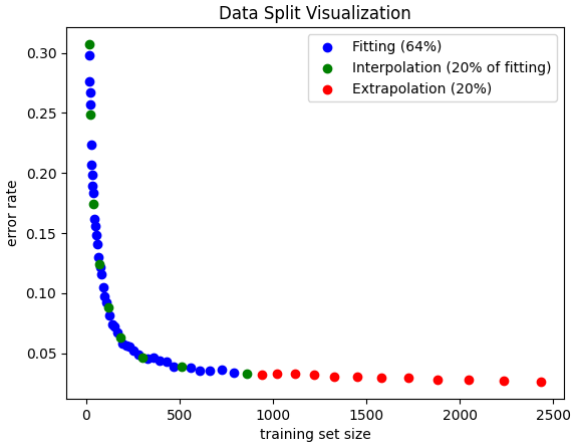

Figure 5: Example of a learning curve divided into fitting data, interpolation data, and extrapolation data, taken from the LCDB database [11]

2.3 Curve Fitting

Curve fitting is a technique that involves trying to fit a parametric model to a given dataset so that the loss function is minimized. One way of doing this is through the Levenberg-Marquardt algorithm (LM)[12]. It combines the ability to converge quickly from the Gradient Descent algorithm with the ability to accurately fine-tune from the Gauss-Newton method. We present a visualization of a learning curve and a parametric model fit on it in figure 2.

2.4 Measuring Performance Using Interpolation and Extrapolation

The primary ways in which we measure the performance of our techniques is through interpolating and extrapolating the data. Both of these techniques make use of the Mean Squared Error (MSE) which is defined as follows:

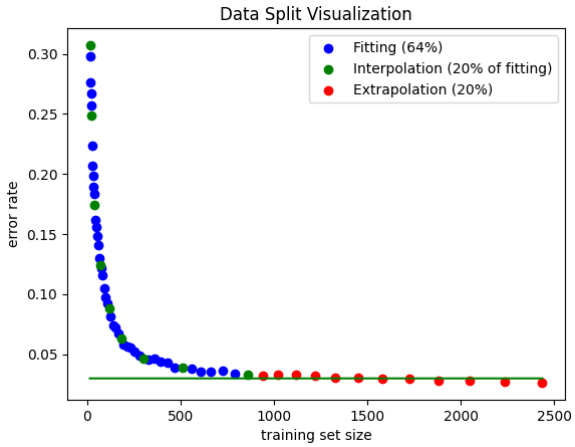

Figure 6: Example of a learning curve divided into fitting data, interpolation data and extrapolation data along with a curve that produces great extrapolation but bad interpolation, taken from the LCDB database [11]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_{actual} - Y_{predicted})^2$$

Here Y_{actual} is the actual data point, which represents the error rate of the classifier when trained on the i-th dataset size, while $Y_{predicted}$ is the predicted error rate at that dataset size according to the parametric model.

Interpolation involves trying to approximate unseen data points between data points that have already been acquired [13]. We illustrate this in figure 5. Even though we have access to error rates of classifiers trained on dataset sizes of, for example, 8 and 11, what if we want to know the error rate of a classifier trained on a dataset of size 9? Having access to a continuous function that fits well on all the data points would allow us to interpolate between these known data points and come up with an estimation of any unknown data point in between them. Extrapolation on the other hand involves calculating the MSE between predicted values and data points to the right of the last point used for the curve fitting. These points are not in between any known points, or at the very least points that were used for curve fitting. One practical use case in which extrapolation is useful is when performing early discarding. Early discarding involves having to choose between multiple classifiers and then discarding some of them if, according to your extrapolation results, they will perform worse compared to the rest and training dataset sizes grow [14]. A curve that gives a good extrapolation value indicates that it more accurately predicts the error rate for larger, yet unseen, amounts of training data than a curve with a lower extrapolation value.

A potential flaw regarding measuring interpolation and extrapolation independently is something we illustrate in figure 6. This figure shows an example of a line that produces great extrapolation but bad interpolation. Even though this line would show great performance if we were to measure only extrapolation, overall it would still be a very bad fit. For this reason, we decided to use the sum of interpolation and extrapolation instead to see how well our curves fit on unseen data in all parts of the graph.

2.5 Fitting after axis scaling

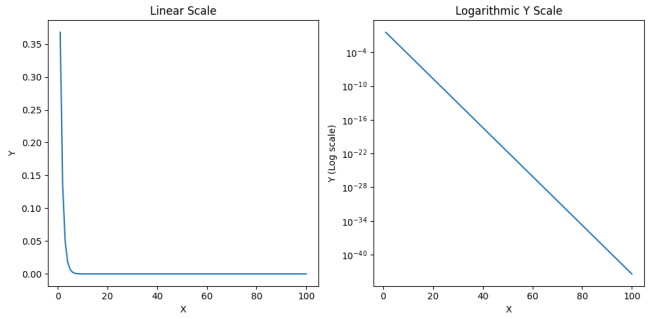

Figure 7: Example plot of the effect of log scaling the Y axis of a learning curve

The main behavior that this paper aims to research is what happens when we try to fit our curves after scaling one of the two axes. To illustrate this, we showcase example figure 7. The data in the left graph shows an exponential relationship between X and Y, but if we plot it after scaling the Y axis, it becomes a linear relationship. We hypothesize that trying to fit this simpler, more linear, curve could result in a faster convergence rate of the MSE, which is what the curve-fitting algorithm is trying to optimize. To then produce a curve that fits in the original space, we scale both the data and the function with the inverse of our original scaling method. Additionally, we also wish to know what the effect would be on the quality of the fit itself, both in terms of whether the curve fits more reliably or if the MSE of the fit turns out to be better. Our general procedure is as follows: First we define the learning curve we want to fit on and the parametric model we want to fit with. The LCDB database comes with learning curves for classifiers evaluated on the training data, evaluation data, and test data. Due to the fact that in the real world, we want to predict how well our classifier will perform on test data, we decided to use this one for our experiments. Furthermore, since the LCDB database comes with learning curves produced using different training data splits, we use the mean curve of all 25 splits to attempt to avoid any bias the curve might have due to taking a specific split. As for our parametric model, we use one of the parametric models defined by Viering[11]:

$EXP3 : a * e^{-b*x} + c$ Next we define and use our scalar function. Due to time constraints, our experiments focus on the effects of scaling the Y-axis. The way we do this is by scaling both the Y values and the entire parametric function with the natural logarithm (in this example). This means that all the data points go from (X, Y) to (X, log(Y)) while our parametric model becomes $log(a * e^{-b*x} + c)$.

For exponential scaling, this is very similar, except we convert all the data points to (X, e^Y) and our parametric model becomes $e^{(a*e^{-b*x}+c)}$ we fit the curve in this space, and subsequently, we take the optimal parameters found in the scaled

space and use them in the original space Lastly we analyze our findings using our metrics. We are most interested in analyzing the following behavior:
First we wish to know the accuracy of the fit in terms of MSE for the sum of interpolation and extrapolation. We use the sum due to the previously stated flaw regarding only measuring one. Secondly we wish to know what the frequency of successful fits is. In the case that two fitting procedures result in similar MSEs, it is still possible that one fitting procedure produces successful fits more frequently which could be beneficial from a computational standpoint as that way fewer retries would be required in case of the curve fitting procedure producing a failed fit. We define a failed fit as one that surpasses the default amount of max function calls that the Scipy curve fit uses, which is 800 or if the best MSE produced by any of the scaling methods is more than 100. This last point was inspired by Viering[11]

3 Experimental Setup

Learner	No Scalar vs Log Scalar	No Scalar vs Exp Scalar
SVC_linear	.022	<.001
SVC_poly	.003	<.001
SVC_rbf	.025	<.001
SVC_sigmoid	.193	.419
DecisionTree	<.001	<.001
ExtraTree	<.001	<.001
LogisticRegression	<.001	<.001
PassiveAggressive	.035	<.001
Perceptron	<.001	<.001
RidgeClassifier	<.001	<.001
SGDClassifier	.002	<.001
MLPClassifier	<.001	<.001
LDA	.082	.013
QDA	.436	.013
BernoulliNB	.047	<.001
MultinomialNB	.738	.008
ComplementNB	.054	.012
GaussianNB	.621	.162
KNN	<.001	<.001
NearestCentroid	.749	.015
ens.ExtraTrees	<.001	<.001
ens.RandomForest	.002	<.001
ens.GradientBoosting	<.001	<.001
DummyClassifier	.010	.043

Table 1: Wilcoxon signed-rank test results comparing different scalars

To conduct our experiment and answer our research question we make use of the following methodology:

- First, we collect the learning curves of the 24 different learners that are provided in the LCDB database. These come with precomputed mean learning curves using different train and test splits to create curves that more accurately represent the true learning rate, instead of being biased due to choosing a specific split.
- Subsequently, we perform the curve fitting using the model EXP3. As for our scalars, we scalars, we scale with the natural log and the exponential according to our previously defined methodology. Furthermore, due to certain learning curves producing error rates of 0, we set the MSE to be a very small number of 0.0001 at a minimum so we can still perform the Natural_log scaling.
- After we have gathered the models and functions, we perform the curve fitting procedure on each unique pair of learners and models using the SciPy [15] curve fitting implementation. Due to the fact that curve fitting has a chance of failing, and because we wish to analyze how curve fitting affects the chance of a successful fit, we perform 50 curve fits per combination of learner and dataset for every scaling technique. The only difference between curve fits is the initial guesses of the parameters. We sample these from the uniform distribution between 0 and 1. Since the LM algorithm is deterministic, sampling for our initial guesses is necessary to produce different fits between our 50 attempts, although we recognize that the choice of our sampling distribution and the number of attempts could possibly have an influence on the results of our experiment.
- Lastly we scale the curve back to the original space, once again as defined in our methodology. We then compare it to a curve that was fit on data in the original space using the following two metrics:
  - The MSE calculated for the sum of the interpolation and extrapolation. We measure interpolation by taking the first 80% of our data points and uniformly taking 20% of that 80 to produce our interpolation points resulting in 64% of the total data being used for fitting. For extrapolation, we use the last 20% of the data.
  - The frequency of a curve fit being successful. Oftentimes the curve fitting process fails, thus a higher success frequency would indicate that the curve fitting procedure was easier to perform.

4 Results

Figure 8 depicts a bar chart depicting the performance of the curve fitting algorithm when using different scaling methods. For each learner, the three bars depict the mean MSE of Sum(interpolation, extrapolation) over all the datasets for not scaling, scaling using the log, and scaling using the exponential function. We see that out of the 3 scaling methods, log scaling produces the best overall MSE, while exponential scaling.

In table 2 we display how often each scaling technique produces the best MSE. To clarify, for each learner we fit on all of the datasets using the specific scaling techniques. Every time a specific scalar produces the best MSE, we count that as one "best MSE". We observe that for every single learner scaling with the log produced the highest frequency of best MSEs while exponential scaling produced the second bet for every learner except DummyClassifier, ens.ExtraTrees and

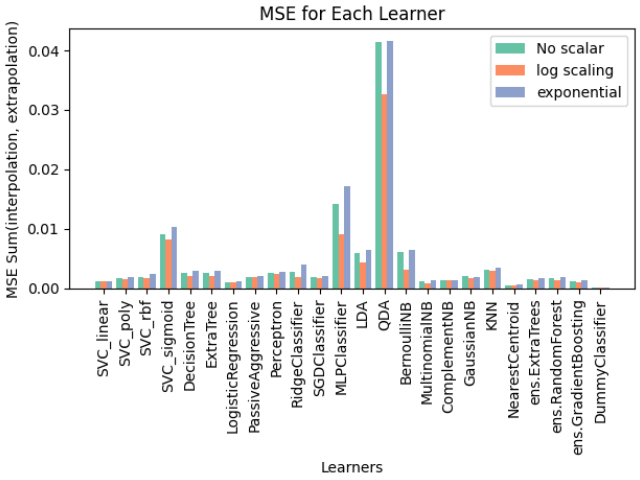

Figure 8: Bar chart depicting the performance of the curve fitting algorithm when using different scaling methods. For each learner, the three bars depict the mean MSE of Sum(interpolation, extrapolation) over all the datasets for not scaling, scaling using the log, and scaling using the exponential function

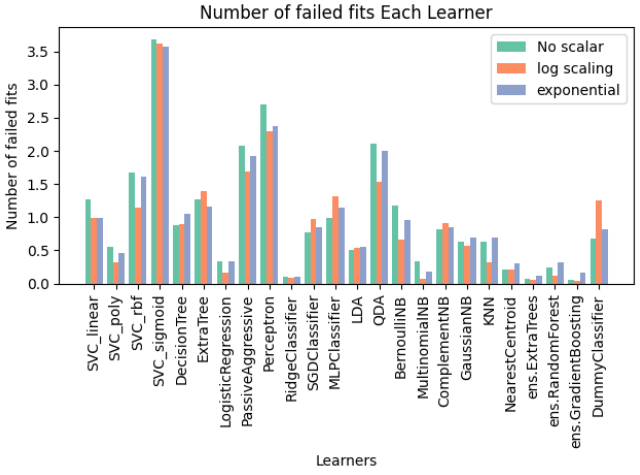

Figure 9: Bar chart depicting the mean frequency of fits failing over the different scaling methods. For each learner, the three bars depict the mean frequencies over all the datasets for not scaling, scaling using the log, and scaling using the exponential function

RidgeClassifier.

In table 1 we produce a Wilcoxon signed-rank test[16] comparing the MSE results between the pairs (no scaling, log scaling) and (no scaling, exponential scaling). This test allows for comparisons between pairs of dependent data. Our null hypothesis in this case is:

$$H_0 :$$

The median difference between paired samples is zero (no difference).

Furthermore, we reject the null hypothesis for $P<.05$
It is important to note that obtaining a low P only indicates that it is highly likely that the two compared datasets follow a different distribution, but not necessarily that one produces better MSEs than the other. For this reason, we have to cross-reference it with figure 8

5 Discussion

5.1 Discussion of Results

Most of our findings, especially table 2 appear to indicate that scaling the Y axis with the log causes the parametric model to create better fits for the entire curve, according to our metric. This finding supports our previously stated hypothesis.
As for scaling the Y axis with the exponential function, it is quite difficult to draw any conclusions. Simply looking at figure 8 seems to indicate a slight deterioration of the mean fit quality for most of the classifiers. However, when looking at figure 2, there seem to be certain classifiers for which scaling with the exponential function produces better fits significantly more often. Examples of these are these are SVC_sigmoid and GaussianNB.

5.2 Potential Flaws and Explanations

Regarding our methodology, there are a couple of potential flaws. First of all, as mentioned before, our method for comparing the frequency of successful fits involves changing the initial guesses of the parameters by sampling from a distribution between our 50 fit attempts. The resulting frequency could however be heavily influenced by the distribution that these guesses are sampled from along with the amount of attempts. Increasing the number of attempts and distributions was however infeasible due to a lack of computational resources.

Furthermore regarding our methodology for Y axis scaling, while this methodology produces a better overall MSE when log scaling compared to not scaling at all, what could be the case is that this is influenced by the loss function that this is trying to optimize. It is possible that the main reason why Y axis scaling produces different parameters is simply because the loss in the scaled space is different than in the normal space. This possibility has been mentioned by [6] and Singh [17] raises concerns about fitting in the log space as well. This could also partially explain why log scaling seems to perform so well. Since most learning curves are

Learner	No scalar	log scaling	exponential
SVC_linear	12	42	18
SVC_poly	9	49	14
SVC_rbf	9	45	18
SVC_sigmoid	5	37	30
DecisionTree	4	61	7
ExtraTree	2	60	10
LogisticRegression	11	50	11
PassiveAggressive	9	42	21
Perceptron	10	48	14
RidgeClassifier	13	47	12
SGDClassifier	14	44	14
MLPClassifier	4	59	9
LDA	9	37	26
QDA	16	34	22
BernoulliNB	10	43	19
MultinomialNB	8	27	21
ComplementNB	6	33	17
GaussianNB	8	35	29
KNN	10	48	14
NearestCentroid	10	30	32
ens.ExtraTrees	12	51	9
ens.RandomForest	13	48	11
ens.GradientBoosting	8	56	8
DummyClassifier	18	39	14

Table 2: Best MSE counts for each learner and scalar. Each cell indicates the number of datasets for which that scalar produced the best MSE for that classifier

well-behaved and more importantly have a generally non-increasing error rate. What might happen is that data points on the left side of the curve, which are bigger compared to the right, have comparatively less weight in the log space, which causes the curve to prioritize later points of the fitting data when curve fitting. Then, because data points on the right side of the curve tend to converge, i.e their difference in terms of error rate is not as much as on the left, having a curve that weighs the later fitting points more could also result in it fitting very well on the extrapolation points.

Lastly, we performed our methodology on the classifier and dataset combinations that the LCDB database provides. However, some of the curves and datasets in this database contain quite a lot of noise. One of which is the learner SVC_Sigmoid, which is also apparent in figure 9 in which it produced the highest mean frequency of failed fits compared to all of the classifiers. While we show our results per classifier, we do not do this per dataset. Including pairs of classifiers and datasets that produced very noisy learning curves could have had a significant impact on our results, however for the sake of not nitpicking we still decided to include them in our analysis.

6 Conclusions and Future Work

6.1 Conclusion

In conclusion, in this paper, we aimed to analyze the effects of curve fitting in a scaled space. We did this by fitting after scaling the Y-axis using log scaling and exponential scaling. We evaluated these fits by measuring the mean sum of the interpolation and extrapolation for every classifier over all the datasets provided by the LCDB database along with performing a Wilcoxon signed rank test and showing a comparison table depicting how often each scaling technique managed to produce the best fit for a specific pair of classifier and dataset. We find that in the majority of cases, scaling with the log produced the best MSEs for different (classifier, dataset) pairs, while the effects of exponential scaling are difficult to conclusively comment on. Furthermore, after running a statistical test we conclude that most of the found results are statistically significant.

We do however note that these results should perhaps not be taken at face value as we have discussed a number of choices that we made in our methodology that could have had a significant impact. The impact of these choices should be investigated in future work.

6.2 Future Work

Most of our recommendations regarding research that aims to expand upon ours have to do with the various potential flaws that we have mentioned earlier.

First of all, the LM algorithm tries to optimize the MSE by default, so future research should look into analyzing the effect of applying our methodology while curve fitting using a different loss function, possibly the absolute distance. Alternatively, something similar to this can also be achieved by making use of the sigma parameter in the curve_fit function,

which allows the user to attach weights to the fitting data.
Second of all, future work should apply our methodology while sampling from a different distribution. Especially for our failed fits analysis, this could have had a significant impact.
Furthermore, while we only report on the sum of the interpolation and extrapolation, researching in what ways those two metrics are influenced individually by our methodology could provide interesting results as well. Lastly, one metric that we did not analyze but we believe could be interesting is the number of function calls that were done during the curve fitting procedure. A lower amount of function calls for a specific scaling technique could indicate that the curve fitting procedure happens faster.

7 Responsible Research

7.1 Ethical Considerations

Since our research is related to machine learning models and figuring out ways to more accurately predict their learning rates, there are numerous ethical concerns to consider.
First of all, while we ourselves do not intend to do this, machine learning models can potentially be used for nefarious purposes. Considering that we attempt to provide ways to more accurately predict the accuracy of a machine learning model as the provided data set grows, this could result in some parties using our findings to better gauge how much data they should gather for their model. Using this more accurate prediction, they might then be more tempted to acquire more data, data which could in turn be acquired illegally or at the very least without informing the sources they got the data from.

7.2 Reproducibility

In terms of the reproducibility of our research, since our code is publicly available and we have outlined our methodology in detail, we believe that our research is fully reproducible to any party that might be interested in doing so. Additionally, the repository containing our code also contains the datasets from the LCDB database we used. The repository containing the latest version of our code can be found here: https://github.com/Coudenho/Research_Project_Q2

7.3 Use of Large Language Models

The ways in which Large Language Models were used were for code generation in my experiment and for general concept explanation for personal use. This does include code for generating latex tables like the ones we use in this report. In no way were they used to produce any of the text in this report. All the text in this paper that is not directly a part of a table or figure, is a part of the title page, or is referenced to be from an outside source is solely written by the author(s) of this paper.

References

[1] Aized Soofi and Arshad Awan. Classification techniques in machine learning: Applications and issues. *Journal of Basic Applied Sciences*, 13:459–465, 08 2017.

[2] Haoyuan Tan. Machine learning algorithm for classification. *Journal of Physics: Conference Series*, 1994:012016, 08 2021.

[3] Marco Loog. Chapter 5 - supervised classification: Quite a brief overview. In Enrico Camporeale, Simon Wing, and Jay R. Johnson, editors, *Machine Learning Techniques for Space Weather*, pages 113–145. Elsevier, 2018.

[4] Felix Mohr and Jan N. van Rijn. Learning curves for decision making in supervised machine learning – a survey, 2022.

[5] Nazri Mohd Nawi, Abdullah Khan, and M.Z. Rehman. A new levenberg marquardt based back propagation algorithm trained with cuckoo search. *Procedia Technology*, 11:18–23, 2013. 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013.

[6] Tom Viering and Marco Loog. The shape of learning curves: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7799–7819, June 2023.

[7] G. M. Weiss and A. Battistin. Generating well-behaved learning curves : An empirical study. 2014.

[8] Michael M. Li and Lily D. Li. A novel method of curve fitting based on optimized extreme learning machine. *Applied Artificial Intelligence*, 34:849 – 865, 2020.

[9] Tuo-Ran Wang, Ning Liu, Lei Yuan, Ke-Xin Wang, and Xian-Jun Sheng. Iterative least square optimization for the weights of nurbs curve. *Mathematical Problems in Engineering*, 2022, 04 2022.

[10] Sidharth S S. Curve fitting simplified: Exploring the intuitive features of curvpy, 2024.

[11] Felix Mohr, Tom J Viering, Marco Loog, and Jan N van Rijn. Lcdb 1.0: An extensive learning curves database for classification tasks. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, volume 13717 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2022.

[12] Henri P. Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems c ©. 2013.

[13] Muhammad Wahab. Interpolation and extrapolation. 01 2017.

[14] Romain Egele, Felix Mohr, Tom Viering, and Prasanna Balaprakash. The unreasonable effectiveness of early discarding after one epoch in neural network hyperparameter optimization. *Neurocomputing*, 597:127964, 2024.

[15] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew

R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[16] Ana Durango and Craig Refugio. An empirical study on wilcoxon signed rank test, 12 2018.

[17] Sameer Singh. Modeling performance of different classification methods : Deviation from the power law. 2005.