Prevalence of non-monotonicity in learning curves

Dinu Gafton[1]

Supervisors: Tom Viering[1]**, Taylan Turan**[1]

[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 28, 2024

Name of the student: Dinu Gafton
Final project course: CSE3000 Research Project
Thesis committee: Tom Viering, Taylan Turan, Hayley Hung

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Learning curves are useful to determine the amount of data needed for a certain performance. The conventional belief is that increasing the amount of data improves performance. However, recent work challenges this assumption, and shows non-monotonic behaviors of certain learners on certain problems

This paper presents a new approach for detecting non-monotonicity in empirical learning curves. This method monitors the degree of monotonicity violation on non-monotonic intervals, using the performance difference. In addition, the accuracy of the algorithm is being assessed through a series of diverse experiments.

The proposed algorithm is applied to a subset of the extensive Learning Curve Database (LCDB). The results indicate an experimental accuracy of 95.5% in identifying non-monotonicity within real learning curves. Importantly, the metric demonstrated its ability to distinguish genuine non-monotonic trends from minor fluctuations attributed to measurement errors.

1 Introduction

Learning curves graphically depict the relationship between a model's performance and the amount of training data. They have emerged as useful tools for assessing and improving machine learning systems, being used in comparing different algorithms, choosing model parameters during design and determining the amount of data used for training.

The study of learning curves in machine learning often revolves around the basic assumption that model performance consistently improves with additional training data. However, the real-world behavior of learning curves is not always strictly monotonic. In some cases, the relationship between model performance and training data exhibits non-monotonic patterns, introducing complexities that challenge initial expectations. Furthermore, there is limited research on the non-monotonicity of learning curves and on fast, reliable methods of identifying it. Thus, this research paper aims to answer the following question:

How to identify non-monotonic learning curves, and what might influence this?

Socol [1] uses the slopes (and their signs) of the discrete learning curve points to identify non-monotonicity in learning curves, but he fails to answer some insightful questions, for example: *How many times is a curve non-monotone?*, and *How significant is the non-monotonicity?*.

This paper gives a different heuristic method of identifying non-monotonicity, it's significance and the number of occurrences in the learning curve, taking into account the noise in data. Then, using this method, an analysis of the LCDB [2] database will be provided, focusing on identifying the non-monotone learning curves, the significance of each non-monotonicity and how many times non-monotonicity is identified in a single curve.

The paper will follow this structure. Section 2 discusses related work from existing literature. Section 3 introduces the methodology of the research, followed by Section 4 which dives into the experimental setup. In section 6 the experimental results and reflections on what has been concluded from the experimental results are provided. Section 6 draws the conclusions of this work and points to future possible research directions. The final section provides a discussion about the ethical aspects of the research.

2 Related work and literature

Viering et al. [3] highlighted the unresolved nature of the monotonicity in learning curves and the need for exploration and resolution. Despite its benefits, the property of monotonicity may not consistently manifest in learning curves. Learning curves can be defined as smooth, monotonically non-decreasing functions [4], meaning that it is assumed the model's error rate won't rise with additional training samples.

Moreover, Mazur et al. [5] mention that learning occurs most rapidly early in training, with equal increments in performance requiring more and more practice later in training. Unfortunately, this not always the case as this research has found learning curves that experience a decline in performance for bigger training samples until the very end.

Mohr et al. [2] introduced a metric to assess non-monotonicity by examining the maximum violation in an overall descending learning curve. This metric provides important insights for some of the experiments conducted during this research and will be discussed more in-depth further down below.

Mohr et al. also discuss ”peaking” (sample-wise double descent/ascent), a phenomenon often observed in neural networks, characterized by a small region of monotonicity violation within an otherwise monotonic curve. Peaking is considered a specific type of non-monotonic behavior in learning curves, but the metric proposed in this research does not distinguish peakings from other types of non-monotonic behavior, identifying them both if encountered on a learning curve.

Prior research conducted by Socol has provided a new metric for identifying non-mononicity by approximating slopes of the discrete learning curve points [1]. A performance comparison between the algorithm discussed in this paper and the above-mentioned algorithm was conducted as an experiment and the results are provided in the following chapters.

According to Viering et al.'s review [6], many scholars assume learning curves to be monotonic, resulting in limited research on the non-monotonic nature of learning curves. Understanding this behavior may facilitate better extrapolation of training costs for a model and assist in the model selection process [7].

3 Methodology

In this section, the new heuristic to identify non-monotonicity in empirical learning curves is introduced and fully explained.

3.1 Identifying non-monotonicity

The first step in studying the non-monotonicity property of learning curves is to define a metric that classifies them as monotonic. Generally, a learning curve is represented by a collection of discrete points. Moreover, the change of performance on the learning curve can be easily observed by looking at the distance between 2 points on the Y axis, from now on called "performance difference". In case of Accuracy learning curve, an increase in value is expected (the performance difference is positive), thus once a negative difference is found between 2 points, non-monotonicity can be assumed (in case of Error Rate learning curves, the opposite is expected). Hence, the performance differences will be used as metric for this heuristic, specifically their signs. For this purpose, we outline the following algorithm. This metric is formally described in Listing 1.

The algorithm takes as input the list of learning curve points, *points*, and the coefficient used for the threshold, limit (by default it is set at 0.8, the tuning and optimisation of this threshold will be discussed in Section 4.2. Firstly, the performance differences (*Y distances*) for each jump between points is being computed by subtracting the values of 2 sequential points. Then the threshold is computed from the average jump (mean of the absolute values of *Y distances*) and multiplied by the limit coefficient. The purpose of this threshold is to make the algorithm less sensitive to data noise and to prevent their misclassification as non-monotone behavior. After that, the first performance difference of the curve is checked and *increasing* is set at the correct value. The purpose of *increasing* is to store the state of the interval the algorithm iterates over (True - the values are increasing on that interval, False - the values are decreasing). The purpose of *current_interval* is to store the absolute distance of the interval the algorithm iterates over. Then the algorithm iterates over each performance difference, one by one, monitoring any changes in the sign of the performance differences. If a change of sign is noticed, the algorithm checks if the distance of the last interval (current interval) is greater than the threshold and if it was a decreasing interval. If both conditions are met, then that interval is labeled as non-monotonic, thus classifying the learning curve as non-monotone. The algorithm then starts to calculate current interval from scratch for the next interval and continues until the last point.

```
def evaluate_learningcurve(learning_curve, limit):
    threshold = 0

    Y_distances = []
    min = np.min(learning_curve)
    max = np.max(learning_curve)
    learning_curve = np.array(learning_curve)

    for i in range(1, len(learning_curve)):
        Y_distances.append(learning_curve[i] - learning_curve[i-1])
        threshold += np.abs(learning_curve[i] - learning_curve[i-1])

    threshold = limit * threshold / len(learning_curve)
    increasing = True

    current_interval = 0
    occurrences = 0
    significances = []

    if Y_distances[0] < 0:
        increasing = False
        current_interval += np.abs(Y_distances[0])

    non_monotone = False


    for i in range(1, len(Y_distances)):
        if increasing == (Y_distances[i] < 0):
            increasing = not increasing
            if current_interval >= threshold
                and increasing and Y_distances[i-1] < 0:
                occurrences += 1
                significances.append(current_interval/(max - min))
                non_monotone = True
            current_interval = np.abs(Y_distances[i])
        elif i == len(Y_distances) - 1
            and current_interval >= threshold
            and Y_distances[i] < 0:
            occurrences += 1
            significances.append(current_interval/(max - min))
            non_monotone = True
        else:
            current_interval += np.abs(Y_distances[i])

    significance = 0

    if len(significances) > 0:
        significance = np.max(significances)

    return non_monotone, occurrences, significance
```

Code Listing 1: Identifying non-monotonicity using performance differences

4 Experimental Setup

This section details the setup of the four experiments conducted on the introduced heuristic and the general environment in which the studies are conducted.

4.1 Experimental Setup

The research outlined in this paper, along with the tool for assessing non-monotonicity, is coded in Python and executed using Jupyter Notebooks. All experiments are connected to the LCDB API to fetch meta-data related to learning curves, including details such as anchor points, training times, and various plotting utility functions. Additionally, the LCDB utilizes Scikit Learn for classifier implementation and OpenML as source of datasets.

4.2 Experiment 1: Accuracy Analysis

The aim of this experiment is to evaluate the accuracy of the proposed algorithm. For this experiment, real learning curves from the LCDB database will be used. Since LCDB does not include any meta-feature related to non-monotonicity, the plots of the learning curves will be used as the "ground truth" labels. The proposed method will be applied on these learning curves and then their plots will be manually checked to measure the amount of correctly identified non-monotone learning curves. In his work, Socol [1] states that Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and Stochastic Gradient Descent (SGD) are some of the classifiers that display more non-monotone behavior than the rest. Thus, learning curves of the above-mentioned classifiers will be chosen for this experiment. Two equal sized sets of learning curves were created: one was used as a training set to tune the threshold parameter of the algorithm by running the algorithm on this set and then change the value of the threshold based on the results and repeating the same process until the best results were achieved, the second set was used as a test set (the results of this experiment are computed using the latter).

4.3 Experiment 2: Significance Test

Mohr et al. introduced a metric to assess non-monotonicity by examining the maximum violation in an overall descending learning curve (error rate learning curve) [2]:

$$\epsilon_{mono} = \max_{s_i, i<T} \{\max(0, C(s_{i+1}) - C(s_i))\}$$

, where *s* is the anchor point at index *i*, *T* is the highest available anchor index and *C(s)* being the model performance for that certain anchor point.

In simpler terms, the degree of violation of monotonicity is the greatest performance difference that violates monotonicity (i.e greatest negative performance difference for Accuracy learning curves and positive performance difference for Error Rate learning curves). Unfortunately, this metric is not insightful enough to be used as a significance metric as the following example will show.

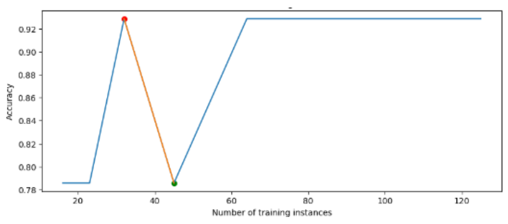

Figure 1: LC with $\epsilon_{mono} \approx 0.15$

In the Figures shown above, both learning curves have a drop of performance of around 15% (0.15). Since these are the biggest drops of performance, they are also considered the degrees of violation of monotonicity for their respective curves. But it can easily be observed that even though the learning curves experience the same drops in performance, the drop of the curve in Figure 1 is more significant than the one of the curve in Figure 2. That is why the metric proposed for the significance of non-monotonicity is the follow:

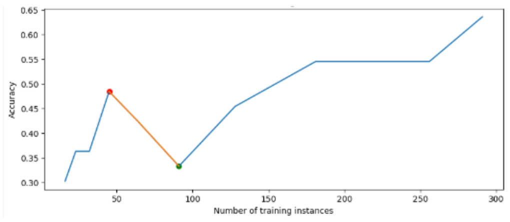

Figure 2: LC with $\epsilon_{mono} \approx 0.15$

$$significance = \frac{current\_interval}{max_{points} - min_{points}}$$

Where max_{points} is the maximum value found on the curve and min_{points} being the minimum respectively. Thus, making the significance relative to the range that covers all the values of the learning curve.
For this experiment, the same set of learning curves from Experiment 1 are being used to test the significance level and number of occurrences.

4.4 Experiment 3: Comparison with other metrics

Codrin Socol's method of identifying non-monotonicity through approximating slopes of the discrete learning curve points [1] is rather intriguing. Thus, the purpose of this Experiment is to compare the performance of the proposed algorithm to the performance of the algorithm proposed by Socol. An important aspect to consider was the time complexity of the slopes approximation algorithm, meaning only a small subset of learning curves was chosen for this experiment, consisting of curvess from the Linear Dis- criminant Analysis (LDA), Passive Agressive Classifier (PAC), Stochastic Gradient Descent (SGD) and Logistic Regression (LR) with a small training time. The experiment will be conducted in the following manner: the same set of learning curves will be run on both algorithms (Socol's algorithm can be found in their public GitHub repository [1], using the same machine, and then a manual check of the learning curves plots will be conducted to measure the accuracy of both algorithms and look for patterns of weaknesses.

4.5 Experiment 4: LCDB meta-analysis

The final Experiment of this research represents a big analysis of the LCDB learning curves using the same methods and metrics that were discussed for Experiment 1 and 2. The experiment will be conducted on big set, consisting of 3000+ learning curves from 15 different learners, irrespective of their training time and datasets.

5 Results and Discussion

In this section, the experimental results are shown and discussed. Then a brief discussion about the interpretation of the results is brought at the end.

[1]https://github.com/CodrinSocol/cse3000-research-project

5.1 Experiment 1: Results

	Actual non-monotonic	Actual monotonic
Classified non-monotonic	526 (99.05%)	23 (24.21%)
Classified monotonic	5 (0.95%)	72 (75.79%)

Table 1: Accuracy Test Results. The brackets describe the percentage of correctly classified curves from the total number of monotonic or non-monotonic learning curves, respectively.

Table 1 describes the results from the accuracy test experiment. The introduced method has correctly identified non-monotonic learning curves with a high accuracy of **99.05%** and monotonic curves with **75.79%** accuracy. The experiment outlines the performance of the metric and represents a good indication that it has the potential of correctly classifying real learning curves. Since there are no studies on how to differentiate between data noise and actual non-monotone intervals on the learning curve, it was challenging to verify the plots and make decisions on whether a learning curve was correctly identified as non-monotone or should have actually been classified as monotone. Thus, only the learning curves with negligible drops in performance were counted as false negatives (curves that were wrongly classified as non-monotone). Examples of false negatives will be provided in Section 5.5.

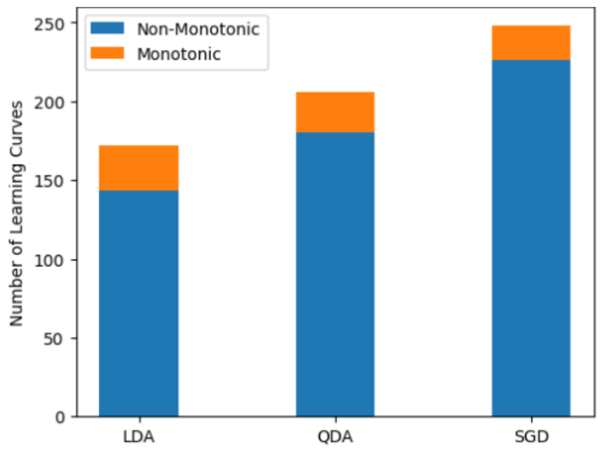

Figure 3: Monotonicity evaluation of 626 learning curves, based on 3 different classifiers. SGD classifier displays the most non-monotone behavior.

Futhermore, Figure 3 shows the distribution of non-monotone and monotone learning curves for each classifier used in the experiment. It can be observed that the Stochastic Gradient Descent (SGD) classifier has the biggest ratio of non-monotone / total nr. of curves.

During the experiment, **172** learning curves had non-monotone intervals identified at the very start, the curve shown in Figure 4 being one example. Also, **83** had non-monotone intervals identified right at the very end, the curve shown in Figure 5 being one example (keep in mind that only the first interval is highlighted, but the other ones on it are also identified).

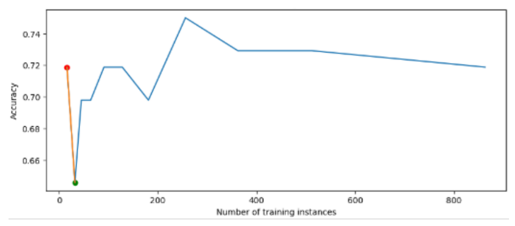

Figure 4: Plot of a non-monotone learning curve, the highlighted orange interval is the first non-monotone interval indentified on the curve

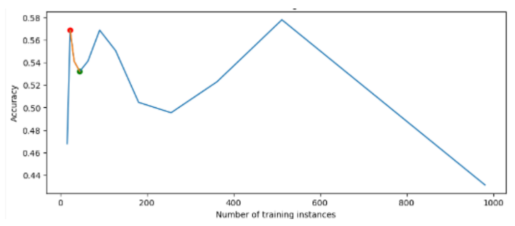

Figure 5: Plot of a non-monotone learning curve with a drop in performance at the very end.

5.2 Experiment 2: Results

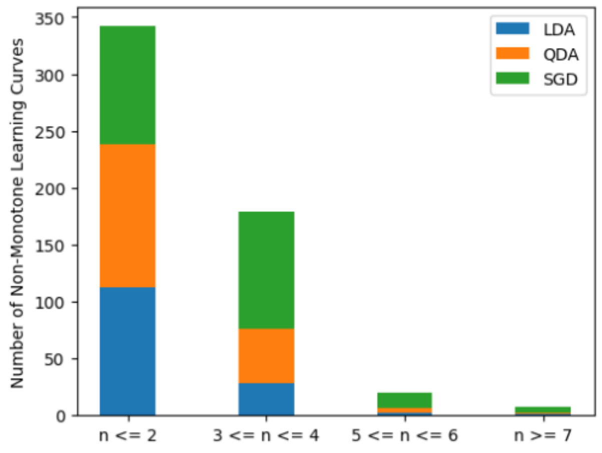

Figure 6: Occurrences of non-monotonicity in learning curves, divided in intervals. The majority have at most 2 occurrences.

In Figure 6, the number of learning curves are displayed based on the number of occurrences of non-monotone intervals found on their respective curve, they are divided based on that number (n in the Figure representing the nr. of occurrences). It can be easily seen that the majority of non-monotone learning curves encounter only one or at most 2 drops in performance on the entire curve, the number of learning curves with more nr. of occurrences than 2 decreasing exponentially. Figure 6 also shows that the Stochastic Gradient Descent (SGD) learner has the most volatile learning curves (displays the behavior of peaking the most).

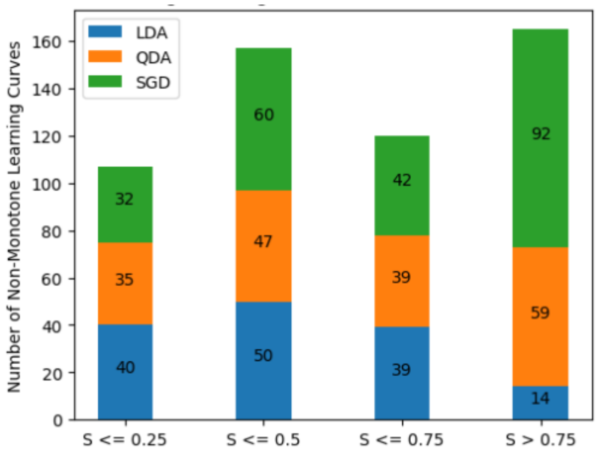

Figure 7: Significance levels of non-monotonicity in learning curves, divided in intervals. Only curves identified as non-monotone were taken under consideration.

Figure 7 displays the number of non-monotone learning curves based on their significance level found on their respective curve, the curves are divided based on 4 equally spaced significance intervals (with S in the Figure representing the significance level). The significance levels are more or less evenly distributed across all intervals. This may indicate that there is little to no correlation between the nr. of occurrences and significance for their respective learning curve. One thing to note is that the number of SGD learning curves increases with the level of significance and there is a big number of SGD learning curves with a huge drop in performance (S >0.75), supporting the idea that SGD displays the most "volatile" behavior.

5.3 Experiment 3: Results

	Actual non-monotonic	Actual monotonic
Classified non-monotonic	135 (100%)	8 (80%)
Classified monotonic	0 (0%)	2 (20%)

Table 2: Accuracy Test Results of Socol's algorithm. The brackets describe the percentage of correctly classified curves from the total number of monotonic or non-monotonic learning curves, respectively.

	Actual non-monotonic	Actual monotonic
Classified non-monotonic	135 (100%)	3 (30%)
Classified monotonic	0 (0%)	7 (70%)

Table 3: Accuracy Test Results of the proposed algorithm. The brackets describe the percentage of correctly classified curves from the total number of monotonic or non-monotonic learning curves, respectively.

Table 2 and Table 3 show the accuracy results of Socol's algorithm and the proposed algorithm in this research respectively. Based on these results, it can be concluded that both algorithms performed well, both being able to correctly identify **100%** of the non-monotone intervals. The main difference resides in the identification of monotone learning curves, Socol's algorithm having a **20%** accuracy compared to **70%**. Socol's algorithm had an average run-time of 20 minutes per learning curve with the proposed algorithm having an average run-time of 2.5 seconds.

5.4 Experiment 4: Results

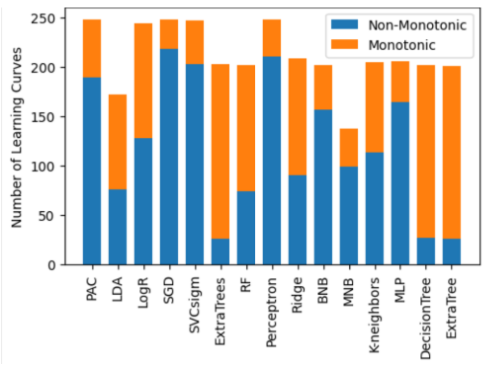

Figure 8: Monotonicity evaluation of 3000+ learning curves, based on 15 different classifiers

Figure 8 shows the distribution of non-monotone and monotone learning curves for 15 different classifiers. It can be seen that Stochastic Gradient Descent (SGD), the sigmoid Support Vector Classification (SVCsigm) and neural network models (such as Perceptron or MLP) show the biggest ratio of non-monotone learning curves. Tree-like learners, such as the Extra Trees Classifier, the Random Forest Classifier and the Decision Tree, show significantly less non-monotonic behaviour than the other learners studied in this experiment. These results may indicate that non-monotonicity can be a property of learners.

5.5 Discussion

Section 3.1 introduced a new algorithm of identifying non-monotonicity in learning curves by tracking the change of signs in performance differences and measuring the distances on the Y axis.
Experiment 1 showed the effectiveness of the proposed

algorithm, correctly identifying most of the non-monotone learning curves. However, a big part of the monotone learning curves were wrongly identified as non-monotone. This might be caused by having a threshold set too low, making the algorithm sensitive to data noise. The learning curve in Figure 9 serves as an example of such case and Figure 10 serves as an example of learning curve wrongly identified as monotone.

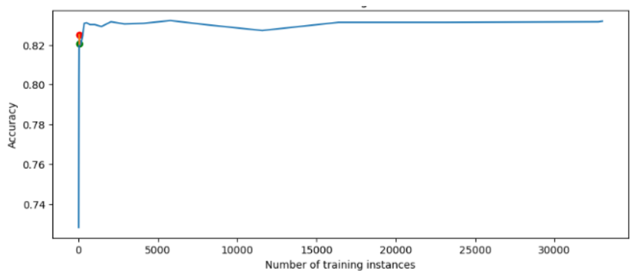

Figure 9: Plot of learning curve wrongly identified as non-monotone on the highlighted interval

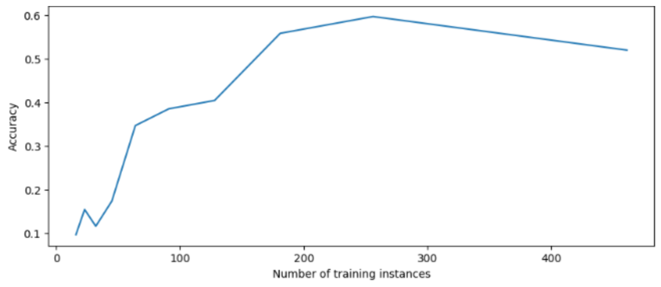

Figure 10: Plot of learning curve wrongly identified as monotone

Experiment 2 showed how the number of occurrences and the significances are distributed among the same set of learning curves used in Experiment 1. The experiment showed that SGD classifier encounters the most significant drops in performance and also the biggest number of occurrences, indicating that SGD might display the most ”volatile” behavior. This conclusion is also supported by Socol's results. To gain more insight in the results of the experiment, significances and number of occurrences can be plotted against each other so that their correlation can be found.

Experiment 3 showed a comparison between Socol's algorithm of approximating point slopes and the proposed algorithm. Both algorithms performed well, having high accuracy. However, due to Socol's algorithm being computationally expensive and not being able to evaluate learning curves without enough training samples (initial set was bigger than 145 curves), it was unfeasible to compare the algorithms on a bigger set of learning curves. Thus, the results might not indicate the real performances of the algorithm. Figure 11 and Figure 12 show examples of wrongly classified monotone learning curves by Socol's algorithm and the proposed one respectively. It can be seen that the reason of misclassification is high sensitivity to data noise.

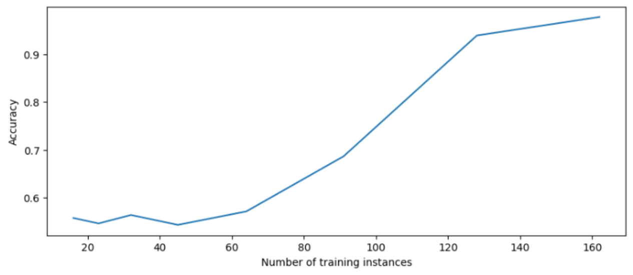

Figure 11: Plot of learning curve wrongly identified as non-monotone by Socol's algorithm

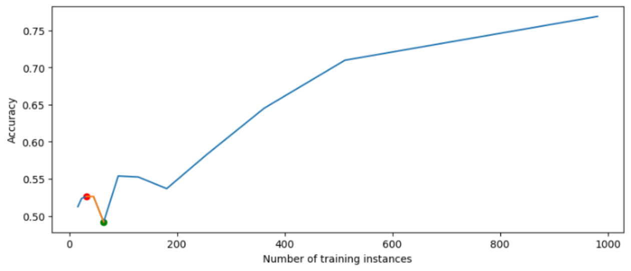

Figure 12: Plot of learning curve wrongly identified as non-monotone by the proposed algorithm

Experiment 4 provided an analysis of a bigger learning curve set of LCDB, showing that SGD, SVC and neural network models display the most non-monotone behavior while tree-like learners displaying the opposite. Both statements are supported by Socol's results. The algorithm also tracks the number of occurrences and significances of learning curves, this data should be analyzed in detail and would provide more insights in the non-monotonic behavior of the learners included in the experiment.

6 Conclusions and Future Work

In conclusion, this paper has addressed the necessity of identifying non-monotonicity in learning curves. To ascertain this characteristic, an algorithm was introduced for assessing the monotonicity of an LC by observing the performance differences and the signs of their values, outlined in Listing 1. Subsequent evaluation in Experiment 1 demonstrated the algorithm's high accuracy, suggesting its suitability for discerning non-monotonic curves. Experiment 2 brought for the first time into discussion the significance and the number of occurrences of non-monotonicity in learning curves, highlighting that there is little to no correlation between them two and also that for the majority of learning curves non-monotonicity is identified only once or twice at most per curve, the number of highly volatile learning curves being very small.

Experiment 3 showed the performance of 2 different methods of identifying non-monotonicity in learning curves. However, due to constraints in time and processing power, only a small

fraction of the LCDB learning curves were used in Experiment 3. It was also highlighted that a tradeoff between speed and accuracy must be taken into consideration when choosing an algorithm for this specific task.
Experiment 4 displayed that the introduced algorithm can investigate huge sets of learning curves in a relative fast time. The results also showed the distribution on non-monotone curves among the most used learners in Machine Learning that can be found in LCDB and the significances of their drops in performance (this results have to be yet added to the paper).
A potential research direction is running the algorithm on the entire LCDB to identify classifiers inherently displaying non-monotonic behavior. Since the complexity of the algorithm is O(N) (where N is the nr. of discrete points on the curve and is memory efficient, the algorithm can be used on the entire LCDB to create a new meta-feature of non-monotonicity and be used in further studies. Optimizing the algorithm's performance is another consideration, possibly by tuning the existing threshold even further or by using the standard deviation instead of the average absolute performance difference will yield better results.

7 Responsible Research

Ensuring the replicability of research findings is an important consideration in the research process. Researchers bear the responsibility of demonstrating the ethical conduct of their studies and maintaining full transparency in their methodologies. In this study, measures were taken to guarantee the reliability of our results through meticulous planning and execution of experiments.
Furthermore, a commitment to reproducibility is evident throughout the research, with transparent reporting reflected in this paper. To facilitate the reproducibility of the experiments, both the experiments and associated data are publicly shared , as well as the source code and plots. These resources are accessible in a GitHub repository [8], where each experiment is documented in a dedicated Jupyter Notebook. The ReadME.md file within the repository provides instructions on installing necessary dependencies and executing the experiments. Following these steps outlined in the Methodology and Experimental Setup sections or running the code in the mentioned repository will yield results consistent with those detailed in Section 5.

References

[1] Codrin Socol. Non-monotonicity in empirical learning curves: Identifying non-monotonicity through slope approximations on discrete points, 2023. http://resolver.tudelft.nl/uuid:3b7f24c8-08a9-4641-be82-38b880ac6898.

[2] Felix Mohr, Tom J. Viering, Marco Loog, and Jan N. van Rijn. Lcdb 1.0: An extensive learning curves database for classification tasks. In Massih-Reza Amini, Stéphane Canu, Asja Fischer, Tias Guns, Petra Kralj Novak, and Grigorios Tsoumakas, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 3–19, Cham, 2023. Springer Nature Switzerland.

[3] Tom Viering, Alexander Mey, and Marco Loog. Open problem: Monotonicity of learning. In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 3198–3201. PMLR, 25–28 Jun 2019.

[4] Gary M Weiss and Alexander Battistin. Generating well-behaved learning curves: An empirical study. In *Proceedings of the International Conference on Data Science (ICDATA)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer ..., 2014.

[5] James Mazur and Reid Hastie. Learning as accumulation: A reexamination of the learning curve. *Psychological Bulletin*, 85:1256–1274, 11 1978.

[6] Tom Julian Viering and Marco Loog. The shape of learning curves: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:7799–7819, 2021.

[7] Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren. Metalearning: Applications to automated machine learning and data mining. *Metalearning*, 2022.

[8] Dinu Gafton. Cse3000-research-project repository. https://github.com/dgafton/cse3000-research-project, 2024.