Discovery of Trend Dependencies Over Time-Series

by

Nicholas Bode

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Science (MSc) in Computer Science

Faculty of Science
University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

November 2023

THESIS EXAMINATION INFORMATION

Submitted by: **Nicholas Bode**

Master of Science in Computer Science

Thesis Title: Discovery of Trend Dependencies Over Time-Series

An oral defense of this thesis took place on November 24th, 2023 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Andrew Houge
Research Supervisor	Dr. Heidar Davoudi
Research Co-supervisor	Dr. Jarek Szlichta
Examining Committee Member	Dr. Pejman Mirza-Babaei
Thesis Examiner	Dr. Ying Zhu

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

We improve constraint-based data quality using trend dependency (TD) discovery, extending existing order dependencies (ODs) to allow variations and exceptions. Unlike ODs, TDs capture approximate functional mappings between attributes, addressing the limitations of monotonicity. Our approach involves automatic discovery over entire datasets and piecewise subsets. Optimizing across all possible mappings is impractical, but a single linear pass enables efficient pruning, making segmentation and trend discovery feasible with minimal accuracy loss. We conducted comprehensive experiments on real-world and synthetic data to evaluate our models.

Keywords: Data quality, Data cleaning, Quality constraints, Symbolic Regression, Segmented Representation

Author's Declaration

I hereby declare that this submission is entirely my own work, in my own words, and that all sources used in researching it are fully acknowledged. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

_____ Nicholas Bode

Statement of Contributions

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and inventive knowledge described in this thesis.

Acknowledgements

I would like to extend my heartfelt gratitude to my dedicated supervisors, Dr. Kourosh Davoudi and Dr. Jarek Szlichta, whose unwavering guidance and mentorship have been instrumental throughout my academic journey. Their expertise, patience, and commitment to my growth as a scholar have shaped not only my research but also my character.

To my beloved family, your continuous support and encouragement have provided me with the foundation and motivation to pursue excellence in my studies. Your sacrifices and belief in me have been my driving force.

I am deeply thankful to the exceptional faculty and my fellow students at Ontario Tech University's Computer Science Department. Your collective knowledge, insights, and camaraderie have enriched my learning experience, fostering an environment where intellectual growth thrives. I am proud to have been part of such a vibrant academic community.

This dedication is a testament to the collaborative efforts and the nurturing environment that have enabled me to reach this milestone in my academic career.

Table of Contents

List of Tables

List of Figures

List of Symbols

abcOD	Approximate Band Conditional Order Dependency
ATD, ψ_e^g	Approximate Trend Dependency
bandOD, $X \mapsto_\Delta Y$	Band Order Dependency
$c(\mathbf{R}_1)$	The cost associated with adding a removed tuple to $\mathbf{R}_1$
CFD	Conditional Functional Dependency
CTD, (ψ_e^g, X_s)	Conditional Trend Dependency
$e(\psi)$	The approximation ratio of a TD
f, F	A function representing a trend, the function search space
FD	Functional Dependency
$g(\mathbf{R}_0)$	The gain associated with removing a tuple from $\mathbf{R}_0$
kNN	k-Nearest Neighbours
OD, $X \mapsto Y$	Order Dependency
P, p	A partition scheme, a single breakpoint
$\mathbf{R}, \mathbf{r}$	A relation, a removal set on $\mathbf{R}$
s, t	A single tuple
SWAB	Sliding Window and Bottom-Up
TD, $\psi : X \mapsto_{f,\phi} Y$	Trend Dependency
x_t, x_s, y_t, y_s	An instance of an attribute within a given tuple
X, Y	An attribute of R
ϕ, Φ	An error function, the search space of error functions

Chapter 1

Introduction

1.1 Motivation

To ensure high-quality, data-driven analysis, you must work with clean data. Providing data riddled with errors to a model during training will result in a skewed representation of the underlying patterns. Data quality constraints give a straightforward method for assessing the cleanliness of a given dataset and offering clear directions on the data cleaning process. In this section, we will first explore two existing forms of data quality constraints: Functional Dependencies (FDs) and Order Dependencies (ODs), as well as a few of their extensions. Our aim is to fill the gaps left by them through Trend Dependencies (TDs).

Functional dependencies and their extensions have been the primary focus. FDs are a set of rules which specify a dependency relationship between attribute values. They are denoted as a logical implication for example the value of attribute *State* is dependent on attribute *ZIP* would be denoted $ZIP \rightarrow State$. A conditional functional dependency [4] (CFD) details the relationship for specific values within each attribute for example: *ZIP:* /060[0-9]{2} $\rightarrow$ *State:* Connecticut. Here the regular expression given corresponds to any 5-digit number beginning with "060". This prefix necessarily implies

that the corresponding address is located within the state of Connecticut meaning our CFD should always hold.

Another extension, order dependencies (ODs) [32], provides a means to model the semantics of monotonically related attributes. As an example, the OD $YoB \mapsto Age$ implies that the table sorted on YoB should also be sorted on Age This order often does not hold perfectly in real-world data and often includes slight variations. To address this, introducing band order dependencies (bandODs) [23] allowed for slight variations in the otherwise strictly monotonic data. Band ODs were then extended further into approximate band conditional order dependencies (abcODs) [24], which allow for a few extreme violations and allow the dependency to hold conditionally over different subsets of the data.

In the evolving data quality landscape, limitations in current approaches become apparent and require attention. Notably, no OD version provides a reasonable way to clean data when presented with deviations from linear monotonicity, a common scenario in real-world datasets. Furthermore, handling missing or erroneous values under these dependencies needs a more systematic and intuitive approach. While they do provide some information about the connection between two attributes, there is far more information that can be gleaned.

To remedy these issues and pave the way for more robust data-driven models, we introduce a novel class of data dependencies known as trend dependencies (TDs). These TDs offer all the advantages of ODs, bandODs, and abcODs while offering simple and practical rules for replacing missing or erroneous values. Under a TD, when inserting a value into a tuple, the only requirement is that it falls within a trend defined by the previous and subsequent values. This approach works well for linear, strictly monotonic data. It also gracefully handles more complex scenarios, as exemplified by the data in Table 1.1 sourced from the OWID COVID-19 dataset [25].

We can see that an OD holds over the data. If the value for March was missing,

Month	Cases
January	555
February	76206
March	341585
April	2553508
May	5110064
June	8805336
July	14713623

Table 1.1: Covid-19 Total Cases by Month (2020)

a simple linear interpolation between the two neighboring elements February and April gives us 1315857. A simple exponential curve fit predicts 819536. While neither is entirely accurate; the exponential fit is half the distance from the true value. We illustrate this in Fig 1.1. This problem becomes even more critical when the data does not adhere strictly to an OD.

1.2 Problem Definition

In this thesis, we consider applying more complex models to enhance the expressive power of constraint-based data quality. The goal is to find a simple-to-describe approximation of a trend to assess whether a given instance is genuinely a member and to fill in any gaps left by either erroneous or missing tuples. In essence, we search for a function that maps approximately between two attributes and use it to clean the given relation.

One challenge that quickly presents itself is the tendency for underlying trends to change due to factors not explicitly present in the data. To handle this, we expand the problem to finding segmented representations, separating the data into chunks that adhere to different functions as in Figure 1.2.

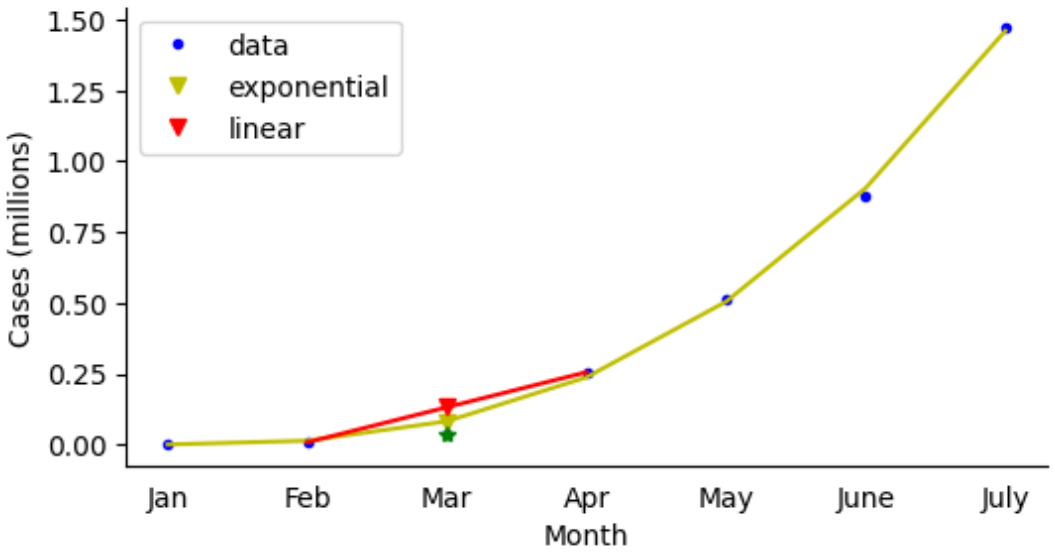

Figure 1.1: Covid-19 Total Cases by Month (2020)

This research compares several different methods for detecting this class of dependency. We compare their similarity to the underlying data and their capacity to detect outliers of varying significance. The naive solution to the problem of segmented symbolic representation is intractable for anything beyond trivially small datasets. We, therefore, design efficient sampling techniques to minimize the resource cost while maintaining a high standard for accuracy.

We perform assessments over real-world and synthetic datasets, encompassing datasets with errors and missing values and those without. For real datasets, we use the OWID COVID-19 dataset [25] and the *Huge Stock Market Dataset* from Kaggle, which together provide thousands of time series that adhere to a complex combination of trends. We divide the synthetic data into two subsets, referred to as "in-model" or "out-of-model," based on whether the models used for prediction include the functions used for generation. For testing the cleaning capabilities we replace artificially removed values and introduce statistically significant errors so we can accurately measure the model's capacity for detection. We also explore the model's capacity for detecting real-world errors,

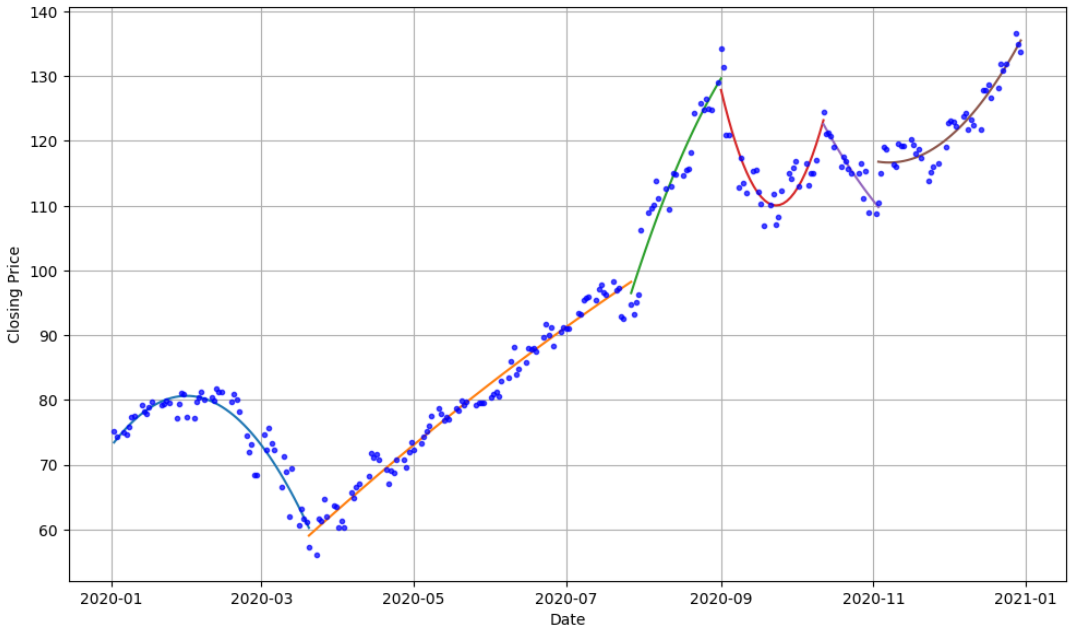

Figure 1.2: APPL Stock Segmented Trend

but the experiments are far less extensive due to the difficulty of finding data where errors are pre-labeled.

1.3 Contribution

We present a novel class of data dependencies focused on identifying and approximating underlying trends in the data. We formalize the definition of trend dependencies and provide concrete metrics for assessing their quality over noisy, error-ridden datasets. We break up the discovery of these into two major components: segmentation and regression. We design and compare a variety of methods for each.

For segmentation, we compare three approaches: standard bottom-up segmentation, sliding window, and bottom-up (SWAB) segmentation [17], and a deep learning, kNN-based binary classifier [6]. We use a simple convergence-based algorithm for selecting the number of desired change points for both classifiers. We also offer a few methods for quickly pruning the initial dataset to reduce the number of potential change points and

anomalies. Each method has strengths and weaknesses, and we provide guidelines for appropriate usage.

Regression presented a significant challenge as the space of possible functions needs to be explored efficiently. Recent work in symbolic regression has demonstrated an impressive capacity to generate functional approximations to data when that data is clean and low-noise [3, 14, 20]. Outside of that scope, things become less simple. Because of this, we cannot use existing models out of the box and instead must train our own which has been tuned for this specific problem, leveraging a combination of a set transformer [21] encoder and transformer [35] decoder. We compare this approach to a Savitsky-Golay curve smoothing filter, simple polynomial regression, and a mix of the two. Because of the nature of symbolic regression, the space of component function types is arbitrarily large. Without loss of generality, we focus solely on combinations of elementary functions. Including additional function types can be achieved by adding them to the model's token dictionary. Working with noisy data presented a significant challenge for both regression models. We developed a simple solution involving an additional noise term, which we learned alongside the base regression model. We compare this alongside the baseline regression model and provide avenues for potential future improvements.

As the number of tuples increases, the regression and segmentation problems rapidly become intractable. To combat this, we present a novel sampling scheme based on noise estimation and change in approximated slope based on a similar approach used in climate change forecasting [38]. This sampling scheme allows us to generate a set of candidate breakpoints (where the slope of a linear fit before the point is significantly different from the slope of a fit after), generate candidate errors (points with high deviation from the linear fits), and perform pre-regression sampling (using points with low deviation from the linear fits). These significantly reduce the time complexity of all aspects of this problem without significantly affecting performance.

The regression component is independently evaluated entirely on synthetic data. We

do all pre-training on synthetic data and describe the general methods for its creation. We test the segmentation on both real-world and synthetic datasets, and these tests rely on the regression algorithms we established earlier. We demonstrate significant improvement in accuracy and speed over standard methods for both and present optimizations that allow the regression step to provide shortcuts for the segmentation.

In summary, our primary contributions are as follows.

1. Trend Dependencies:
   (a) We introduce a novel class of data dependencies focused on identifying and approximating underlying trends in the data continuing work done on Order Depdencies [32] and their extensions [22–24]
   (b) We formalize the definition of trend dependencies and provide concrete metrics for assessing their quality over noisy, error-ridden datasets.
2. Segmentation Methods:
   (a) We compare three approaches for segmentation: standard bottom-up segmentation, sliding window, and bottom-up (SWAB) segmentation [17] and a binary classifier.
   (b) We introduce a deep learning, kNN-based binary classifier similar to [6] and provide an algorithm for selecting the number of desired change points for both classifiers.
   (c) We offer methods for quickly pruning the initial dataset for potential change points and provide usage guidelines.
3. Regression Techniques:
   (a) We use symbolic regression, leveraging a combination of a set transformer encoder [21] and transformer decoder [35], tuned for this specific problem.

(b) We compare this approach to curve smoothing filters, simple polynomial regression, and a mix of the two.

(c) We address handling noisy data by introducing an additional noise term learned alongside the base regression model.

4. Efficient Scaling:

(a) We propose a novel sampling scheme based on noise estimation and change in approximated slope.

(b) This sampling scheme allows us to reduce the time complexity of all aspects of this problem without significantly affecting performance.

5. Comprehensive Evaluation:

(a) We thoroughly evaluate our segmentation and regression methods on both real-world and synthetic datasets.

(b) We demonstrate substantial improvements in accuracy and speed over traditional methods and discuss optimizations.

1.4 Thesis Outline

This thesis is organized into six Chapters and structured as follows:

- Chapter 2 introduces a few different classes of data dependencies, which provide the framework around which we built trend dependencies. We also describe some segmentation and regression components in our final architecture.

- Chapter 3 takes a deeper look at the evolution of constraint-based data quality. It also showcases some previous work on segmented data representation alongside an exploration of each task independently.

- Chapter 4 first provides our explicit definition of trend dependencies. We then detail the different options for the regression step and follow the same process for segmentation. Next, we describe the optimization and sampling techniques we used to improve the practicality of these models. Finally, we describe the complete architecture for trend dependency discovery and the steps required for cleaning.

- Chapter 5 reports on the success of each of the different components individually before comparing the most successful variants against other data-quality constraints. The tests assess the model accuracy and rate of error discovery as the error significance decreases, as well as the time complexity for each piece with and without sampling.

- In Chapter 6, we highlight the key contributions and some of the limitations of the thesis research. The Chapter also presents some interesting future directions along which others can extend the research presented in this thesis.

Chapter 2

Background

2.1 Definitions and Notation

We use the following notational conventions:

- **Relations:** We denote a relation or table with $\mathbf{R}$. A relation is a set of tuples denoted t, s. Each tuple is comprised of attributes belonging to attribute sets X, Y with X generally being the independent variable and Y the dependent. An instance of a given attribute is denoted with its lowercase subscripted with the tuple for which it is a member (i.e. x_s, x_t, y_s, y_t). The list of all instances of a given attribute is given by its corresponding attribute letter in bold-face $\mathbf{X}, \mathbf{Y}$

- **Partitions:** A partition or segmentation scheme P is a set of breakpoints for a table $\mathbf{R}$ sorted on attribute $\mathbf{X}$. The values of P, denoted p refer to a single tuple in $\mathbf{R}$. A segment S corresponds to a range of indices $[i, j]$. A segment of a relation; the corresponding subset of tuples is denoted by the attribute on which it is sorted subscripted with the index range either $\mathbf{X}_S$ or $\mathbf{X}_{i,j}$.

- **Functions:** A function, when used to represent a trend, is denoted f. When used to represent the margin of error, we denote it ϕ. We currently limit the error

function to a constant leaving the extension to a general error function to future work. We denote the search space of functions with their corresponding capitals F and Φ

See 2.1 for a table of all notation conventions used in this document.

We then explicitly define a few terms used regularly throughout the paper.

Definition 2.1.1 (Segment) *A segment refers to a contiguous subset of the data when ordered on an attribute. Given a relation* $\mathbf{R}$ *ordered on* X *a segment is defined as a sub-sequence* $\mathbf{X}_S = [x_i, x_{i+1}, \ldots, x_j]$*, where* $1 \leq i \leq j \leq n$*.*

Definition 2.1.2 (Segment Purity) *Segment purity measures how well a segment captures a distinct pattern within the time series data. It quantifies the coherence and homogeneity of the observations within a segment.*

We can use several measures to assess segment purity, including:

- **Within-Segment Variance:** This is our primary measure of purity. Generally this corresponds the average deviation of each point from some central value. For a trend dependency this corresponds to the average distance from the estimated trend; in other words, the mean absolute error (MAE):

$$MAE = \frac{\Sigma_{i=0}^{n} |y_i - f(x_i)|}{n} \tag{2.1}$$

  where $(x_i, y_i) \in \mathbf{R}$ sorted on $\mathbf{X}$ and n is the number of tuples in $\mathbf{R}$

- **Correlation:** This measures the linear relationship between observations within a segment. Generally associated with the $R^2 = 1 - (\frac{RSS}{TSS})$ Where RSS is the sum of the squared residuals and TSS is the sum of squared values. We choose not to use this as, when dealing with non-linear trends, the R^2 value is no longer normalized. Instead we divide the MAE by our estimated noise value for a general metric.

- **Entropy:** Another useful metric which we won't use here but is important to consider. Entropy measures the amount of information required to convey the possible values of a given random variable. It was originally conceived for discrete variables but extensions to continuous probability distributions exist. All require an estimate of the underlying continuous distribution. Since we limit ourselves to constant error functions we can neglect the entropy of the noise but, when extending to arbitrary error functions the relative entropy will become an important metric to consider.

In our case, the variance and correlation are both used in different contexts. We largely ignore the entropy of the segment as we interpret any randomness as noise and account for it accordingly.

Moving on to our dependencies, we first define ODs and their extensions.

Definition 2.1.3 (Order Dependency) *Given attribute sets X, Y on a relation $\mathbf{R}$. An order dependency of the form $X \to Y$ states that if we sort $\mathbf{R}$ on X, it is also sorted on Y in ascending or descending order.*

Example 2.1.4 *Figure 1.1 has a clear example of a dataset over which the OD $Month \to Cases$ holds.*

We further extend these to allow for slight variations from monotonicity with bandODs, ignore a few significant exceptions, and hold conditionally over subsets of the data with abcODs.

Definition 2.1.5 (Band Order Dependency) *Given attribute sets X, Y on a relation $\mathbf{R}$. A band order dependency of the form $X \mapsto_\Delta Y$ holds if $x_s < x_t \implies y_s < y_t + \Delta$ or $y_s > y_t - \Delta \forall t, s \in \mathbf{R}$*

We will provide definitions for Trend Dependencies and their extensions in Chapter 4.

2.2 Summary

This chapter has established the fundamental concepts and definitions related to trend dependencies (TDs) within the broader context of data quality enhancement. The subsequent chapters build upon these foundations by delving into the automatic discovery of TDs, their applications, and their impact on data quality improvement. In later sections, we also discuss the evaluation of these dependencies using real-world and synthetic datasets in detail.

$\mathbf{R}$	A relation or table; a set of tuples with corresponding attributes
X, Y	An attribute of $\mathbf{R}$; X is independent Y is dependent
t, s	A single tuple
x_t, y_t, x_s, y_s	A single instance of an attribute within a given tuple
P	A partition scheme given as a set of breakpoints
p	A breakpoint in P; a single instance of the independent attribute X
f	A function used to represent the underlying trend
F	The function search space for trend discovery
ϕ	A function used to describe the allowable distance from the underlying trend
Φ	The search space of possible error functions
$\mathbf{X}_{i,j}\mathbf{X}_S$	A segment of a relation $\mathbf{R}$ sorted on the attribute X
$X \mapsto Y$	An order dependency holds between X and Y; X orders Y
$X \mapsto_\Delta Y$	A band order dependency holds between X and Y with band width Δ
$\psi : X \mapsto_{f,\phi} Y$	A trend dependency holds between attributes X and Y
$\mathbf{r}$	A removal set on $\mathbf{R}$
$e(\psi)$	The approximation ratio of a trend dependency
$g(\mathbf{R}_0)$	The maxinmal gain associated with removing a tuple from R_0
$c(\mathbf{R}_1)$	The cost associated with adding a removed tuple back to R_1
ψ_e^g	An approximate trend dependency with gain threshold g
$(\psi_e^g, \mathbf{X}_S)$	A conditional trend dependency over segment $\mathbf{X}_S$

Table 2.1: Notation Table

Chapter 3

Literature Review

Constraint-based data quality has a long history, of which we will provide a brief overview. Following that, we will look at the regression and segmentation steps individually alongside a few sources that combine the two. Finally, we will highlight the essential resources we drew on for the sampling techniques.

3.1 Discovering Data Dependencies

Constraint-based data quality assessment has a rich and storied history, representing a foundational aspect of data management. In this section, we embark on a journey to explore the core principles of data dependencies, which lie at the heart of data quality evaluation and database management. Much like the overarching chapter, we will briefly overview this historical context before delving into two pivotal types of data dependencies: Functional and Order. Our exploration will encompass their formal definitions, discovery methodologies, and intrinsic importance within the landscape of data-driven analysis. This section sets the stage for a comprehensive understanding of data dependencies, an essential foundation for our subsequent discussions on regression, segmentation, and synthesizing these concepts.

3.1.1 Functional Dependencies

Functional dependencies (FDs) are fundamental to data management, providing a formal framework for defining how the values of specific attributes uniquely determine others within a dataset [13]. FDs are pivotal in database design, aiding schema normalization and query optimization [8]. Their discovery is crucial for data quality assessment, with various algorithms and techniques developed to automate this process, facilitating data profiling and cleansing [27].

3.1.2 Conditional Functional Dependencies

Conditional Functional Dependencies (CFDs) represent an extension of traditional Functional Dependencies (FDs) that consider dependencies between attributes within specific conditions or subsets of data [4,7,8]. Unlike standard FDs, which hold universally across the entire dataset, CFDs capture context-specific dependencies. These dependencies have applications in various domains, mainly where data exhibit conditional relationships.

CFDs have gained significant attention in data quality assessment, enabling the discovery of nuanced dependencies that may be crucial for understanding and improving data quality within specific contexts [12]. They provide a means to identify constraints that hold for particular subsets of data while allowing exceptions in others, making them a valuable tool for data profiling and cleansing. The discovery and utilization of CFDs contribute to more accurate and context-aware data quality improvements.

3.1.3 Order Dependencies

Order Dependencies (ODs) are a cornerstone in data dependencies, providing a formal means to model the semantics of attributes that exhibit a monotonically related ordering within a dataset [15,16,31–33]. They define constraints that specify the order of attribute values, ensuring that when we sort a dataset on one attribute, we know we have sorted

it on another, either in ascending or descending order. ODs are essential in database schema design and optimization, guiding data organization for efficient querying and analysis.

3.1.4 Band Order Dependencies

Band Order Dependencies (bandODs) [23] were introduced to accommodate the often-realistic scenario of slight variations from strict monotonicity in real-world data. BandODs allow minor deviations in the otherwise monotonically ordered data, providing a more flexible approach to modeling data semantics. These dependencies are instrumental in scenarios where data exhibits inherent variability, enabling a more robust representation of the underlying relationships.

3.1.5 Approximate Band Conditional Order Dependencies

The evolution of ODs led to the development of Approximate Band Conditional Order Dependencies (abcODs) [22,24]. abcODs extend the concept further, allowing dependencies to hold conditionally over subsets of the data and permitting a few extreme violations. These dependencies capture nuanced relationships within datasets, making them particularly valuable for modeling complex data semantics and allowing for exceptions that may be critical in real-world applications.

3.2 Symbolic Regression

3.2.1 Genetic Programming

Genetic Programming (GP) is a widespread technique in symbolic regression. GP applies the principles of evolutionary algorithms to evolve mathematical expressions that best fit the given dataset [1]. It explores a population of candidate solutions represented as

tree structures, iteratively applying mutation, crossover, and selection operators to evolve more accurate expressions over generations. GP has been employed in various fields to discover symbolic formulas that describe complex relationships within data.

3.2.2 Neural Networks

The use of Neural Networks (NNs) in symbolic regression has gained attention in recent years, leading to two notable subdomains: Neural Symbolic Regression and End-to-End Symbolic Regression.

Neural Symbolic Regression

Neural-symbolic regression (NSR) [3, 18] methods aim to extract interpretable symbolic formulas from neural network architectures, allowing for a more transparent understanding of the underlying relationships within data. This fusion of neural networks and symbolic reasoning offers the potential to uncover precise mathematical expressions that humans can readily interpret, bridging the gap between data-driven modeling and traditional symbolic regression.

End-to-End Symbolic Regression

End-to-End Symbolic Regression (E2E-SR) [14] represents a recent improvement on NSR. E2E-SR leverages the capabilities of Transformers to directly learn symbolic representations from data without the need for intermediate stages. We train these models to predict symbolic expressions from input data, providing a seamless, end-to-end approach to symbolic regression. E2E-SR promises to simplify the modeling process, making it more accessible and efficient for various applications.

3.2.3 Deep Symbolic Regression

Deep Symbolic Regression (DSR) [29] presents a slightly different approach to symbolic regression. DSR leverages deep learning techniques to approximate complex functions and extract interpretable symbolic expressions from the learned models. Deep Symbolic Regression offers the advantage of automatic feature extraction and the ability to effectively handle high-dimensional and noisy data.

Deep Symbolic Regression is continually evolving, with ongoing research exploring novel architectures and training strategies to improve the accuracy and interpretability of symbolic expressions extracted from deep neural networks.

3.2.4 Benchmarking

Benchmarking symbolic regression algorithms is essential to assess their performance and identify their strengths and weaknesses. Several studies have focused on benchmarking state-of-the-art symbolic regression algorithms [34, 36] and many popular machine learning datasets include symbolic regression problems [28]. These evaluations help researchers and practitioners select appropriate techniques for specific tasks and datasets, ensuring that the chosen methods align with the desired accuracy and efficiency criteria. Benchmarking also aids in advancing the field by providing insights into areas that require further improvement and innovation.

3.3 Segmentation

Segmentation is fundamental in discovering TDs and patterns within time series data. Two prominent approaches to segmentation are Sliding Window and Bottom-up (SWAB) Segmentation and ClaSP.

3.3.1 Sliding Window and Bottom-up (SWAB)

SWAB is an algorithm designed for segmenting time series data efficiently [17]. SWAB employs a sliding window technique to identify segments within a time series. SWAB performs bottom-up segmentation, a slow but relatively accurate DP-based segmentation algorithm, over a sliding window. SWAB reduces time complexity significantly as the sliding window execution increases linearly while the segmentation decreases quadratically with decreasing window size.

3.3.2 ClaSP

ClaSP is a parameter-free time series segmentation method that offers efficient and effective data segmentation [6]. Unlike approaches that require manual parameter tuning, ClaSP automatically determines the optimal segmentation without user-defined parameters. This lack of hyperparameters makes ClaSP a robust choice for various time series analysis tasks, providing reliable segmentation results without the burden of parameter selection.

SWAB and ClaSP are solutions to the segmentation phase of TD discovery, facilitating the identification of distinct patterns within time series data for further analysis.

3.4 Segmented Regression

In discovering data dependencies, segmented regression plays a pivotal role in modeling relationships between variables when distinct patterns emerge within the data. Research in this area has predominantly focused on two categories: linear segmented regression and non-linear segmented regression. In this section, we delve into both these domains, highlighting essential methods and approaches that enable us to uncover meaningful structural changes within data.

3.4.1 Linear Segmented Regression

There has been extensive research on linear segmented regression models to capture piecewise linear relationships between variables. These models are helpful when data exhibits different linear behaviors in various regions. Here, we review prominent methods and techniques related to linear segmented regression.

Fast Grid Search Algorithms for Multi-phase Regression Models

Qianqian Chen's work on fast grid search algorithms for multi-phase regression models [5] provides insights into efficient techniques for estimating breakpoints in segmented regression. The research focuses on developing algorithms that can identify structural changes within data, enabling precise modeling of the linear relationships in distinct segments.

Interval Estimation for the Breakpoint

Vito MR Muggeo's study on interval estimation for the breakpoint in segmented regression [26] presents a smoothed score-based approach for accurately estimating breakpoints in linear segmented regression models. This methodology enhances the reliability of identifying structural shifts in data and contributes to robust modeling.

Computation and Analysis of Multiple Structural Change Models

Jushan Bai and Pierre Perron's work on computation and analysis of multiple structural change models [2] offers valuable insights into statistical methods for detecting structural changes in linear segmented regression. Their research contributes to developing algorithms that can effectively identify multiple breakpoints, facilitating modeling of complex linear relationships in data.

The exploration of linear segmented regression methods provides a foundation for understanding how we can capture structural changes in data through piecewise linear

modeling. These techniques enable researchers to unveil hidden dependencies and patterns within the data, making them a valuable tool in data dependency discovery.

3.4.2 Non-Linear Segmented Regression

While linear segmented regression models see wide usage, there are many scenarios where data relationships exhibit non-linear patterns with structural changes. Non-linear segmented regression approaches offer a solution to model such complexities. In this section, we explore methods and studies related to non-linear segmented regression, including fitting segmented polynomial regression models [9, 10] and fast grid search and bootstrap-based inference for continuous two-phase polynomial regression models [30].

3.5 Sampling

The Running Slope Difference (RSD) t-test, developed initially as a statistical method for detecting trend turning points, has found versatile applications beyond climate science. This innovative approach, as demonstrated by [38] in their work titled "Assessment of the Running Slope Difference (RSD) t-Test," has proven valuable in various fields, including data analysis and anomaly detection. By evaluating the differences in slopes between consecutive segments of time series data, the RSD t-test provides a robust framework for identifying breakpoints and anomalies. In data sampling, this method allows us to pinpoint significant changes or irregularities within datasets, facilitating more informed decision-making and data-driven insights across diverse domains.

Chapter 4

Methodology

In this section, we provide concrete definitions for the various classes of trend dependencies described in earlier sections. We then introduce the various models utilized for trend discovery, segmentation, noise estimation, smoothing, and sampling, along with the architecture that combines them.

4.1 Defining Trend Dependencies

We consider three classes of trend dependencies: standard, approximate, and conditional. Approximate TDs (ATDs) hold over the data for all but a few exceptions. Conditional TDs (CTDs) hold over only a subset of the data. CTDs may either be approximate or standard.

Definition 4.1.1 (Trend Dependency) *Given a function f, an error function ϕ, attribute sets X, Y on a relation $\mathbf{R}$ over which a bandOD holds, a Trend Dependency of the form $\psi : X \rightharpoonup_{f,\phi} Y$ holds perfectly over a set of tuples $\mathbf{R}$ if*

$$\forall x, y \in \mathbf{R} : |f(x) - y| < \phi(x) \tag{4.1}$$

Example 4.1.2 *In Figure 4.1 the TD Month* $\rightarrow_{f,\phi}$ *Cases holds without exception, where* $f = 121942x^{2.67}$ *and* $\phi = 2.55e5$

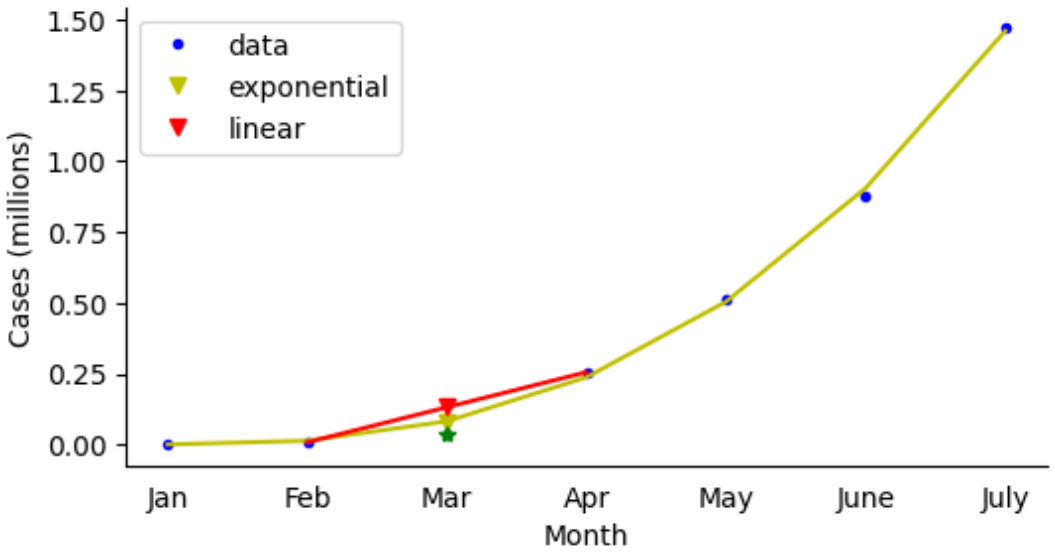

Figure 4.1: Covid-19 Total Cases by Month (2020)

Moving on to ATDs, we first define a few critical metrics for discovering these dependencies.

Definition 4.1.3 (Removal Set) *Given a relation* $\mathbf{R}$*, we define a* removal set $\mathbf{r}$ *on* $\mathbf{R}$ *as the set of tuples that we must remove for a TD* ψ *to hold perfectly. The ratio between the cardinalities of* $\mathbf{R}$ *and* $\mathbf{r}$*,* $e(\psi)$ *is called the approximation ratio of* ψ*.*

Example 4.1.4 *Looking at Figure 4.2 for the TD* ψ : $Years \rightarrow_{f,\phi} Salary$ *with* $f = 9449x + 25792$ *and* $\phi = 10000$ *to hold we must remove the removal set* $\mathbf{r} = \{19, 23\}$ *meaning it requires an approximation ratio* $e(\psi) = 2/30$ *on* $\mathbf{R}$

During the discovery process, we need some way of quantifying the value of increasing the size of the removal set. Initially, we used the difference in MAE between the function before and after removal. The change in MAE worked well for smaller datasets but

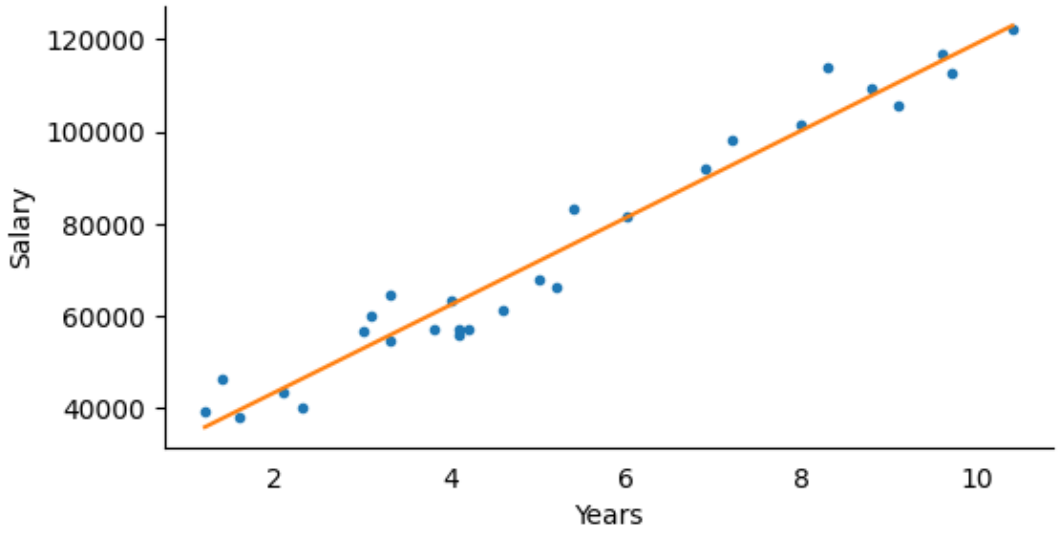

Figure 4.2: Salary vs. Years Experience

struggled with larger ones as the effect of removing an error is less substantial. Instead, we use the difference in the maximum error, in other words, the change in the value of ϕ.

Intuitively we can think of this as the percent reduction of the degree of the largest deviation from the predicted underlying function. A good heuristic for this is simply the difference between the first a second largest deviations. This gain value is a good indicator of the likelihood of a given tuple being an error.

Definition 4.1.5 (Gain) *Given a removal set* $\mathbf{r}_0$ *and a relation* $\mathbf{R}$*, we define the gain function* $g(\mathbf{R}_0)$ *of increasing the size of* $\mathbf{r}$ *by one as the most significant possible decrease in the maximum error.*

Where $\mathbf{R}_0 = \mathbf{R} \setminus \mathbf{r}_0$

$\mathbf{r}_1$ *is a possible new removal set value after adding a new tuple from* $\mathbf{R}_0$*, i.e.:*

$$\mathbf{r}_1 = \mathbf{r}_0 \cup \{s\} : \forall s \in \mathbf{R}_0 \tag{4.2}$$

and $\mathbf{R}_1 = \mathbf{R} \setminus \mathbf{r}_1$ *is the resulting relation.*

$$g(\mathbf{R}_0) = \frac{M_0 - m_1}{M_0} \tag{4.3}$$

where

$$M_0 = \max_{t \in \mathbf{R}_0}(|y_t - f_0(x_t)|) \tag{4.4}$$

represents maximum absolute error before removal,

f_0 *represents the function mapping* $X \rightharpoonup Y$ *found over* $\mathbf{R}_0$ *and,*

$$m_1 = \min_{s \in \mathbf{R}_0}(\max_{t \in \mathbf{R}_1}(|y_t - f_1(x_t)|)) \tag{4.5}$$

represents the minimum, max absolute error after removal,

Definition 4.1.6 (Cost) *With a similar intuition as the gain, given that* $\mathbf{r}_0$ *is a possible new removal set value after selecting a tuple from* $\mathbf{r}_1$ *to reinsert, i.e.,*

$$\mathbf{r}_0 = \mathbf{r}_1 \setminus \{s\} : \forall s \in \mathbf{r}_1 \tag{4.6}$$

and $\mathbf{R}_0 = \mathbf{R} \setminus \mathbf{r}_0$ *is the resulting relation.*

We define the minimum cost of decreasing the approximation ratio $c(\mathbf{R}_1)$ *as*

$$c(\mathbf{R}_1) = \frac{m_0 - M_1}{M_0} \tag{4.7}$$

where

$$m_0 = \min_{s \in \mathbf{r}_1}(\max_{t \in \mathbf{R}_0}(|y_t - f_0(x_t)|)) \tag{4.8}$$

represents the minimum max error across all possible values of $\mathbf{R}_0$*,*

f_1 *represents the function mapping* $X \rightharpoonup Y$ *found over* $\mathbf{R}_1$

$$M_1 = \max_{t \in \mathbf{R}_1}(|y_t - f_1(x_t)|) \tag{4.9}$$

represents the maximum error across all values in $\mathbf{R}_1$*,*

Example 4.1.7 *Take the simple example* $\mathbf{R}_0 = \{(1,6),(2,5),(3,8),(4,3),(5,2)\}$ *where* $\mathbf{R} = \mathbf{R}_0$ *and* $\mathbf{r}_0 = \varnothing$. *Assuming we use a new linear fit for each iteration, the gain* $g(\mathbf{R}_0) = 3.2$ *with new removal set* $\mathbf{r}_1 = (3,8)$. *The cost* $c(\mathbf{R}_1)$ *is identical. If we instead only calculate the fit for the initial iteration, which is far less computationally expensive,* $g = 2.4$ *and* $c = 4.0$

Definition 4.1.8 (Approximate Trend Dependency) *We say an ATD of the form* $\psi_e^g = X \rightarrow_{f,\phi} Y$ *holds on a relation* $\mathbf{R}$ *if the TD* ψ *holds with approximation ratio* $e(\psi) = e$ *to a threshold* g *meaning no potential addition to the removal set provides* $g(\mathbf{R}, \mathbf{r}_1) > g$. *We can omit the sub or superscript as each is implied by the other, but at least one is required.*

Example 4.1.9 *The ATD represented in Figure 4.2 is denoted* $\psi_{2/30}^{0.1} = Years \rightarrow_{f,\phi} Salary$ *with the same* f *and* ϕ.

Lastly, we define conditional trend dependencies (CTDs).

Conditional Trend Dependency (CTD)

Generally, we denote a CTD with an ATD and a range of indices $(\psi_e^g, \mathbf{X}_s)$. It holds over all the same conditions of an ATD over said index range.

Function	MAE
Linear	1.71e6
Quadratic	2.49e5
Cubic	1.64e5
Logarithmic	∞
Exponential	1.63e5

Table 4.1: Comparison of Fit Errors on COVID-19 Data

4.2 Trend Discovery

This section explores methods for fitting a trend to a given data set. In Sec. 3.1, we focus on clean data that adheres perfectly to an OD. Sec 3.2, we include deviations from strict order incorporating noise. Finally, in Sec 3.2, we explore the detection of ATDs on data that have significant anomalies.

4.2.1 Discovery on Strictly Ordered Data

To analyze our mini COVID-19 dataset (as shown in Figure 1.1), we immediately turn our attention to the fitting of mathematical functions. An exponential function provides a reasonably good fit for this data. In contrast, achieving a comparable level of accuracy with a polynomial necessitates an increase in degree beyond three. It is rare to see such high polynomial degrees outside of fourier transforms and they are not recommended for use in regression discontinuity systems [11]. Consequently, we confine our analysis to three categories of functions: polynomials with degrees less than or equal to three, exponentials of the form $a(x+b)^e+c$, and logarithmic functions of the form $a\log_b(x+c)+d$. Through a systematic evaluation of these three function types, as presented in Table 4.1, it becomes evident that the exponential function exhibits the most negligible absolute error (MAE), thereby warranting its selection as the most suitable trend function.

Furthermore, we explore the potential of symbolic regression for function detection. Notably, when data strictly adheres to an order dependency, the improvement brought about by symbolic regression is virtually negligible. However, there are promising indications of its effectiveness, particularly in scenarios involving Approximate Trend Dependencies (ATDs).

4.2.2 Noisy Data with No Errors

Computational efficiency becomes a primary concern when dealing with noisy datasets that do not inherently contain errors. Due to its lower computational demands, we adopt a strategy that prioritizes polynomial fitting over exponential and logarithmic functions.

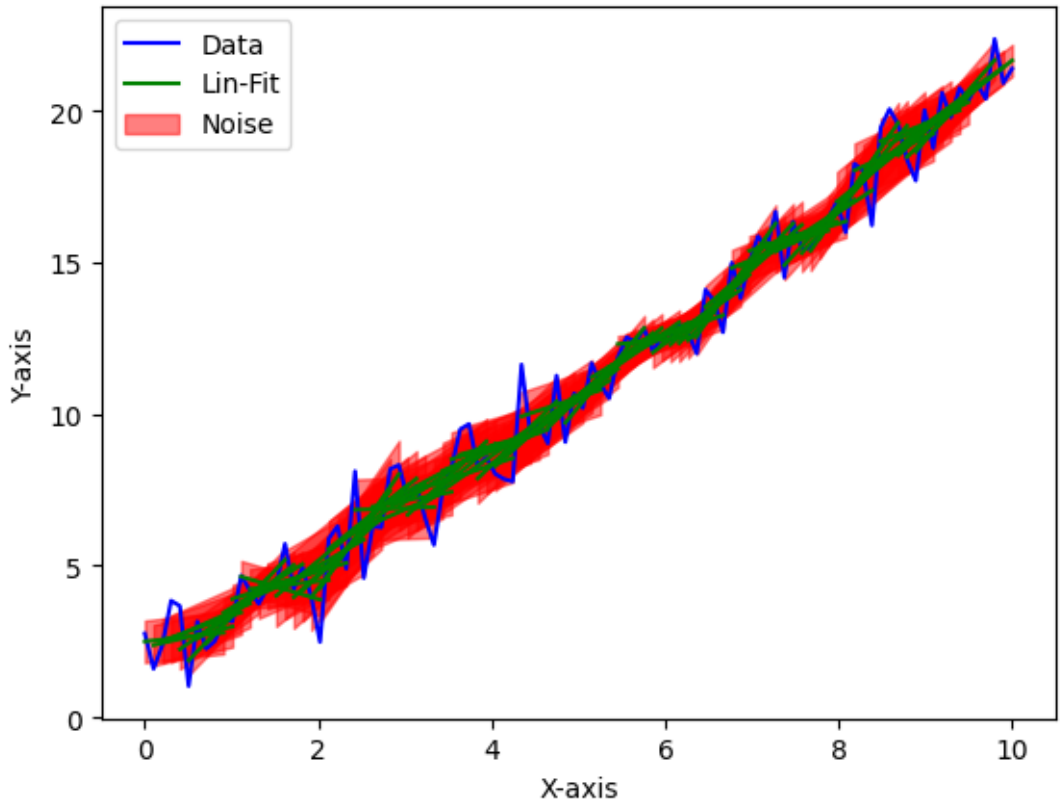

Figure 4.3: Noise Estimate: $w = 10$, $\eta_{est} = 0.69$, $\eta_{true} = 0.8$

Our selection of the appropriate function model hinges on the Mean Absolute Error (MAE) relative to our estimated noise level. If the MAE significantly surpasses our noise

estimate, we consider exploring logarithmic and exponential functions. This approach optimizes computational resources, balancing model complexity and data accuracy.

To illustrate our noise estimation method, we use the synthetic dataset in Figure 4.3. This dataset is generated using the function $y = 2x + \mathcal{N}(\mu, \sigma)$ with $\mu = 0$ and $\sigma = 0.8$ As can be seen, the noise estimate is slightly lower than the true generating noise. This is to be expected a smaller number of sample points allows large deviations to skew the fit resulting in them having less impact on the mean error. We can somewhat remedy this by increasing the window size at the risk of overweighting any changes in the underlying trend. Since the trend in Figure 4.3 is completely linear any increase in window size would result in the noise estimate asymptotically approaching the true value. However, this solution is not general and any dataset with a non-linear underlying trend would suffer greatly after a certain threshold.

4.2.3 Approximate Trend Discovery

In data analysis, outliers can manifest in various forms, including erroneous values, missing data, or points that deviate significantly from the norm. Regardless of their nature, outliers pose a common challenge by exerting a disruptive influence on regression models. For instance, replacing a single salary data point with an erroneous zero can more than double the Mean Absolute Error (MAE) of a regression model. Importantly, this impact on error metrics may not reflect a corresponding effect on noise estimation. Consequently, without proper mitigation, outliers can lead to the erroneous assignment of complex trends to otherwise straightforward datasets.

To address this issue, we introduce Approximate Trend Dependencies (ATDs). Since we often need more prior knowledge regarding the quantity and locations of anomalies within a dataset, detecting these anomalies becomes a critical step to exclude them from the trend-fitting process.

Optimal Point Removal Strategy and Gain Calculation

In practice, the optimal point removal strategy is not merely about minimizing errors but also about understanding when a removal set is sufficiently large or when further removals are necessary. A crucial factor to consider is the gain achieved by adding or removing a data point from the removal set. The high value suggests we should remove the data point, while the low gain is inconclusive, especially when multiple anomalies of similar magnitude are present.

Estimation of Noise Distribution and Computational Efficiency

Our solution to this challenge involves estimating a distribution for the noise term around a predicted function, excluding the potential outlier. We identify outliers as data points with a distance greater than three standard deviations from the mean. While this will result in some false positives, the impact of false negatives is generally far greater and the associated cost is worth the improved detection rate.

Efficient Outlier Detection

Exhaustively calculating this for every data point is computationally infeasible due to the exponential growth in potential removal sets, resulting in a $O(2^N)$ complexity. To mitigate this, we employ a more efficient approach with a $O(N^3)$ complexity, utilizing an $O(N^2)$ regression function. We sequentially consider individual outliers for function prediction, resulting in $O(AN^2)$ complexity, where A represents the number of detected outliers.

Pruning Techniques for Outlier Removal

We optimize further by pruning the initial dataset, providing insights into conditional Trend Dependency (TD) discovery. Our pruning techniques rely on the same filter used for noise estimation. We explore three distinct methods:

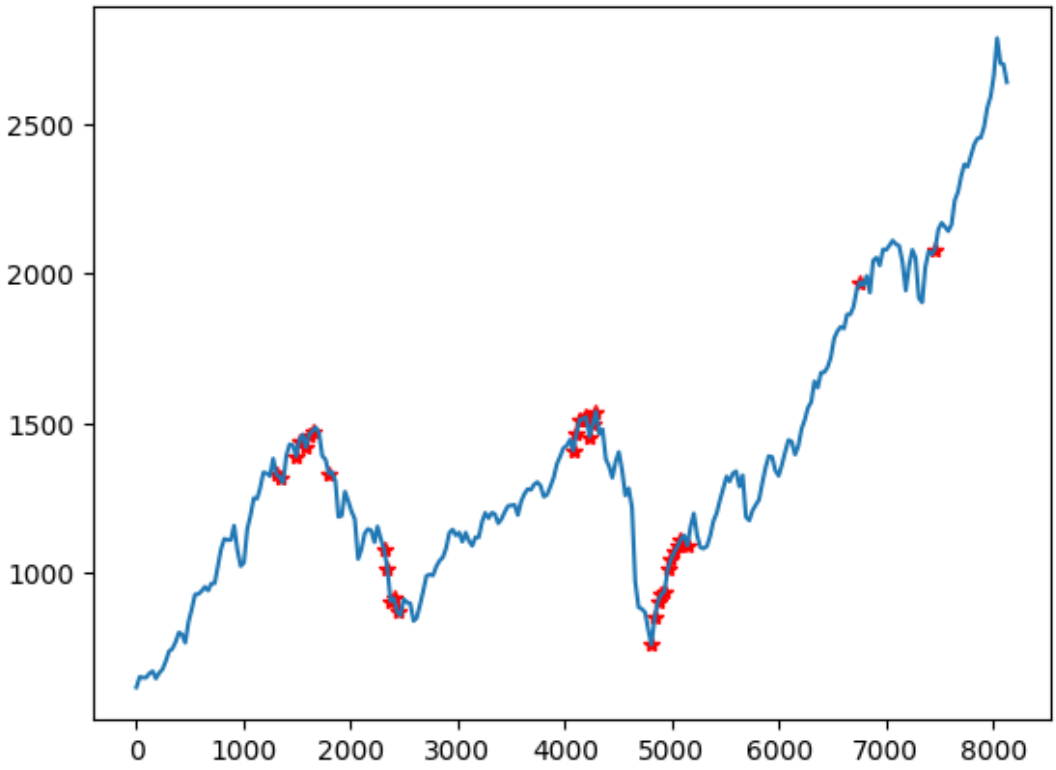

Figure 4.4: Candidate breakpoints Slope Difference

1. **Distance-Based Pruning:** Initially, we measure the distance of data points from the predicted line within a window, typically using three times the standard deviation as a baseline. This method has limitations, particularly in the case of skewed windows and conflicting classifications due to data points appearing in multiple windows.

2. **Count-Based Pruning:** Two additional techniques are introduced to address these limitations. The first method counts the number of windows in which a data point is the farthest from the predicted line. We consider it a candidate outlier if it consistently holds this distinction in most windows. Removing points with high counts below the majority threshold improves the trend prediction.

3. **Slope Difference Method:** The final and preferred method relies on analyzing the difference in slopes when a data point is at the edges of a sliding window.

As illustrated in Figures 4.5 and 4.6. Pseudocode for this method is provided for reference in Algorithm 1, as it is employed regularly. This technique detects abrupt changes in the underlying trend and individual data points using the absolute angular distance between the two (Line 17 in Algorithm 1). Although it requires a second pass over the data, its efficiency remains favorable, given its potential to reduce regression steps and enhance the pruning process. As illustrated in Figures 4.5 and 4.6. Peak detection is the most effective way to find candidates but is more expensive than simply selecting the top-k values. It is important to note that for points close to the edge (Line 4 in Algorithm 1) we treat the slope difference as zero as there isn't enough data to make a reasonable slope estimate.

Algorithm 1 Calculating Slope Differences

Takes a sliding window over the data set and approximates the slope at each. Then calculates the difference and slope at each point using the window before and after it.

Input: D - data, w - window size

```
1: procedure CalculateSlopeDifferences(D, w)
2:     S ← []                                   ▷ List to store slope differences
3:     for i ← 0 to len(D) − 1 do
4:         if i < 3 or len(D) − i < 3 then
5:             append 0 to S                    ▷ No significant difference
6:         else
7:             if i < w then
8:                 L ← slope(D[: i])            ▷ Left slope
9:             else
10:                L ← slope(D[i − w : i])
11:            end if
12:            if len(D) − i < w then
13:                R ← slope(D[i :])            ▷ Right slope
14:            else
15:                R ← slope(D[i : i + w])
16:            end if
17:            append |(arctan(L)) − (arctan(R))| to S   ▷ Slope difference
18:        end if
19:    end for
20:    return S
21: end procedure
```

Final Prediction and Complexity Analysis

Once we identify candidate errors, we employ the remaining data points to generate our segmented representation. Subsequently, we iteratively apply our gain or cost metric to select the ideal removal set based on some threshold as in Algorithm 2. This approach maintains a complexity of approximately $O(N^2)$, akin to the iterative point removal process.

Algorithm 2 Generate Removal Set

Uses the gain metric to continuously expand the removal set until a threshold is reached

Input: D - data, T - threshold

1: **procedure** GenerateRemovalSet(D, T)
2: $C \leftarrow$ GenerateCandidates(D) ▷ C_x - X values, C_y - Y values
3: $g_0 \leftarrow \infty$ ▷ Set initial max gain to max value
4: $f \leftarrow$ TrendDependency(D)
5: $\mathbf{r} \leftarrow \varnothing$
6: **while** $g > T$ **do**
7: $\mathbf{r} \leftarrow \max(|f(C_x) - C_y|)$
8: $R = D - \mathbf{r}$
9: $M_0 \leftarrow \max(|f(D_x) - D_y)|)$
10: $m_1 \leftarrow \max(|f(R_x) - R_y)|)$
11: $g = \frac{M_0 - m_1}{M - 0}$
12: **end while**
13: **end procedure**

The same logic can be applied in reverse using the cost in Equation 4.7. Instead starting from $\mathbf{r} = C$ and removing until the cost is less than some threshold T.

Advantages and Limitations

Importantly, this method offers notable advantages, including a reduced risk of error-induced prediction skew and identifying optimal points for exclusion, enabling sampling of smaller data subsets and reducing complexity. However, it is essential to acknowledge that this may lead to less precise function predictions in specific scenarios. We empirically test various sampling schemes over real-world data to further investigate these outcomes in Section 5.

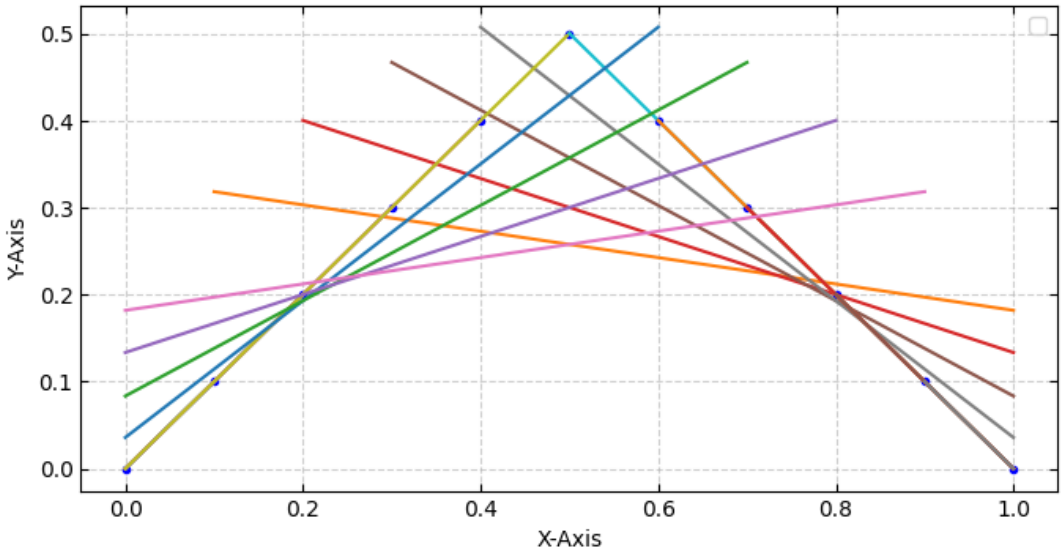

Figure 4.5: Slope Difference Calculation

4.2.4 Deep Learning for Symbolic Regression

Symbolic regression has seen a resurgence with the recent successes of neural network-based approaches, marking a shift from traditional genetic programming methods [1,37]. This section explores the application of deep learning techniques for symbolic regression, mainly focusing on their adaptability to noisy data and their recent outperformance of genetic programming-based solutions, in general, [36].

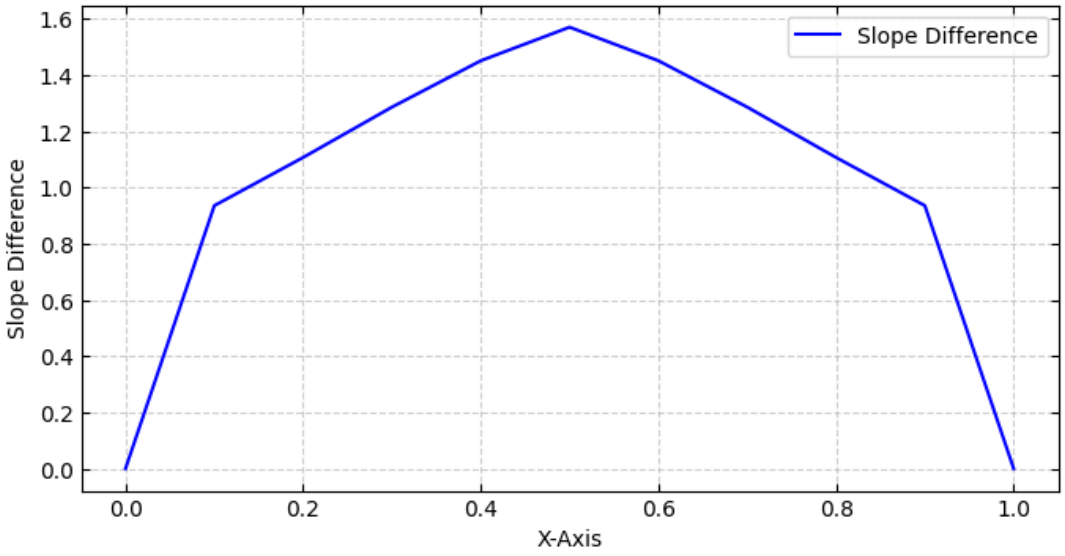

Figure 4.6: Resulting Slope Difference

Neural Symbolic Regression

Our initial approach drew inspiration from Neural-Symbolic Regression (NSR) proposed by Kamienny et al. in their work [14]. The model architecture consists of a set transformer for encoding low-dimensional data representations and a transformer decoder for generating symbolic expressions in prefix notation as illustrated in Figure 4.7

While NSR was designed initially for error-free data, we assessed its performance on datasets containing noise and errors. We carefully created our synthetic training dataset to encompass various function types. To balance computational efficiency and model expressiveness, we constrained the skeleton size to 8 tokens, including coefficients, to limit the function search space.

When applied to real-world data, mainly focusing on datasets with approximate monotonicity, the NSR model exhibited modest improvements in accuracy. However, it incurred significantly slower inference times due to increased model complexity, which may only sometimes align with practical objectives. To address this slowdown, we explored aggressive sampling techniques, which, though effective in mitigating inference

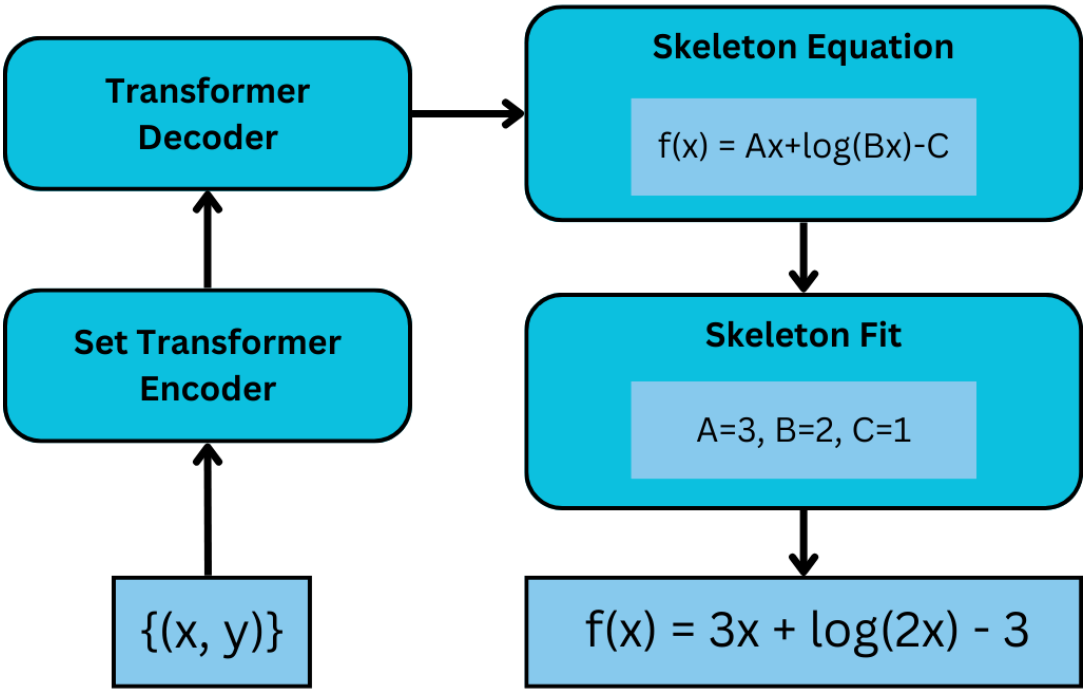

Figure 4.7: Neural Symbolic Regression

time, introduced a noticeable drop in accuracy.

Our subsequent analysis, detailed in later sections, includes a comprehensive evaluation of the NSR-based approach and other methodologies, such as our baseline least squares regression. This evaluation adheres to established benchmarking guidelines for symbolic regression, emphasizing synthetic data geared toward trend discovery.

Direct Mapping

Our second approach involved training a traditional multi-layer perceptron (MLP) to map input-output pairs using a custom policy gradient. This method excelled in handling clean, noisy data but struggled with more significant errors. It outperformed even our primary regression method in speed, primarily because it required fewer data points for accuracy.

However, the initial pruning step became the most time-consuming part of the process. With it, the method's susceptibility to outliers rendered it more practical, ultimately reducing its time-related advantages. Although it achieved superior accuracy compared to neural symbolic regression, it sacrificed interpretability and necessitated larger storage space to maintain the entire model state for each predicted function.

We used a reinforcement learning-based approach to allow for a variable reward function based on the expected gain. This design also allows us to mute the effects of each new relation by reducing any changes to the policy network.

Our approach involved pretraining on a general task and fine-tuning for task-specific execution.

Pretraining on Synthetic In-Model Data (*syn-im*):

We initiated our work by pretraining the model on *syn-im*, a synthetic dataset carefully constructed to represent a broad spectrum of distinct functions. We optimized the MLP for handling continuous input and output values. We randomly initialized the policy network's weights with the most successful iterations afterward to expedite the learning process and ensure adaptability to various relations.

A critical aspect of our approach was formulating a policy gradient objective function. This objective aimed to maximize the expected return or reward for a continuous action space, a fundamental requirement for effective regression. The policy gradient loss took the form: z

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log(\pi_\theta(a|s) Q^\pi(s,a))] \tag{4.10}$$

Here, θ denoted the policy network's parameters, $\pi_\theta(a|s)$ represented the policy network's output as a probability distribution over actions, and $Q^\pi(s,a)$ indicated the expected gain associated with taking action a in state s which is the anticipated reduction in MAE from the current model evaluated over a sample.

We generated synthetic data representing various relations within *syn-im* to ensure robustness and adaptability. These relations encompassed various functional forms and

behaviors, challenging the model to generalize effectively across them. We trained the policy network using REINFORCE with a drop-off after every new relation.

Evaluation on Synthetic Out-of-Model Data (*syn-oom*):

After successfully pretraining the model on *syn-im*, we evaluated its performance on the synthetic out-of-model data, *syn-oom*. This dataset consisted of relations distinct from those encountered during pretraining, putting the model's adaptability and generalization capabilities to the test.

Our fine-tuned policy network, enriched with the knowledge acquired during pretraining, was applied to *syn-oom* data. By doing so, we assessed its ability to perform regression on relations it had not explicitly encountered during training. This evaluation allowed us to gauge the model's robustness and capacity to effectively handle diverse, out-of-sample relations.

Comparison and Evaluation

Our experiments involved comparing these models against our baseline least squares regression to assess their strengths and weaknesses. We followed established benchmarking guidelines for symbolic regression [36], explicitly focusing on synthetic data tailored to our problem.

4.3 Discovering Conditional Trend Dependencies

With our various methods for detecting ATDs over an entire dataset, we now move to the problem of detecting multiple dependencies that individually only hold over a subset of the data. To do this, we propose a few methods for segmenting the data using our trend discovery techniques as a metric for accuracy. The naive solution is completely intractable with exponential complexity for just exploring the search space independently. The pruning technique using the slope difference test described above

dramatically reduces this effect, but it still leaves significant room for improvement.

We give an example of such a candidate error/breakpoint set on a graph of stock data in Figure 4.4

4.3.1 Standard Methods

We will look at a few standard segmentation methods first before getting into more advanced techniques.

Sliding Window

The first is a sliding window-based approach, which expands a segment (Line 4 in Algorithm 3) until it causes a significant decrease in segment purity (Line 5 in Algorithm 3). Once the dip happens, we cut the segment, and a new one begins (Line 6-7 in Algorithm 3). This process works well in principle because it avoids the requirement for setting a specific number of segments. It also results in a significant reduction in time complexity, exploring only N possible segments. Its shortcomings, however, are far more critical. The segmentation produced by this method is heavily affected by noise and errors. A single error is almost always enough to warrant a new segment. Additionally, it is challenging to fine-tune the threshold for a change point, meaning we must change it for each new dataset.

Algorithm 3 Sliding Window Segmentation

Iteratively increases the size of a window until the resulting approximation is worse than the previous iteration. Then cuts and begins a new segment.

Input: D - data, w - window size

```
1: procedure SegmentData(D, w)
2:     S ← []                                  ▷ Initialize an empty list for segments
3:     s ← 1                                   ▷ Initialize the start index
4:     for e ← w to len(D) do                  ▷ Loop through the data
5:         if PurityDecreases(D[s : e]) then   ▷ Check if purity decreases in the window
6:             CutSegment(S)                   ▷ Cut the segment and append to S
7:             s ← e + w                       ▷ Update the start index
8:         end if
9:     end for
10:    CutSegment(S)                           ▷ Cut the final segment
11:    return S                                ▷ Return the list of segments
12: end procedure
```

Bottom Up

The second method we looked at is the bottom-up approach. It starts with small segments (Line 3 in Algorithm 4) and iteratively merges adjacent segments (Line 17 in Algorithm 4) based on predefined criteria (Line 9-10 in Algorithm 4) until we form more significant, more homogeneous segments. This method helps capture both abrupt and gradual changes in data patterns. However, defining appropriate merging and stopping criteria is essential to avoid over-segmentation and achieve meaningful results. We based our merging criteria for the bottom-up method (Line 9 in Algorithm 4) on the capacity for the trend predicted on one segment to predict values on the other. The threshold for this change is a hyper-parameter that must be tuned accordingly, but it results in far more accurate segmentation. This parameter is chosen automatically by assessing the overall noise and size of the current segment; however, it is significantly slower than the sliding window method.

Top Down

The last standard method we look at is the top-down approach. Starting with one large segment (Line 3 in Algorithm 5) , we assess whether a split is needed (Line 4,6-8,18-21) in Algorithm 5)and continue to do so until we reach our stopping criteria. We base the decision to split on the break p_b with the maximum gain g_b. Picking where to split, however, is a significant endeavor requiring N comparisons in general. The threshold for splitting is somewhat arbitrary. As a rule of thumb, we continue until the noise estimate is less than double the estimate from the preprocessing step we call this the noise threshold $\mathcal{N}$. We can tune this threshold further by setting a gain threshold for introducing a split; however, this is only sometimes a significant improvement.

Algorithm 4 Bottom-Up Segmentation

Starts with a set of n-1 short segments. Iteratively merges until some merge score threshold is reached.

Input: D - data, T - threshold

```
1:  procedure BottomUpSegmentation(D, T)
2:      Initialize S with small initial segments
3:      while S can be merged do                ▷ Check if segments can be merged
4:          p_b ← None                          ▷ Best merge pair
5:          q_b ← −∞                            ▷ Best merge score
6:          for s1 in S do                      ▷ Iterate through segments
7:              for s2 adjacent to s1 do
8:                  q ← CalculateMergeScore(s1, s2)
9:                  if q > q_b then
10:                     p_b ← (s1, s2)          ▷ Update best merge pair
11:                     q_b ← q
12:                 end if
13:             end for
14:         end for
15:         if q_b ≥ T then                     ▷ Threshold check
16:             MergeSegments(p_b)
17:         else
18:             break
19:         end if
20:     end while
21:     return S
22: end procedure
```

Algorithm 5 Top-Down Segmentation

Starts with one large segment, iteratively splits until some split score threshold is reached

Input: S - data, $\mathcal{N}$ - noise threshold, Q - score threshold, P - breaks

procedure TopDownSegmentation($S, \mathcal{N}, Q, P = []$)
  Initialize one large segment S covering the entire dataset
  while stopping criteria not met **do**
    $MAE =$ FitMAE(S) ▷ Estimate MAE
    if $MAE < \mathcal{N}$ **then**
      break ▷ Stop if noise is within acceptable range
    end if
    $p_b \leftarrow$ None ▷ Best split point
    $q_b \leftarrow 0$ ▷ Best split gain
    for each $p \in S$ **do**
      $q \leftarrow$ CalculateSplitGain(S, p)
      if $q > q_b$ **then**
        $p_b \leftarrow$ point
        $q_b \leftarrow q$
      end if
    end for
    if $q_b \geq Q$ **then**
      ▷ TDS is TopDownSegmentation
      return TDS($S[: p_b], \mathcal{N}, Q, P$) +$[p_b]$+ TDS($S[p_b :], \mathcal{N}, Q, P$)
    else
      return P ▷ Return the list of segments
    end if
  end while
end procedure

4.3.2 SWAB Segmentation

Our first non-standard segmentation scheme is Sliding Window and Bottom-up (SWAB) segmentation. SWAB is an attempt to reduce the significant increase in cost associated with bottom-up segmentation without compromising accuracy.

The general outline for this segmentation starts with a sliding window covering a significant portion of the data, ideally larger than your largest segment. We then perform bottom-up segmentation over this window (Line 6 in Algorithm 6), keeping the leftmost segment and moving the window to its rightmost edge (Line 5,8 in Algorithm 6). We repeat this process until the window reaches the end of the dataset (Line 4 in Algorithm 6). At this point, a final bottom-up segmentation is performed over the entire dataset to merge any segments larger than the initial window size.

The pseudocode for this method is as follows:

SWAB, while seeking to reduce computational costs, comes with several weaknesses that we must consider. One crucial aspect is the selection of the sliding window size, which can significantly impact the quality of the segmentation results. If the window size is too small, it might miss essential changes or patterns that occur over a larger span. At the same time, a too-large window may lead to over-segmentation, overlooking subtle changes within segments.

Additionally, the SWAB approach independently performs bottom-up segmentation within each sliding window, potentially missing larger-scale patterns that span multiple windows and limiting its ability to capture global information. The sliding window movement can also introduce boundary effects, affecting the quality of segments near the window edges. Overlapping segments can emerge as the window moves, leading to redundant segmentation, while the final merging step might only partially capture the actual relationships between segments. Moreover, the sensitivity of the segmentation results to the window placement can reduce stability and reproducibility.

Lastly, despite its attempts to reduce computational costs compared to traditional

Algorithm 6 SWAB

Starts with a large window, performing bottom-up segmentation on it. Take the leftmost resulting segment and start a new window from its rightmost point. Continue until the window reaches the end of the dataset.

Input: D - data, w - window size

1: **procedure** SWAB(D, w)
2: $P = []$
3: $s = 0$
4: **while** $e < len(D)$ **do**
5: $e = min(s + w, len(D))$
6: $S = BottomUp(D[s : e])$ ▷ From Algorithm 4
7: $P.append(S[0])$
8: $s = S[0]$
9: **end while**
10: **return** $BottomUp(P)$
11: **end procedure**

bottom-up segmentation, SWAB still requires multiple iterations over the data, which can be demanding for large datasets or complex merging criteria. As such, careful parameter tuning and consideration of the data's characteristics are crucial.

4.3.3 kNN-Based Classifier

We also developed a novel kNN-based classification model based on ClaSP [6]. Since a change point is a local phenomenon, its kNN should best indicate its potential. The classifier provides a score in [0, 1], which suggests the probability of it being the best breakpoint. We then select the global maximum as a breakpoint, repeating the process recursively on each segment. The process is the same as a top-down segmentation, except we compute the loss once we complete the segmentation.

The approach is heuristic, so we can significantly improve the model's success using a multi-shot system. The classifier still significantly reduces time complexity over other segmentation schemes and should still result in log-linear scaling.

Algorithm 7 Recursive Top-Down Segmentation with kNN-based Classifier

Iteratively segments data using a classifier score until some threshold is reached.

Input: D - data, T - threshold

procedure SegmentationWithClassifier(D, T)
    $P = []$ ▷ Partition
    $kS \leftarrow$ ComputeKNNClassifierScores(D, classifier)
    $p_b \leftarrow$ FindGlobalMaximum(kS)
    if $kS[p_b] > T$ **then**
        $P \leftarrow \text{SWC}(D[: p_b], T) + \text{SWC}(D[p_b :], T)$
    end if
    return P
end procedure

Compute Classifier Score

The classifier is a simple neural network that encodes the vector of kNNs as a single value. We split the gradient calculation over the resulting output of each training example, with the loss being the ratio of the MAE to the noise term of the resulting model.

Algorithm 8 Training a Classification Score based on MAE/Noise Ratio

Procedure for training our classification score using unlabelled training data

Input: D - training data, C - classifier, H - hyperparameters

procedure TrainClassificationScore(D, C, H)
    Initialize model parameters randomly or with pre-trained values ▷ Initialize model
    for e in 1 to num_epochs **do** ▷ Loop over epochs
        Shuffle the training data ▷ Shuffle data
        for x in D **do** ▷ Loop over examples in training data
            $P \leftarrow$ SegmentationWithClassifier(x, C) ▷ Segmentation
            $M \leftarrow$ ATD(x, P) ▷ Model
            $\hat{y} \leftarrow M(X)$ ▷ Predicted output
            $MAE \leftarrow$ MAE($\hat{y}, y$) ▷ Mean Absolute Error
            $NR \leftarrow$ ComputeNoiseRatio($\hat{y}, y$) ▷ Noise Ratio
            $L \leftarrow$ MAEToNoiseRatio(MAE, NR) ▷ Loss
            $\nabla L \leftarrow \nabla$MAEToNoiseRatio($L$) ▷ Gradient
            $C.parameters \leftarrow C.parameters - H.\text{learning\_rate} \times \nabla L$ ▷ Update
        end for
    end for
    return Trained classification score model ▷ Return trained model
end procedure

The result is a top-down classifier that considers the entire segmentation rather than

only the current split. Using the classifier's top-k suggestions is also possible, but the resulting accuracy improvement could be more impressive, and the stop conditions could be more precise. With the top-down method, we reset if no value in the classifier reaches some threshold. An alternative approach also uses the maximum deviation from the mean classification score as the indicator, but we needed help finding a consistent method for determining the threshold in advance.

Experimental Validation

In our experiments, we compared the performance of our multi-shot binary classifier against three other segmentation schemes and a naive solution, particularly for smaller datasets. Our results demonstrate the binary classifier's superiority in terms of speed and accuracy. This novel approach streamlines the segmentation process while enhancing the reliability and objectivity of trend discovery, marking a significant advancement in our methodology.

4.3.4 Recap

In conclusion, this methodology chapter presents a comprehensive framework for trend discovery and data segmentation. We have explored various approaches to fitting trends and adapting to data characteristics. Our methodology addresses clean data handles noisy datasets, and identifies Approximate Trend Dependencies (ATDs) caused by outliers and anomalies. This adaptability ensures robust trend discovery in diverse scenarios.

Moreover, our investigation into segmentation methods offers valuable insights. We have examined standard techniques, such as sliding window, bottom-up, and top-down approaches, alongside a novel SWAB segmentation scheme. Additionally, we introduced a kNN-based binary classifier, which significantly streamlines the segmentation process while maintaining accuracy. Our experiments demonstrate its superior performance in speed and reliability, marking a noteworthy advancement in data segmentation.

Overall, this methodology equips us with a versatile toolkit for robust trend discovery and segmentation, capable of handling a wide range of data challenges. These methods lay the foundation for our subsequent analysis and provide a solid framework for extracting meaningful insights from complex datasets.

Chapter 5

Experimental Results

We structure this chapter into four sections, each contributing to our comprehensive evaluation:

- Section 5.1 introduces the datasets and metrics used for our experiments.
- Section 5.2 evaluates the performance of our models in trend discovery on a single segment.
- Section 5.3 compares our sampling schemes and assesses the improvements in time complexity
- Section 5.4 extends the evaluation to conditional trend discovery.

Through these sections, we aim to provide a detailed analysis of our experimental findings and their significance in addressing the challenges of trend dependency discovery.

This chapter presents the experimental evaluation of our model's scalability, accuracy, and cleaning capacity. Our research aims to address [briefly mention the research problem/objective]. We conduct experiments on a machine equipped with an Intel 6-core 2.6GHz CPU and an NVIDIA RTX 3070Ti GPU with 8GB of VRAM and 32GB of RAM.

5.1 Experimental Methodology and Data Sources

We perform our experiments over four datasets: two synthetic and two from the real world.

1. **COVID-19 (covid):** We used the OWID COVID-19 dataset, containing 246 country and region-specific data for deaths, cases, tests, and more for a total of over 1k unique tables with up to 1k records each. We employed this dataset to explore detection across multiple attributes.

2. **Historical Stock Prices (stock):** This dataset comprises daily historical data from over 8k stocks with ETFs on the NYSE and NASDAQ from 1986 until 2018. Each is broken into Open, High, Low, Close, and Volume. Giving over 40k unique tables with up to 11k tuples each.

3. **Synthetic Data (syn-im):** We generated synthetic data with arbitrary data points using tokens from the model vocabulary. We added noise from a normal distribution with random variance. Tables were generated as needed with both single and multiple generating functions. Each table contained between 1K and 10M records.

4. **Out of Model Synthetic Data (syn-oom):** Similar to **syn-im**, this dataset contained synthetically generated data, but we used out-of-model tokens (e.g., $x^n, \log_n$) as basis functions.

The two synthetic datasets differ because the first (*syn-im*) uses only in-model tokens for generation. In contrast, the second (*syn-oom*) can include tokens beyond the model scope. This distinction aims to ensure the model generalizes to more complicated trends where the underlying function is unclear.

We took the real-world datasets from the OWID COVID-19 [25] and the Kaggle Huge Stock Market Dataset. These datasets are used only for detecting conditional

trend dependencies at small scales. We use the **syn-im** dataset exclusively at larger scales.

We base our methodology for accuracy measurements on SR Bench [19], replacing the R^2 metric with MAE and neglecting the semantic similarity metric entirely. The switch to MAE provides a better comparison with our noise estimation systems. We ignore the semantic similarity metric as we are only concerned with the capacity to represent data and not with extracting specific function classes, especially in the case of *syn-oom*. However, we used the semantic similarity method for the training process in symbolic regression.

5.2 Trend Discovery

In this section, we focus on the crucial task of trend discovery and its evaluation through experiments. We begin by assessing the scalability and accuracy of our regression models. Experiment 1 explores scalability without segmentation, measuring runtime performance on the *syn-im* and *syn-oom* datasets. Experiment 2 delves into accuracy, using real-world COVID-19 data to assess error detection and its sensitivity to error degrees. By examining trends within single data segments, we gain insights into our models' performance under varying conditions. These experiments lay the foundation for understanding how our models handle real-world data and their effectiveness in identifying trends.

5.2.1 Experiment 1: Scalability - Regression

The assessment of scalability in trend discovery represents a pivotal aspect of our research, bearing direct implications for the practical utility of our models. In this experiment, we scrutinize the runtime performance of our trend discovery methodologies, focusing on the absence of segmentation. This omission allows us to evaluate the inherent capabilities of our models when confronting data trends without predefined partitioning. We consider

two datasets, namely, the **syn-im** and **syn-oom**, allowing us to assess how the model behaves with expected and unexpected trends.

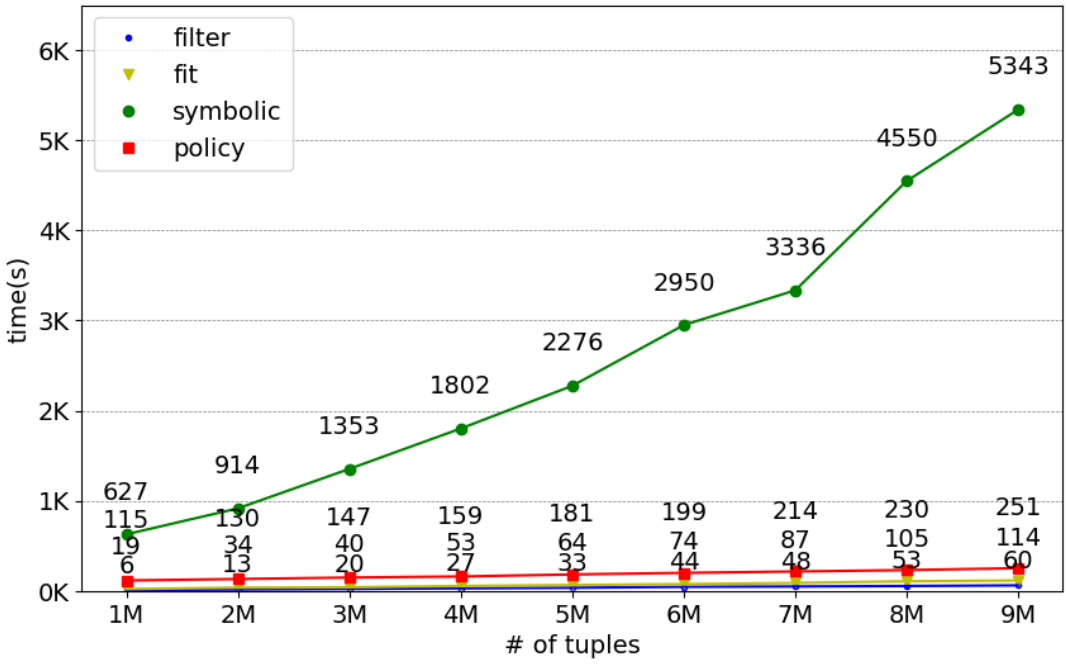

Figure 5.1: Regression Scalability (*syn-im*)

This experiment serves as an indispensable benchmark for comprehending how our models navigate the landscape of trend discovery within real-world scenarios. By quantifying the time required to execute these tasks, we gain insights into the practicality and feasibility of our models across diverse applications. The observed temporal patterns also offer insights into model convergence dynamics, while the scaling characteristics unveil aspects of their computational efficiency.

Of note is the consistent scaling patterns exhibited across diverse models, typically aligning within log-linear to quadratic scaling. It is noteworthy that the symbolic regression model demonstrates a discernibly elevated baseline time compared to the policy generator, highlighting the interplay between model complexity and computational efficiency. This experiment is a foundational cornerstone, providing insights for subsequent investigations and model refinement in trend discovery tasks.

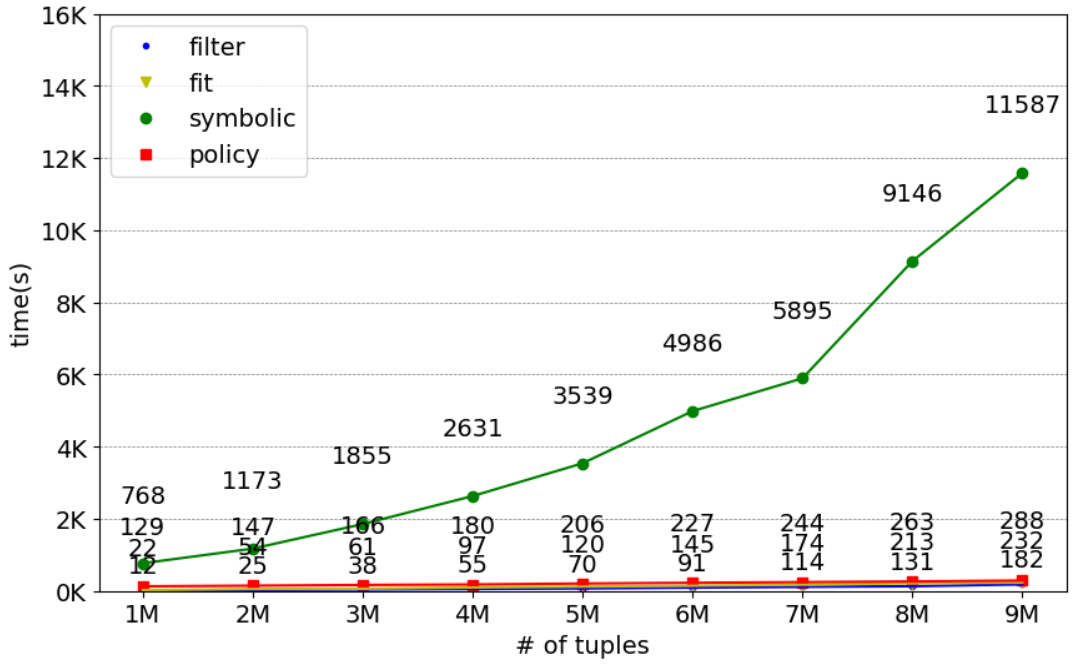

Figure 5.2: Regression Scalability (*syn-oom*)

5.2.2 Experiment 2: Accuracy - Regression

In this experiment, we delved into the critical aspect of accuracy in trend discovery, mainly focusing on regression models. To gauge the accuracy of our models, we conducted an exhaustive assessment across cumulative COVID-19 datasets, encompassing metrics related to testing, cases, and deaths on a global scale. Our accuracy assessment hinges on calculating a fundamental ratio - the relationship between the noise estimate and the Mean Absolute Error (MAE) derived from our model's predictions. It is essential to note that lower values in this ratio signify enhanced accuracy.

Our investigation included comparing error detection rates across three distinct error degrees, with Found(1.25) as a benchmark. The significance of Found(1.25) lies in its implication of a deviation 25 percent greater than the maximum noise value, making it a crucial reference point for our evaluation.

The graphical representations in Figure 5.3 and Figure 5.4 provide insightful visualizations of our findings. Figure 5.3 illustrates the fit error as a fraction of the noise

estimate, shedding light on the precision of our models in capturing the underlying trends in the data. Figures 5.4, on the other hand, offers a detailed view of the error detection rates across varying noise levels, providing a nuanced perspective on how well our models identify and classify errors.

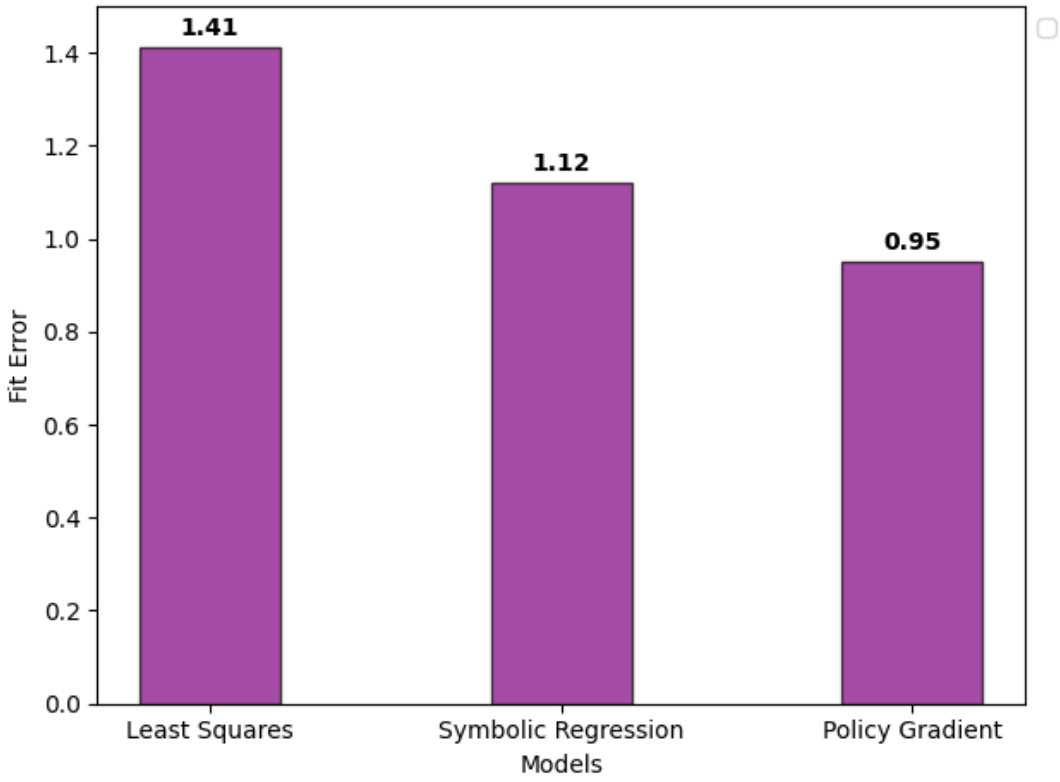

Figure 5.3: Fit Error as a Fraction of Noise Estimate

Among the notable outcomes of this experiment is the exceptional performance of the policy gradient model in terms of accuracy. Symbolic regression also emerges as a strong contender, surpassing least squares in error detection capabilities. However, it's essential to acknowledge that symbolic regression, while demonstrating robust accuracy, often demands extensive sampling or substantial computational resources. This observation underscores the trade-offs between accuracy and computational complexity, a crucial consideration in the practical applications of our models.

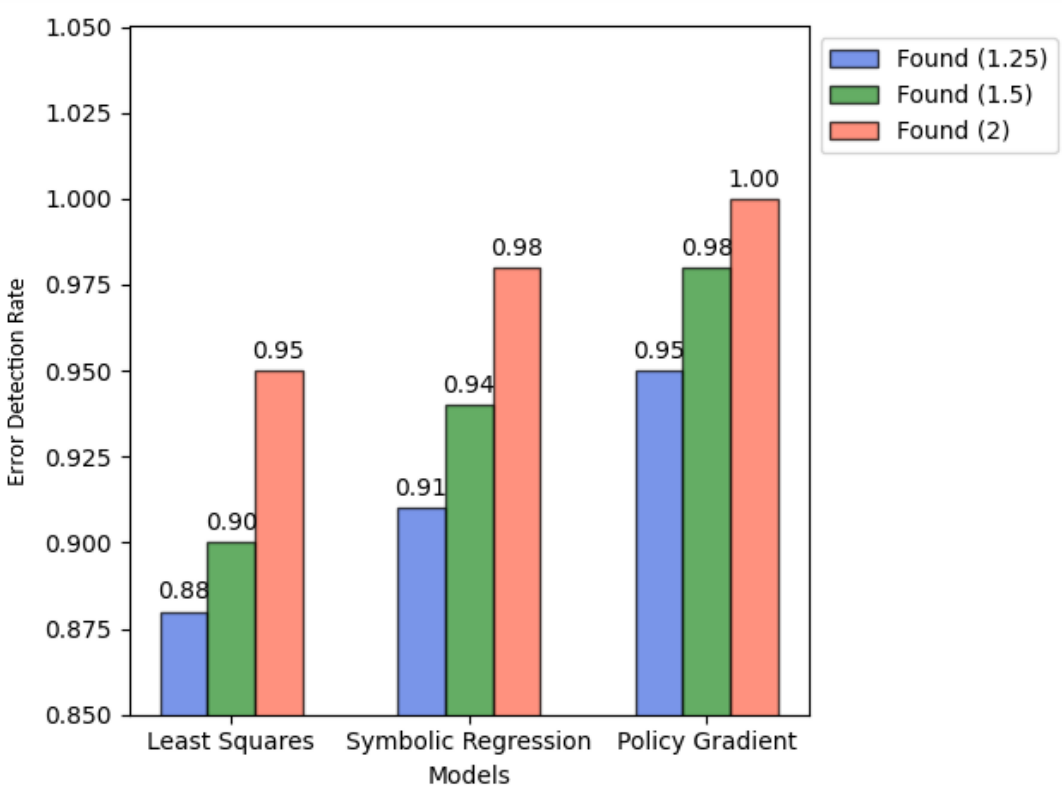

Figure 5.4: Error Detection Rate at Different Noise Levels

5.3 Filters and Sampling

This section delves into experiments designed to optimize our trend dependency discovery process. Each investigation focuses on a crucial aspect of this process, ensuring efficiency and accuracy.

Experiment 3: Candidate Error Detection assesses the accuracy of our initial candidate error detection mechanism. **Experiment 4: Sampling Effectiveness** explores various sampling schemes' efficacy in influencing the trend dependency discovery process's accuracy. Finally, **Experiment 5: Sampling Scalability** measures the time complexity of our trend discovery methods across different sample sizes, providing insights into scalability. These experiments collectively enhance the precision and efficiency of our trend dependency discovery approach.

5.3.1 Experiment 3: Candidate Error Detection

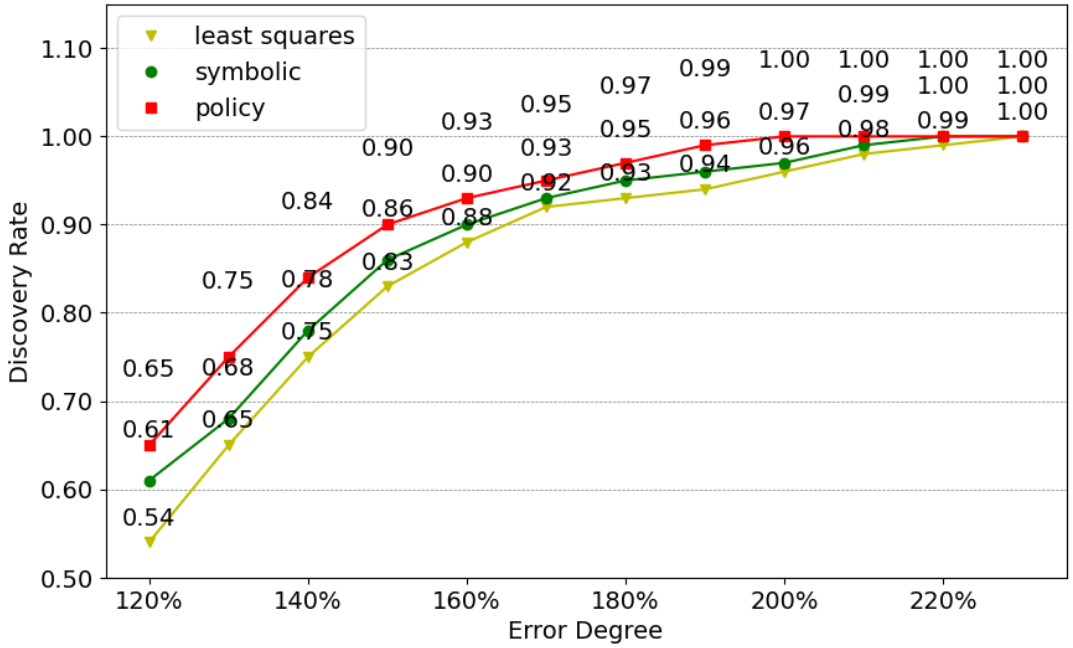

Figure 5.5: Error Discovery Rate vs. Error Degree

In this experiment, we introduced anomalous tuples into the COVID-19 dataset and examined the effectiveness of our initial filter in identifying these anomalies based on their error degree. This test focuses solely on the initial detection phase, evaluating whether we successfully identify the anomalies at this early stage before any subsequent processing within the trend dependency discovery process.

This experiment aims to assess the accuracy of our candidate error detection mechanism rigorously. We sought to determine how effectively our models can discern anomalies in the presence of varying error degrees.

Our findings indicated that all models exhibited similar accuracy in detecting anomalies, with the policy gradient model slightly outperforming the regression models as shown in Figure 5.5. However, a notable proportion of missed anomalies resulted from setting too high an initial error detection threshold, mainly when consecutive errors of similar degree occurred. The fact that missed anomalies usually occur under these scenarios sug-

gests that fine-tuning the error threshold could substantially reduce these instances in practical applications. The experiment outcomes offer valuable insights into optimizing the error detection process for more precise trend dependency discovery.

5.3.2 Experiment 4: Sampling Effectiveness

In Experiment 4, we explored the effectiveness of various sampling schemes in influencing the accuracy of our trend dependency discovery process. The symbolic regression model was applied to the **syn-im** dataset with a substantial sample size of $100,000$. We examined three distinct sampling strategies:

Random Sampling (Evenly): This scheme selected data points randomly, ensuring an even distribution across the dataset.

Minimum Distance from Smoothing: Data points were chosen based on their proximity to the smoothed trend, aiming to capture significant fluctuations in the data.

Minimum Slope Difference: Sampling focused on data points with the most negligible differences in slope, emphasizing areas of potential trend changes.

Our analysis, as shown in Figure 5.6, revealed that, with our chosen dataset, we maintained a low loss even at very high sampling rates. This finding underscores the effectiveness of sampling as a strategy for trend dependency discovery. By employing appropriate sampling schemes, we can substantially reduce the computational demands of the trend discovery process while preserving high accuracy. Next, we measure the degree to which this sampling can reduce our computational need.

5.3.3 Experiment 5: Sampling Scalability

This experiment focuses on assessing the time complexity of our trend discovery methods. We conducted a series of measurements to evaluate the runtime performance of our symbolic regression model under various sample sizes. We performed these tests on the **syn-im** dataset, with a sample size from 100k to 900k. It's worth noting that we excluded

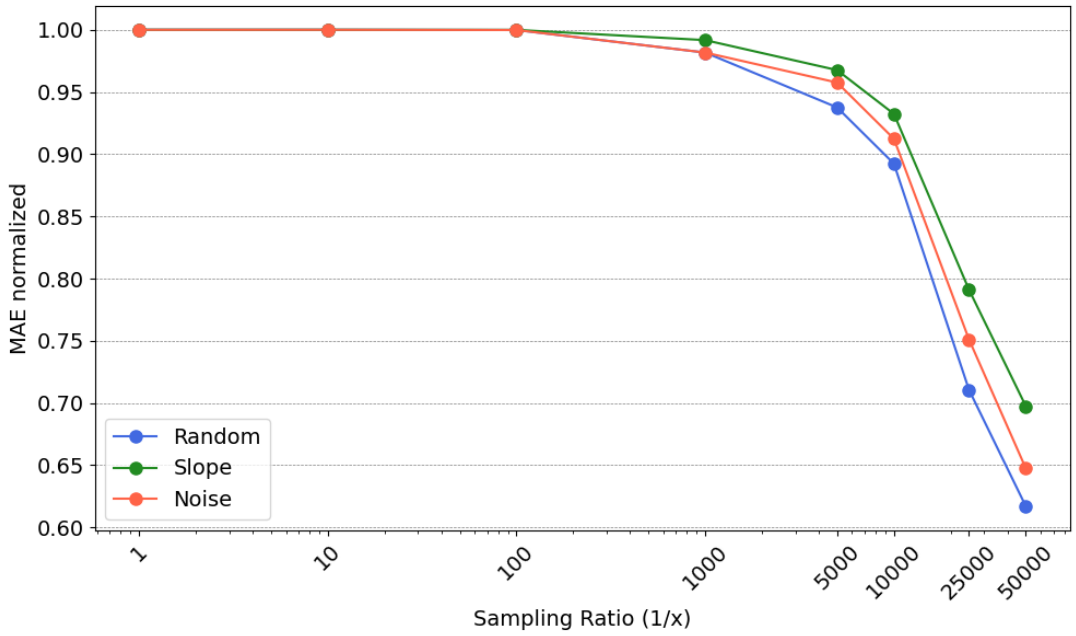

Figure 5.6: Effect of Different Sampling Schemes on MAE

the policy generator from these assessments as it already samples by default.

The results, depicted in Figure 5.7, clearly visualize how different sample sizes affect the time required to complete our trend discovery tasks. As expected, reducing the input size resulted in improved time efficiency. Notably, the baseline time for our models corresponds to at least the duration of the filtering step.

This experiment contributes valuable insights into the relationship between sample size and time complexity, guiding our approach to scalable trend dependency discovery.

5.4 Conditional Trend Discovery

In pursuit of a more comprehensive understanding of trend dependency discovery, we extended our experimental scope to encompass segmentation techniques coupled with regression models. What drove this expansion was the need to evaluate the performance of our models in scenarios where data subsets exhibit distinct trends. By introducing segmentation, we aimed to assess the models' adaptability and effectiveness in identifying

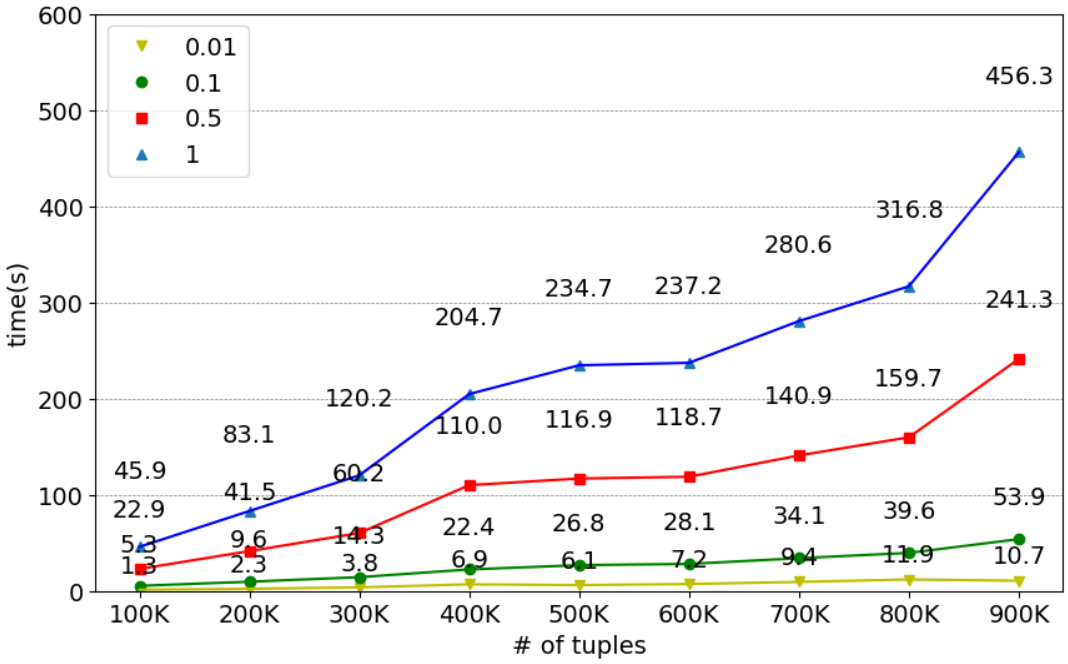

Figure 5.7: Sampling Effects on Time Complexity

dependencies within segmented data, a common real-world scenario.

The minimal perceived advantage in segmented contexts justifies the exclusion of the policy gradient model from this analysis. This decision allowed us to focus on models that could benefit more from segmentation, contributing valuable insights into the interplay between segmentation strategies and trend discovery within distinct data subsets. This experiment is crucial in bridging the gap between theoretical model capabilities and their practical applicability in real-world scenarios where data often exhibits multifaceted trends.

5.4.1 Experiment 6: Scalability - Segmentation

We conducted Experiment 6 to assess the scalability of our segmentation methods under different schemes. In this experiment, we focused on measuring the time required to perform data segmentation using three distinct approaches: standard bottom-up, SWAB, and classifier-based methods. The datasets used for testing encompassed various con-

texts, including financial data represented by the *stock* dataset and public health data defined by the *covid* dataset. Additionally, we employed the *syn-im* dataset for testing at high volume.

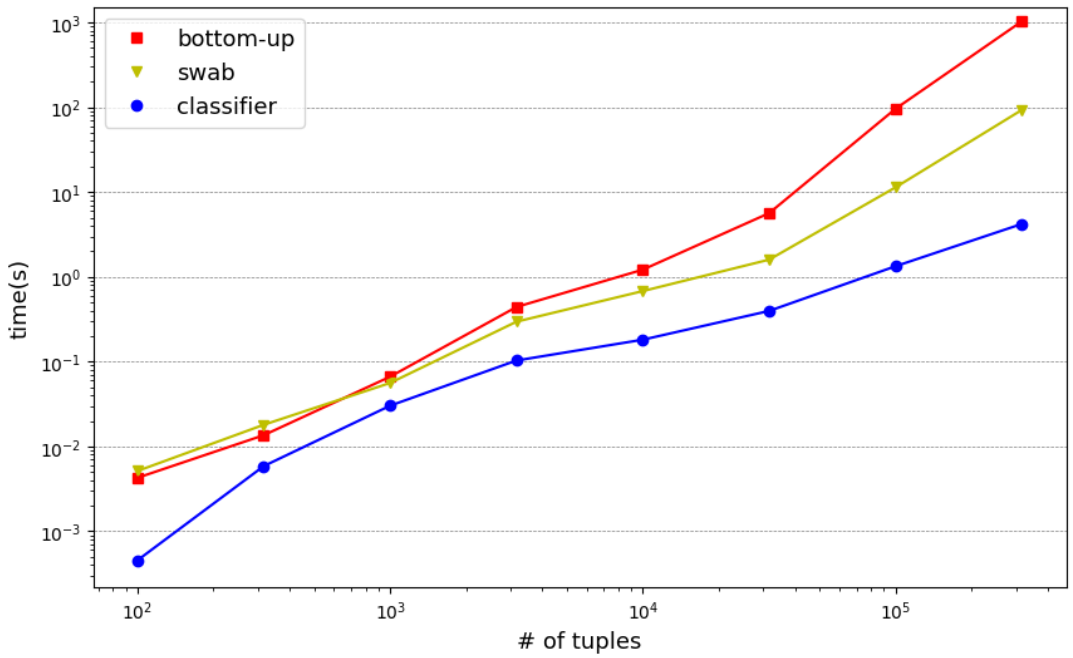

Figure 5.8: Segmented Trend Discovery Time Scaling

The rationale behind this experiment was to investigate our segmentation methods' computational efficiency, particularly in handling diverse datasets. Our results are summarized in Figure 5.8. As expected, standard bottom-up segmentation exhibited less favorable scaling behavior, aligning with prior field knowledge. However, the notable finding was that classifier-based segmentation consistently outperformed both SWAB and traditional bottom-up approaches. We attributed this superiority to the linear relationship between the classifier-based method and candidate breakpoints, even when no prior training data was available (zero-shot classification). This robust performance across different datasets and complexities highlights the potential for classifier-based segmentation to enhance the efficiency of trend dependency discovery processes.

The results of Experiment 6 provide valuable insights into the scalability and adapt-

ability of segmentation strategies in trend analysis tasks. These findings are particularly relevant for researchers and practitioners seeking efficient approaches to handle diverse datasets while maintaining the accuracy and effectiveness of trend dependency discovery.

5.4.2 Experiment 7: Accuracy - Segmentation

These experiments focus on assessing the accuracy of our trend dependency discovery process when segmentation is employed. This experiment encompasses three sub-experiments, each addressing different facets of accuracy within the segmentation context.

As we delve into the results of these sub-experiments, we aim to discern which segmentation strategies and classifiers demonstrate superior accuracy. This investigation provides valuable insights into optimizing trend dependency discovery, mainly when segmentation is crucial.

Sub-Experiment 7.1: MAE Comparison - Segmentation

We compared the Mean Absolute Error (MAE) against the trend and the noise estimate as part of our accuracy assessment in the segmentation context. We analyzed this to evaluate how well our models' predictions aligned with trend and noise levels. These results, shown in Figure 5.9, were primarily derived from synthetic noise estimates since ground truth data was unavailable for real datasets.

Sub-Experiment 7.2: Breakpoint Discovery - Segmentation

This sub-experiment examined the ratio of discovered breakpoints concerning ground truth. Our focus was to measure our segmentation algorithms' effectiveness in accurately identifying breakpoints. This evaluation, summarized in Figure 5.10, provided insights into how well our algorithms captured data transition points.

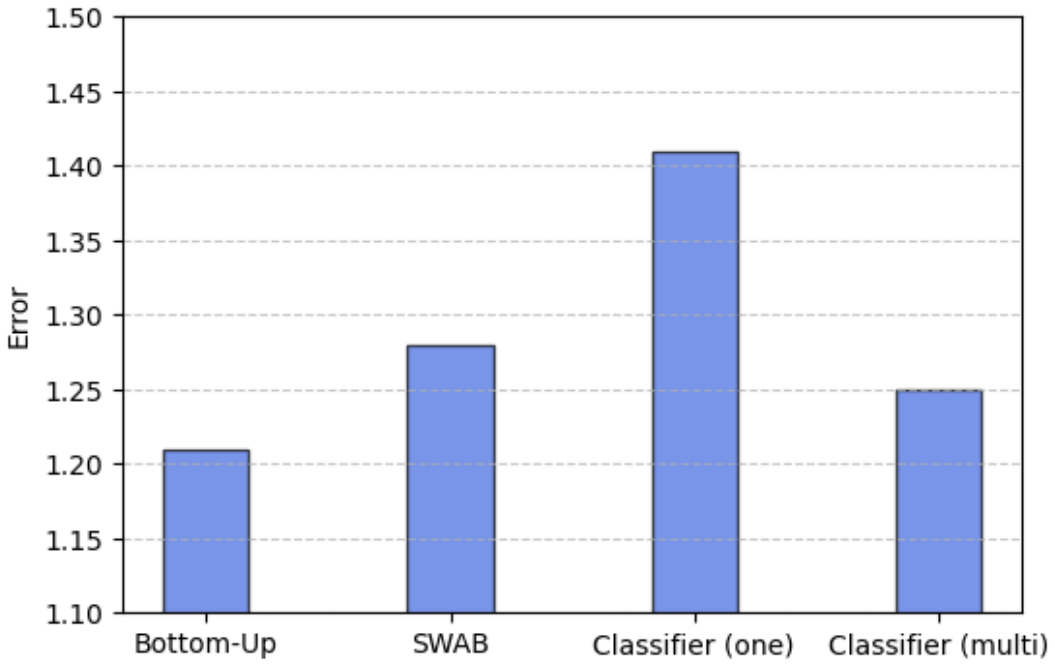

Figure 5.9: Mean Relative Error by Segmentation Algorithm

Sub-Experiment 7.3: Error Recovery - Segmentation

We investigated the number of errors accurately recovered by our segmentation algorithms. This analysis, shown in Figure 5.11, aimed to determine the models' capability to detect and recover errors within the segmented data. Specifically, we examined the performance of multi-shot, SWAB, and single-shot classifiers in this context.

In summary, the findings from this experiment underscore the significance of segmentation techniques and classifiers in the context of trend dependency discovery. The results reveal the multi-shot classifier's superiority in accuracy and breakpoint detection, making it a valuable choice for enhancing the precision of trend analysis. Its effectiveness in identifying trend transition points and errors within segmented data reduces false positives and improves the overall reliability of trend analysis outcomes.

Conversely, the SWAB segmentation strategy demonstrates its capacity to enhance data segmentation correctness but has a limited impact on improving data fitting accuracy. These observations emphasize the inherent trade-offs in segmentation approaches

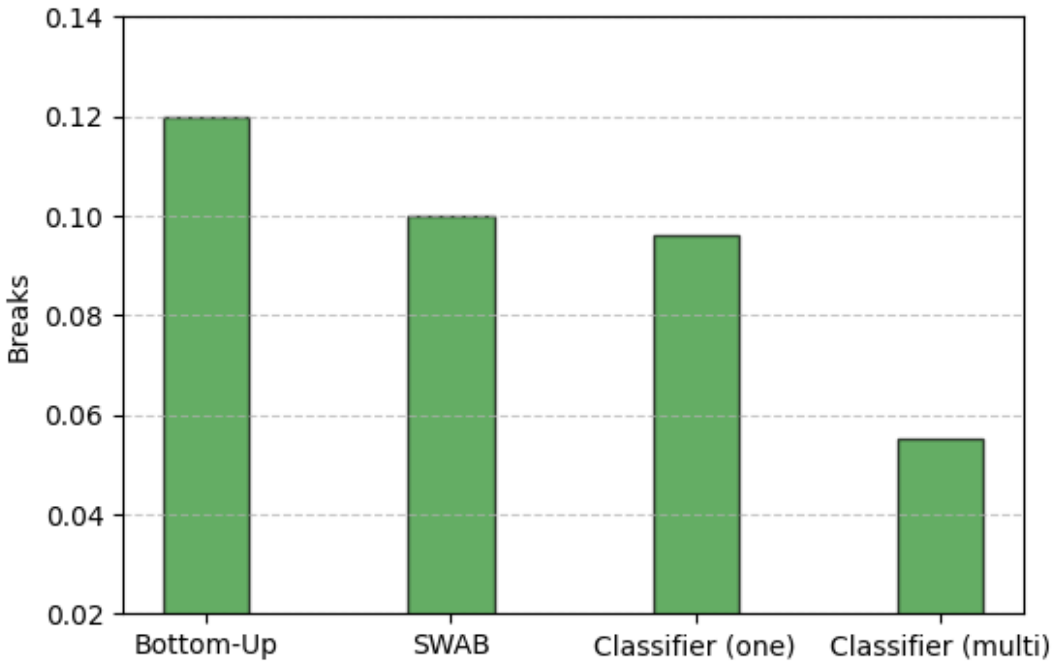

Figure 5.10: Ratio of Discovered Breakpoints vs Ground Truth

and the need for careful consideration when selecting the most suitable strategy and classifier for specific trend analysis tasks, particularly concerning computational resources and objectives. Researchers and practitioners should thoroughly assess their requirements to make informed choices that balance accuracy and efficiency in the trend dependency discovery process.

In conclusion, our series of experiments aimed to comprehensively evaluate the performance of our trend dependency discovery models and associated techniques.

Experiment 1 highlighted the crucial aspect of scalability in trend discovery, showcasing the challenges of increasing dataset sizes and emphasizing the need for efficient methods. Experiment 2 demonstrated our models' remarkable accuracy in detecting trends and provided insights into their potential applications, with symbolic regression showing great promise despite computational demands. Experiment 3 focused on candidate error detection, reaffirming the models' competence in identifying anomalies while underscoring the importance of setting appropriate error thresholds.

Figure 5.11: Number of Errors Accurately Recovered

Experiments 4 and 5 delved into the efficacy of sampling, revealing its potential to significantly reduce computational demands without compromising accuracy. Experiment 6 explored segmentation's impact on scalability, highlighting its advantages in specific scenarios. Finally, Experiment 7, centered on conditional trend discovery, showcased the effectiveness of multi-shot classifiers.

In summary, our findings underline the versatility and efficiency of our trend dependency discovery models, emphasizing their potential in real-world applications across varying dataset sizes and complexities.

Chapter 6

Conclusions

6.1 Conclusion

In conclusion, this thesis has explored and presented a comprehensive examination of data quality constraints and dependencies, focusing on the introduction of 'trend dependencies' (TDs) as a novel addition to this domain. The thesis has advanced our understanding of how data quality can significantly impact the performance of data-driven models.

Throughout our investigation, we have empirically demonstrated the practicality and effectiveness of TDs in various aspects of data analysis. While TDs will not replace existing constraints like functional dependencies (FDs) or order dependencies (ODs), they offer a valuable alternative in scenarios where data exhibits complex, non-linear relationships.

The experiments conducted in this thesis have emphasized the scalability, efficiency, and effectiveness of TDs in diverse applications. They have proven their worth in error detection and data cleaning, especially when dealing with complex, real-world datasets.

Furthermore, exploring different sampling schemes has highlighted how trend discovery can be significantly optimized, reducing computational demands while preserving accuracy.

This thesis has contributed to the data quality constraint landscape by introducing TDs as a pragmatic and versatile solution. These findings offer a valuable addition to the toolkit of data analysts, researchers, and practitioners working in data-driven modeling, facilitating more accurate and reliable analyses of complex datasets. As data plays a pivotal role in various domains, introducing TDs represents a significant step towards improving data quality and enhancing the performance of data-driven models in practical applications.

6.2 Future Work

This section outlines potential avenues for future research and development in trend dependencies (TDs) and their applications.

6.2.1 Multi-Attribute Dependency Discovery

One promising direction for future work involves extending the TD discovery process to encompass multiple attributes simultaneously. This expansion could include starting from scratch to identify TDs between various attribute combinations and leveraging previously discovered TDs to predict new dependencies between other features. By exploring multi-attribute TDs, we aim to provide a more holistic understanding of complex relationships within datasets, enabling more comprehensive data analysis and modeling.

6.2.2 Unified Architecture for Regression and Segmentation

Another area of exploration is the possibility of unifying the regression and segmentation steps into a single architecture rather than treating them as separate problems with overlapping preprocessing steps. This integration could streamline the trend discovery process, improving efficiency and scalability. Investigating the development of a unified framework that seamlessly combines regression and segmentation for TD discovery

represents an exciting avenue for future research.

These future directions underscore the ongoing potential for innovation and advancement in TDs. Addressing these research areas as data-driven models and analyses evolve can further enhance our ability to extract valuable insights from complex datasets, ultimately contributing to more robust and accurate data-driven decision-making processes.

Bibliography

[1] Douglas Adriano Augusto and Helio JC Barbosa. Symbolic regression via genetic programming. In *Proceedings. Vol. 1. Sixth Brazilian symposium on neural networks*, pages 173–178. IEEE, 2000.

[2] Jushan Bai and Pierre Perron. Computation and analysis of multiple structural change models. *Journal of applied econometrics*, 18(1):1–22, 2003.

[3] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pages 936–945. PMLR, 2021.

[4] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *2007 IEEE 23rd international conference on data engineering*, pages 746–755. IEEE, 2006.

[5] Qianqian Chen. *Fast Grid Search Algorithms for Multi-phase Regression Models*. University of Washington, 2020.

[6] Arik Ermshaus, Patrick Schäfer, and Ulf Leser. Clasp: parameter-free time series segmentation. *Data Mining and Knowledge Discovery*, 37(3):1262–1300, 2023.

[7] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Transactions on Database Systems (TODS)*, 33(2):1–48, 2008.

[8] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering*, 23(5):683–698, 2010.

[9] Erich Fuchs, Thiemo Gruber, Jiri Nitschke, and Bernhard Sick. Online segmentation of time series based on polynomial least-squares approximations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2232–2245, 2010.

[10] A Ronald Gallant and Wayne A Fuller. Fitting segmented polynomial regression models whose join points have to be estimated. *Journal of the American Statistical Association*, 68(341):144–147, 1973.

[11] Andrew Gelman and Guido Imbens. Why high-order polynomials should not be used in regression discontinuity designs. *Journal of Business & Economic Statistics*, 37(3):447–456, 2019.

[12] Lukasz Golab, Howard Karloff, Flip Korn, Divesh Srivastava, and Bei Yu. On generating near-optimal tableaux for conditional functional dependencies. *Proceedings of the VLDB Endowment*, 1(1):376–390, 2008.

[13] Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999.

[14] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

[15] Reza Karegar, Parke Godfrey, Lukasz Golab, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. Efficient discovery of approximate order dependencies. *arXiv preprint arXiv:2101.02174*, 2021.

[16] Reza Karegar, Melicaalsadat Mirsafian, Parke Godfrey, Lukasz Golab, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. Discovering domain orders via order dependencies. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1098–1110. IEEE, 2022.

[17] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining*, pages 289–296. IEEE, 2001.

[18] Samuel Kim, Peter Y Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE transactions on neural networks and learning systems*, 32(9):4166–4177, 2020.

[19] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. *arXiv preprint arXiv:2107.14351*, 2021.

[20] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pages 5979–5989. PMLR, 2021.

[21] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[22] Pei Li, Jessica Jessica, Naida Tania, Michael Böhlen, Divesh Srivastava, and Jaroslaw Szlichta. abcod: Mining band order dependencies. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 3162–3165. IEEE, 2022.

[23] Pei Li, Jaroslaw Szlichta, Michael Bohlen, and Divesh Srivastava. Discovering band order dependencies. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1878–1881. IEEE, 2020.

[24] Pei Li, Jaroslaw Szlichta, Michael Böhlen, and Divesh Srivastava. Abc of order dependencies. *The VLDB Journal*, 31(5):825–849, 2022.

[25] Edouard Mathieu, Hannah Ritchie, Lucas Rodés-Guirao, Cameron Appel, Charlie Giattino, Joe Hasell, Bobbie Macdonald, Saloni Dattani, Diana Beltekian, Esteban Ortiz-Ospina, and Max Roser. Coronavirus pandemic (covid-19). *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[26] Vito MR Muggeo. Interval estimation for the breakpoint in segmented regression: A smoothed score-based approach. *Australian & New Zealand Journal of Statistics*, 59(3):311–322, 2017.

[27] Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*, pages 821–833, 2016.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[29] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering math-

ematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*, 2019.

[30] Hyunju Son and Youyi Fong. Fast grid search and bootstrap-based inference for continuous two-phase polynomial regression models. *Environmetrics*, 32(3):e2664, 2021.

[31] Jaroslaw Szlichta, Parke Godfrey, Lukasz Golab, Mehdi Kargar, and Divesh Srivastava. Effective and complete discovery of order dependencies via set-based axiomatization. *Proceedings of the VLDB Endowment*, 10(7), 2017.

[32] Jaroslaw Szlichta, Parke Godfrey, and Jarek Gryz. Fundamentals of order dependencies. *Proceedings of the VLDB Endowment*, 5(11), 2012.

[33] Jaroslaw Szlichta, Parke Godfrey, Jarek Gryz, and Calisto Zuzarte. Expressiveness and complexity of order dependencies. *Proceedings of the VLDB Endowment*, 6(14):1858–1869, 2013.

[34] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] Jan Žegklitz and Petr Pošík. Benchmarking state-of-the-art symbolic regression algorithms. *Genetic Programming and Evolvable Machines*, 22(1):5–33, 2021.

[37] Jinghui Zhong, Liang Feng, Wentong Cai, and Yew-Soon Ong. Multifactorial genetic programming for symbolic regression problems. *IEEE transactions on systems, man, and cybernetics: systems*, 50(11):4492–4505, 2018.

[38] Bin Zuo, Zhaolu Hou, Fei Zheng, Lifang Sheng, Yang Gao, and Jianping Li. Assessment of the running slope difference (rsd) t-test, a new statistical method for detecting climate trend turning. In *EGU General Assembly Conference Abstracts*, page 4065, 2020.