



# MAST90109 Research Report: Test Time Attack in Learned Index Structure

by

Saransh Srivastava - 1031073

in the

Department of Computer Science and Statistics

Melbourne School of Engineering

**THE UNIVERSITY OF MELBOURNE**

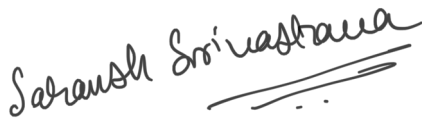
November 2020

# Abstract

Learning models have recently gained popularity at a staggering rate in the commercial industry. They are increasingly becoming part of all mainstream softwares. This includes the recently published work advocating use of learned models in place of traditional data structures in database indexing. Unfortunately, as we take this leap forward in technology with intelligent systems, security and safety of such models are not studied aggressively. While fitting a model on any dataset, the model generalizes the data leading to areas where the model perform poorly. The main contribution of this report is (a) to provide a methodology to detect such poor performing areas in the data (b) an evaluation mechanism to study how this method performs with respect to the brute force mechanism. Together, this system provides a way to better understand the association of the data with the learned index models making it more reliable and commercially viable.

# Declaration

I certify that this report does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text. The report is 4100 words in length (excluding text in images, tables, bibliographies and appendices).



---

Saransh Srivastava



---

Date

# Acknowledgement

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to thank my supervisors, Associate Professor Ben Rubinstein and Assistant Professor Renata Borovica-Gajic, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to acknowledge my colleagues from my Capstone Research Project at the University of Melbourne for their wonderful collaboration. In addition, I would like to thank my brothers and parents for their wise counsel and sympathetic ear. Finally, I could not have completed this dissertation without the support of my friends, Mansimrat Singh Badhesha, Waqar Ul Islam and Carlos Andres Davalos, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Declaration</b>	<b>ii</b>
<b>Acknowledgement</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>3</b>
<b>3 Problem statement</b>	<b>4</b>
<b>4 Methodology</b>	<b>6</b>
4.1 Motivation . . . . .	7
4.2 Change point detection - CUSUM . . . . .	9
4.2.1 CUSUM on adjacent key differences . . . . .	10
4.3 Evaluation Measure . . . . .	10
<b>5 Experiments</b>	<b>12</b>
5.1 Datasets . . . . .	12
5.2 Anomaly Detection . . . . .	12
5.3 2 way CUSUM . . . . .	14
5.4 Analysis . . . . .	15
<b>6 Conclusion</b>	<b>17</b>
6.1 Future work . . . . .	17
<b>Bibliography</b>	<b>19</b>

# Chapter 1

## Introduction

Database indexing is one of the corner stones of the introduction of computer softwares into mainstream commercial industry [1] [2]. Traditional data structures such as B-trees have always been the primary choice for underlying indexing model [3] for databases but recent works by researchers [4] [5] [6] have paved the way towards the introduction of machine learning models in database indexing. The data structures have always been agnostic to the properties of the keys used for indexing. The B+ tree can be considered as a model which takes in an input key from a sorted array and predicts an index where the key and its corresponding data entry are stored [4]. Further, the authors argue that the mapping from keys to their respective indexes can be described as the cumulative distribution function [7]. Learning models can understand patterns of the keys to predict their location in the sorted array. As proved in their paper, not only are such models tested to be efficient in space but also comparable in accuracy. This introduces a very interesting inter-disciplinary field of research questions including re-imagining addition, deletion, range and point queries and existence queries. However, traditional database systems are very well understood, tuned and researched concepts over many years [8]. So in order to replace such systems commercially would require machine learning models to prove their resilience and safety.

Cyber security is as old a topic as computer engineering itself and has parallelly evolved over the course of decades [9]. As more and more machine learning models are being introduced in real world software systems, their safety and security has become an evolving matter of concern. With the exponential rise of machine learning models, adversarial example analysis has become a recent topic of interest in the research community. While a lot of work is being done in trying to fool machine learning models in computer vision

[10] [11], to the best of our knowledge nobody is presently looking at attack scenarios in learned index models. In this report, we will explore the properties of keys in learned index scenarios and exploit them for evasion attacks analysis. As machine learning models tries to learn the behaviour of the underlying CDFs, we will explore how the actual data and CDF varies at different points. Further, we will introduce a methodology of detecting such points in the dataset which may be independent of model we use for index prediction. Drawing our inspiration from adversarial machine learning, we will formalize the attacks and address how much data knowledge is required and quantify the relevance of the extracted data points. We will explain why information retrieval evaluation metrics will not be sufficient in capturing the amount of attack on such systems and then introduce novel metric to describe our results.

Section 2 presents some of the recent related work in learned models, its application in databases and adversarial machine learning. As both fields have been evolving parallelly, it is interesting to see how different ideas can be related with each other. Section 3 introduces our problem statement. In section 4, we explore the properties of the keys as our motivation for this analysis. In this section, we will also introduce a statistical tool to detect points of concern. The evaluation metrics for our methodology is also explained in this section. Section 5 explains in detail our experimentation along with tabular comparison of the two methods introduced. We provide our analysis of the methodology in this section. In the last section (section 6 ), we present our conclusions of the report and future directions.

# Chapter 2

## Related Work

Learned index structures gained a lot of popularity in the research community and rightly so, because it gave a fresh perspective to decades old technology using very well understood and research data structures. However, databases have many interesting challenges ranging from insertions, deletions and updates. Each problem comes with its own set of challenges, for example [12] introduces how learned index models can be modified to include adaptive updates. Similarly, [5] explains how splines recreate the underlying cumulative distribution functions of the keys using a combination of array points, splines and interpolations. All these domains are interesting for us, because we wish to analyze our methodology and the behavior of learning index structures.

Commercialization of machine learning models requires them to be resilient to attacks. In adversarial machine learning study, researches found that introduction of very small but specific noise in the test data leads to huge misclassifications [13] in not only statistical models but heavily in deep neural networks [10] [14]. Specifically in computer vision [11], noise addition not visible to human eye incurred large classification errors and raised serious concerns for such models. In the case of train time attack, one way to introduce noise is to find small changes and store it in the data, such that it makes the average classification error high. To achieve this is by taking the weights of the training function  $\omega_{Xy}$  such that the predictions are  $\omega_{Xy} \cdot X_{test}$ . Comparing these predictions with  $y_{test}$  L2 loss and differentiating w.r.t  $X_{test}$  would tell us how to change  $X_{train}$  a little bit to make loss on queries worse. Additionally, in [15] the authors observe certain key properties of the data and develop a statistical-based poisoning attack which requires very little information of the actual model. Our work will also be in this direction, but address the problem and its solution in a different way.



# Chapter 3

## Problem statement

One of the primary reasons of databases being used in the industry so extensively today is due to the fact that they are researched and understood so well. Replacing core functionality in such systems and to make it commercially viable will require same quality standards. In our research, we will explore the vulnerabilities associated with machine learning models in database setting. Our primary aim is to understand how the learning models interact with underlying dataset - in which regions do they represent the data well and in which they represent it poorly. We believe it is important to analyze worst case performance of the models rather than just the average error. This will help developers and researchers to examine their own model while claiming their efficient usability. We will thus identify input keys which will lead to maximum accuracy errors in the system. These keys will act as surrogate to test time attack. We wish to extract a subset of worse performing keys rather than just the worst one.

Systems attacks are usually of two types. Train time attacks and test time attacks. A train time attack happens when attackers get hold of a small subset of training data and injects changes into it making the model sensitive to certain behavior. When such a model is deployed in production system, the system becomes vulnerable to certain specific attack due to the "*poisoning*" of the data. Such attacks are very popular in computer vision system. A test-time attack occurs when the attacker has no or some knowledge of the system and deliberately presents it with certain inputs that the model misclassify them. In [16] the authors shows how properties of neural networks makes them vulnerable to such perturbations.

The purpose of the research project is to introduce the idea of adversarial thinking in learned index model, so that the research community starts to optimize solutions keeping

---

security aspect in mind. We believe this will not only make machine learning systems more trustworthy but will potentially have more applications in the domain. In the next section, we will describe our methodology of detecting such worst performing keys.

# Chapter 4

## Methodology

We formulate our solution from the adversarial perturbation point of view. Realizing the limitation of resources and time constraints in this report, we will concentrate on database systems which do not stream data continuously. Rather, the database writes a dataset into its system completely first and then queries are made to retrieve indexes corresponding to particular keys (range or point). As in traditional databases, we will assume that the keys are in sorted order. To further keep our analysis simple, we will assume that there is no deletion and/or addition of keys. The system can however, update in batches. A real world example of such kind of system can be considered in a university registration setting, where each year students are given a roll number and register their information like name, address, contact information in the database. They are given a student card in return and each time they need to enter a secure facility they need to use that ID card to gain access. The finite set of students information is entered once and then multiple queries are made to the database throughout the year.

It is interesting to note here that the case of learned index is unique from other machine learning models. This is because in a database, queries are usually made for keys that are already present in the system. So the prediction of the machine learning model is actually prediction on known data used during training rather than a new one. This characteristic of the problem makes the adversarial machine learning problem even more interesting. In essence, we need to understand which keys will be worst performing in the training set. Our case is limited to test time attacks. so we do not wish to manipulate a subset of training set rather analyse the behaviour of the training keyset and understand which keys subset are potential adversaries. In [4], the authors, explains how the mapping between

sorted keys and their indexing can be assumed as a cumulative distribution function. We will now exploit some properties of such CDF.

## 4.1 Motivation

As in the case of traditional databases, the keys of the dataset needs to be ordered for indexing. This not only allows the B+ trees to return range queries but also make them time and computationally efficient. The keys being ordered will thus always be monotonically increasing and belong to the 1<sup>st</sup> quadrant always below the 45° line where y axis is the index and x is the sorted keys in the data. Another observation here is that the gradient of the CDF will be directly proportional to the density of the keys in the dataset. Regions in the dataset where keys are more dense will have a larger gradient and vice-versa.

Treating this as a regression problem, we aim to detect those keys which are producing maximum residuals. In our example, this means the attacker would like to access a set of student IDs which always give large accuracy errors in the system thus making it extremely slow to retrieve, to the point where it delays response significantly. Knowing such vulnerabilities to the system can help system architects and administrators to handle such problem well in advance.

From figure 2, we see that the maximum residual producing keys are at the corners of change in density. Based on this observation, we defined our methodology by evaluating CUSUM quality control charts on the adjacent key differences. Differences between adjacent keys will change abruptly near the corners and the change point detection discussed in section 4.2 will be instrumental in detecting those points.

For the ease of inference and as a starting analysis, we will use a linear polynomial regression fitting on the datasets. Figure 1 shows well dispersed keys and the model fitted on it. The model desires to overfit the data so that it predicts the index of the key as accurately as possible. In the next figure (Figure 2) we zoom into the rectangular region and observe some of the points more closely where fitting is not well. It is evident that when the density of keys distribution changes abruptly, there is an *elbow* creation in the plot. The fitted model clearly performs bad at such *elbows*. This is because the model fitting is trying to recreate a continuous distribution function of the keys and any abrupt change at the corners are smoothed by the CDF. The more abrupt the change

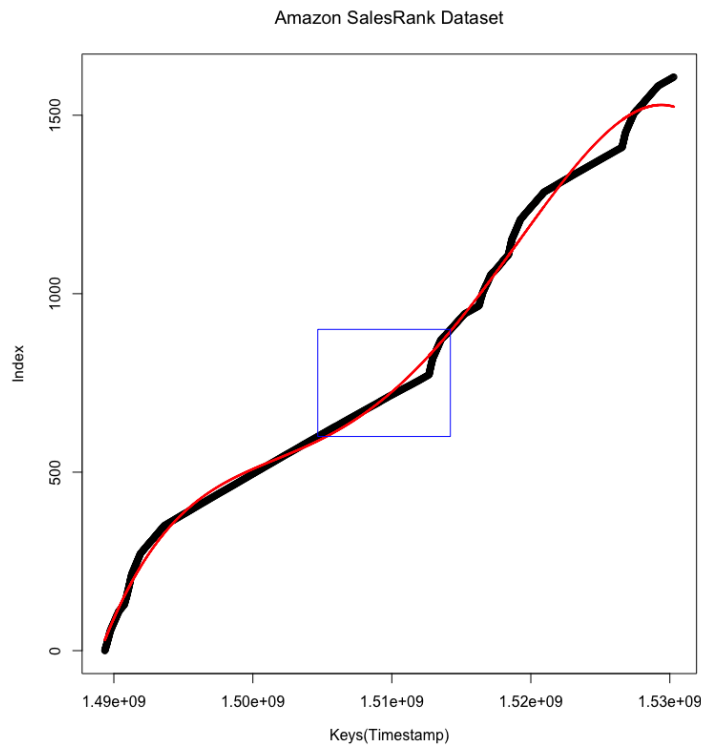


FIGURE 1: Model Fitting

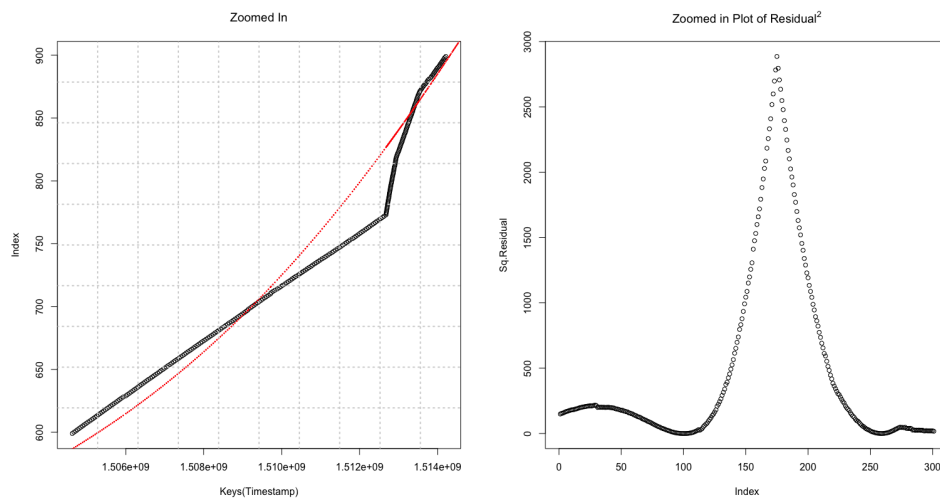


FIGURE 2: Zoomed in view of a selected region

is, the steeper the *elbow* and the more possibility of producing large residuals from the fitted model.

It is important to observe here that because the learned models are essentially trying to empirically model the continuous cumulative distribution function, we may choose to worry less which specific model we are fitting. Rather we concentrate on which points

the dataset falls far away from the CDF. The naïve brute force method to detect such points would involve traversing the entire dataset and comparing the raw residuals with the fitted model and then sorting them. This is computationally very expensive. In the next section, we will introduce a well known statistical method of change point detection. We will use the understanding from this method for the identification of corner points.

## 4.2 Change point detection - CUSUM

We will briefly review the general concepts and application of CUSUM algorithm for change point detection here. A change point detection is based on the hypothesis testing. Let there be a sequence of random variables  $(X_1, \dots, X_n)$  defined over a measurable space  $(\Omega, \mathfrak{F})$ . The sequence is used to evaluate the likelihood that the new samples have a distribution  $Q_1$  rather than the original  $Q_0$ . The likelihood ratio is denoted by  $L(x) = dQ_1/dQ_0(x)$  and the CUSUM process is defined by:

$$Z_0 = 0, \quad Z_k = \max(0, Z_{k-1} + \log L(X_k)).$$

When the samples are very likely coming from  $Q_0$  rather than  $Q_1$ , the value of likelihood coming from  $Q_1$  is "small" and the logarithm will be mostly negative, keeping the CUSUM process zero. However, when the samples are more likely to come from  $Q_1$ , the likelihood is larger than 1 and as  $k > \tau$ , a change point is detected. Here  $\tau$  represents a random variable in the space  $(\Omega, \mathfrak{F})$ .

A threshold  $h$  is defined as the value above which the sequential process is assumed to come from  $Q_1$  rather than  $Q_0$  and the time of occurrence is defined as:

$$T = \min\{k \geq 0 : Z_k \geq h\},$$

In particular, when the observations are assumed to come from a normal distribution, the algorithm takes the following form.

$$Z_0 = 0, \quad Z_k = \max\left(0, Z_{k-1} + (\mu_1 - \mu_0)X_k - \frac{1}{2}(\mu_1^2 - \mu_0^2)\right).$$

Further, the same algorithm can be applied while traversing the dataset in reverse. This is possible when the dataset is available in advance. Thus applying CUSUM while going forward on the data points  $(X_1 \dots X_N)$  will yield a change point detection  $T^{(1)}$ , whereas

reversing the data points ( $X_N \dots X_1$ ) will detect a different point of change  $T^{(2)}$ . Averaging the two points in the dataset will predict a closer location to the actual corner of the abrupt change in density of the keys.

### 4.2.1 CUSUM on adjacent key differences

The algorithm defined in section 4.2 assumes that the mean of both  $Q_0$  and  $Q_1$  processes are known in advance but in the case of learned index, we know only an estimate of  $Q_0$ , the underlying distribution mean using the samples available to us. This can either be calculated using the entire dataset available (for better accuracy) but can also be obtained by sub-sampling the dataset. We clearly do not know the distribution of  $Q_1$ , so we now use a more simplified version of the CUSUM algorithm originally proposed by Page [17] and extensively used in quality control charts [18]. The process evolves as follows:

$$\begin{aligned} Z_{n+1}^+ &= \max(0, Z_n^+ + X_{n+1} - (\mu_0 + K)) \\ Z_{n+1}^- &= \max(0, Z_n^- - X_{n+1} + (\mu_0 - K)), \end{aligned} \tag{1}$$

where the starting values are  $Z_0^+ = Z_0^- = 0$ . The parameter  $\mu_0$  is the original process mean and  $K$  is called the reference value (or the slack value). Again the rejection region for the hypothesis is defined by a threshold  $h$ , and the stopping time

$$T = \min\{k \geq 0 : Z_k \geq h\},$$

is declared as the estimated change point. In [18] the general control chart practice suggested is to choose  $K = \frac{\mu_0}{2}$  and  $h = c\sigma$ , where  $c > 4$ . We manipulate the factor  $c$  to suit our case. The value of  $c$  will be a contributing factor in the quantity of datapoints the algorithm visits. Assuming normality of the adjacent key differences over the entire dataset, we fit a normal distribution on the training data and estimate sample mean and sample standard deviation as stated in Table 1. Using these CUSUM parameter estimates we draw our control charts and threshold.

## 4.3 Evaluation Measure

The CUSUM algorithm of section 4.2.1 will be able to highlight portions of the datapoints which are our regions of concern. We now need a metric to be able to measure the

<b>Dataset</b>	$\mu_0$	$\sigma$	$c$
Time-1608	25471.06	16302.26	5
Time-2730	10740.64	7383.74	6
Time-5257	6286.30	2901.14	4
UID <sub>A</sub> – 0.4M	2.89	2.41	5
UID <sub>B</sub> – 0.4M	4.67	2.21	5
UID <sub>Train</sub> – 1M	1.7	0.73	4

TABLE 1: Fprward CUSUM parameters in different datasets

performance of our system. Traditional information retrieval systems evaluation measures are used to assess the quality of retrieval results based on the search query [19]. The measuring criterion are formulated using two properties of the result, namely, how many relevant documents are obtained from the result and in which order or rank they are shown to the user. Although, the relevance and rank hold significance in our test time attack in learned index model scenario, we do not bother about the order of retrieval. In our system, rank corresponds to the adverse relevance or affect the particular key has on the system. So the key producing maximum residual to the fitted model will be ranked highest in our methodology and those producing no residuals will be ranked bottom most. This clearly cannot be justified using the traditional IR evaluation measure. We now define two of our evaluation measures.

The first criteria for evaluation is the number of keys visited by our methodology representing the "quantity". This will give us a sense of how much computation we saved by not traversing the entire datapoints. We wish to obtain as many worst performing keys as possible while visiting as less keys as possible.

$$\%Visited = \frac{\text{Number of keys visited}}{\text{Total Number of keys in the dataset}} * 100$$

With the amount of data visited, we also need to identify the "quality" of data we are extracting. We evaluate this based on the rank of the keys.

$$\%Relevance = \frac{\sum_{found}(\frac{1}{Rank})}{\sum_{attack}(\frac{1}{Rank})} * 100, \quad (2)$$

where  $Rank \propto Residual$



# Chapter 5

## Experiments

### 5.1 Datasets

For experimentation, we used two types of datasets . The first is a subset of the Amazon sales rank dataset freely available on Kaggle [20]. We used three different subsets represented by  $Time - \#rows$ . The complete dataset was used by the authors of Learned Index Structures for benchmarking [21] and we used a subset of it. This dataset represents the normalized timestamp of points in time when a particular author’s book was sold on Amazon website and the corresponding rank of the day for that author based on the sales of their book(s). For database indexing, we are interested in the timestamp as keys in this dataset. The other dataset used for keys were UID entries of Advertisement CTR competition freely available online [22] with nearly 0.4 million entries. We used the two test datasets provided in this competition and their user ID as the key. The experiments were performed on an Apple laptop running MacOS Catalina in R Studio using 1.1.463v with R version 4.0.2 (2020-06-22) on the platform x86\_64-apple-darwin17.0. The links to the entire code generated is available here [23].

### 5.2 Anomaly Detection

Using these datasets, we ran our feed forward CUSUM (Eq. 1) on adjacent key differences to detect anomaly points. The parameters of the CUSUM are defined empirically by observing the adjacent key difference values as shows in figure 4. Whenever  $Z^+$  or  $Z^-$  crosses the threshold, the entry and exit points are collected and keys in close proximity

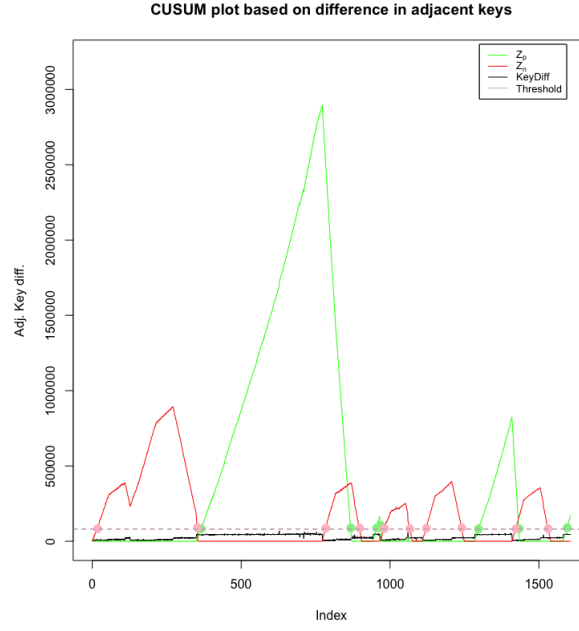


FIGURE 3: Forward CUSUM plot on adjacent key differences

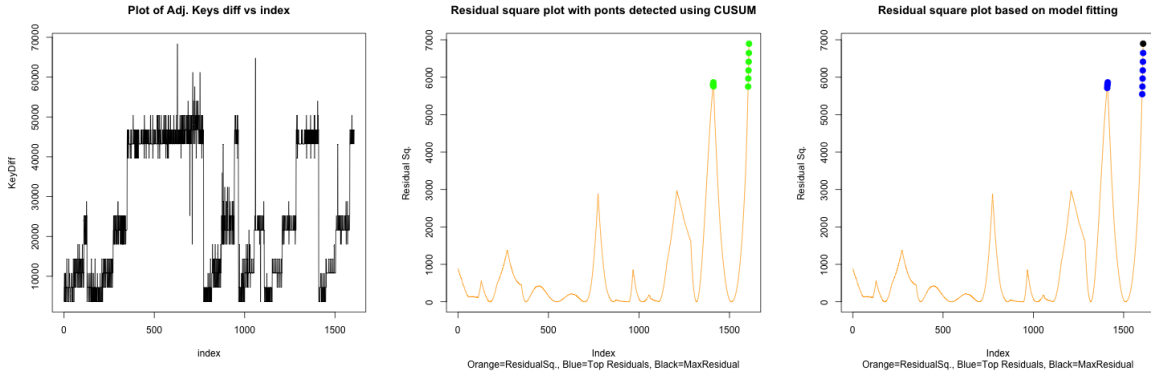


FIGURE 4: Left figure shows the adjacent key differences. Middle figure shows the top 10 points detected by feed forward CUSUM algorithm. Right figure shows the actual top 10 maximum residual producing keys based on model fitting

of these points are analyzed. These points are expected to produce large residuals on learned index models.

As discussed before, for evaluation we fit a polynomial linear model on the entire dataset and look for maximum residual producing points. These points are then ranked based on the amount of residuals produced by them to the fitted model. The largest residual producing point(s) are top ranking and compared with extracted points by our method. Table 2 shows the details of our analysis on the two types of data having in total 5 datasets of different sizes. The evaluation metric defined in section 4.3 is used here.

#Data	#Attack	#Found	%Visited	%Relevance
Time-1600	10	10	42.3	100.0
Time-1600	30	23	42.3	93.2
Time-1600	90	58	42.3	85.2
Time-2730	36	22	70.3	70.1
Time-2730	192	109	70.3	66.3
Time-5257	609	181	41.8	62.7
UID <sub>A</sub> – 0.4M	1620	406	54.5	54.7
UID <sub>B</sub> – 0.4M	6940	2138	74.9	69.1

TABLE 2: Analysis of test time attack points detection in Learned Index Model datasets

### 5.3 2 way CUSUM

Original CUSUM algorithm was designed for change point detection in time series data. However, when the entire dataset is available at hand, we can apply CUSUM in the reverse direction as described in section 4.2. This lets us reduce the number of candidates by nearly half improving both the quality of obtaining corner points as well as reducing the quantity of data points required to visit. We store in forward and reverse direction, the point at which a change is detected and average their value. Then we search in close proximity to obtain the number of datapoints required to visit, as before. Table 3 shows the results of 2 way CUSUM analysis obtained using grid search for hyperparameter tuning and figure 5 shows the forward and backward CUSUM plots on Time-1600 when we were looking for worst 30 performing keys

#Data	#Attack	#Found	%Visited	%Relevance
Time-1600	10	10	33.6	100
Time-1600	30	28	44.8	98.2
Time-1600	90	82	44.8	95.9
Time-2730	36	29	58.6	89.6
Time-2730	192	156	58.6	83.6
Time-5257	609	487	60.8	94.4
UID <sub>A</sub> – 0.4M	1620	551	36.7	72.7
UID <sub>B</sub> – 0.4M	6940	2904	59.0	50.4

TABLE 3: Forward and backward CUSUM Analysis of test time attack points detection in Learned Index Model datasets

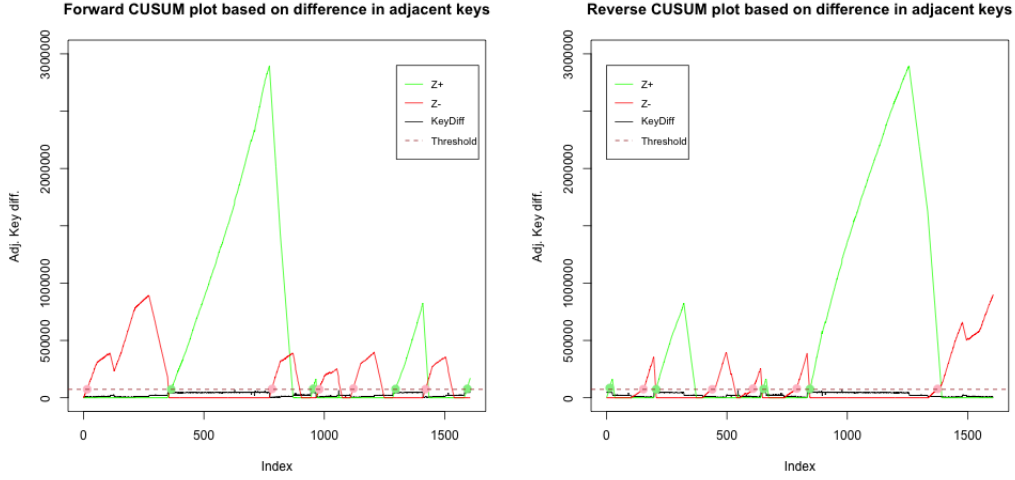
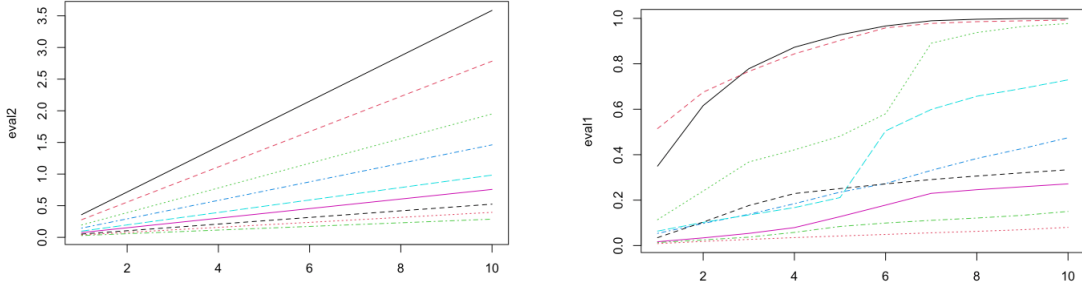


FIGURE 5: Forward and reverse CUSUM plot on adjacent key differences

FIGURE 6: Grid Search evaluation of % visited and % relevance for dataset  $UID_B$ 

## 5.4 Analysis

Our methodology, both feed forward CUSUM and two way CUSUM performs well in the detection of worst performing data points. For example, in the case of Amazon dataset *Time* – 2730 having 2730 rows, using feed forward CUSUM, out of the top 192 worst performing keys, 109 were detected by the method. The algorithm traversed 70.3% of the dataset and collected 66% of relevant points. Similarly, when two way CUSUM was performed on the same dataset, 156 out of 192 were extracted by traversing only 58.6% of the dataset with a 83.6% relevance. The higher value of relevance signifies that points ranked higher were extracted here. The two way CUSUM is evidently performing much better than the feed forward method, both in high number of relevant point extraction and low number of points required to visit. This is understandable because 2 way CUSUM narrows down the corner point from both direction, thus giving more accurate estimation. The algorithm can be further fine tuned using more detailed grid search (figure 6) and other machine learning parameter tuning methodology.

However, there are certain objectives this methodology cannot fulfil. Firstly, it cannot extract worst performing data points in a decreasing order, rather it extracts a subset of them. The peaks of the CUSUM variables do not corresponds to the amount of residuals they produce. It detects how abruptly there is a change of key density - the more abrupt the change, the higher the peak. The method also suffers from late change point detection, which means that it detects a change only after that change has occurred in the past.

# Chapter 6

## Conclusion

Artificial intelligence is the new electricity of the 21<sup>th</sup> century [24] but to make it truly industrial, it needs same level of guarantees as traditional commercial software systems provide. We have shown that using our methodology, we can analyze the datasets during development of learned index models and detect regions of worst performance. Our methodology is using a well understood statistical tool to analyze datasets and detect regions where learned models will be expected to perform poorly. We have explained why traditional IR evaluation metrics are not useful here and introduced novel measures for adversarial perturbation evaluations. Further, we explained the shortcoming of our devised methodology and what it cannot do. Lastly, we kept the scope of the problem exploration simple and intuitive so that we can draw inferences from it and perhaps motivate fellow researchers towards thinking about adversarial machine learning problem while developing their systems.

### 6.1 Future work

The idea of replacing machine learning model in database indexing has opened a fresh and exciting new direction in interdisciplinary computer science. In the future, we would like to explore behavior of non-linear neural networks when under perturbation attack. Neural networks have shown tremendous success in commercial softwares but have also been under criticism for high classification error under test time attack. We wish to explore more types of datasets including cases of sparse sets to understand where learned index models fail to perform well.

---

We would also like to formulate training time poisoning of the datasets and develop interesting research questions. Ideally, when changing training dataset, we should change the predictions to stay consistent. This is obviously not trivial and would require more innovative ways of solving the problem. Our eventual goal by understanding these adversarial attacks is to develop a defence algorithm in parallel to learned models so that such models can become more resilient to attacks and commercially successful.

# Bibliography

- [1] Kevin Driscoll. From punched cards to” big data”: A social history of database populism. *communication+* 1, 1(1):1–33, 2012.
- [2] Avi Silberschatz, Michael Stonebraker, and Jeff Ullman. Database systems: Achievements and opportunities. *Communications of the ACM*, 34(10):110–120, 1991.
- [3] Douglas Comer. Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137, 1979.
- [4] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pages 489–504, 2018.
- [5] Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. Radixspline: a single-pass learned index. *arXiv preprint arXiv:2004.14541*, 2020.
- [6] Naufal Fikri Setiawan, Benjamin IP Rubinstein, and Renata Borovica-Gajic. Function interpolation for learned index structures. In *Australasian Database Conference*, pages 68–80. Springer, 2020.
- [7] V. Leis P. Boncz, T. Neumann. The case for b-tree index structures, 2017. <http://databasearchitects.blogspot.com/2017/12/the-case-for-b-tree-index-structures.html>.
- [8] Elisa Bertino and Ravi Sandhu. Database security-concepts, approaches, and challenges. *IEEE Transactions on Dependable and secure computing*, 2(1):2–19, 2005.
- [9] Julian Jang-Jaccard and Surya Nepal. A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5):973–993, 2014.



- [10] Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In *Advances in Neural Information Processing Systems*, pages 3910–3920, 2018.
- [11] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.
- [12] Jialin Ding, Umar Farooq Minhas, Jia Yu, Chi Wang, Jaeyoung Do, Yinan Li, Hantian Zhang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, et al. Alex: an updatable adaptive learned index. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 969–984, 2020.
- [13] Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. Is deep learning safe for robot vision? adversarial examples against the icub humanoid. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 751–759, 2017.
- [14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [15] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.
- [16] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [17] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.
- [18] Douglas C Montgomery. *Introduction to statistical quality control*. Wiley, 2020.
- [19] Evaluation measure (information retrieval). [https://en.wikipedia.org/wiki/Evaluation\\_measures\\_\(information\\_retrieval\)#Online\\_metrics](https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Online_metrics).
- [20] Amazon sales rank data for print and kindle books. <https://www.kaggle.com/ucffool/amazon-sales-rank-data-for-print-and-kindle-books>.

- 
- [21] Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. Sosd: A benchmark for learned indexes. *arXiv preprint arXiv:1911.13014*, 2019.
  - [22] 2020 digix advertisement ctr prediction. <https://www.kaggle.com/louischen7/2020-digix-advertisement-ctr-prediction>.
  - [23] S. Srivastava. Adversarial perturbation on learned index models, 2020. [https://github.com/srivastava-sar/LearnedIndex\\_AdversarialML](https://github.com/srivastava-sar/LearnedIndex_AdversarialML).
  - [24] Andrew Ng. Artificial intelligence is the new electricity, 2017. <https://www.youtube.com/watch?v=21EiKfQYZXc&feature=share>.