12-31-2021

# Long Term Predictive Modeling on Big Spatio-Temporal Data

Yong Zhuang
*University of Massachusetts Boston*

# LONG TERM PREDICTIVE MODELING ON BIG SPATIO-TEMPORAL DATA

A Dissertation Presented

by

YONG ZHUANG

Submitted to the Office of Graduate Studies,
University of Massachusetts Boston,
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2021

Computer Science Program

# LONG TERM PREDICTIVE MODELING ON BIG SPATIO-TEMPORAL DATA

A Dissertation Presented

by

YONG ZHUANG

Approved as to style and content by:

---

Wei Ding, Professor
Chairperson of Committee

---

Ping Chen, Associate Professor
Member

---

Nurit Haspel, Associate Professor
Member

---

Shafiqul Islam, Professor
Tufts University
Member

---

Dan Simovici, Professor
Member

---

Dan Simovici, Program Director
Computer Science Program

---

Mark Pomplun, Chairperson
Computer Science Department

# ABSTRACT

## LONG TERM PREDICTIVE MODELING ON BIG SPATIO-TEMPORAL DATA

December 2021

YONG ZHUANG

B.Eng., HARBIN ENGINEERING UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS BOSTON

Ph.D., UNIVERSITY OF MASSACHUSETTS BOSTON

Directed by: Professor Wei Ding, Professor

In the era of massive data, one of the most promising research fields involves the analysis of large-scale Spatio-temporal databases to discover exciting and previously unknown but potentially useful patterns from data collected over time and space. A modeling process in this domain must take temporal and spatial correlations into account. However, with the dimensionality of the time and space measurements increasing, the number of elements potentially contributing to a target sharply grows, making the target's long-term behavior highly complex, chaotic, highly dynamic, and hard to predict. Therefore, two different considerations are taken into account in this work: identifying the most relevant and meaningful features from the original Spatio-temporal feature space; and modeling complex space-time dynamics with sensitive dependence on initial conditions.

First, identifying strongly related features and removing the irrelevant or less important features to a target feature from large-scale Spatio-temporal data sets is a

critical and challenging issue in many fields. The optimal sub-feature-set containing all the valuable information is called the Markov Boundary. Unfortunately, the existing feature selection methods often focus on identifying a single Markov Boundary when real-world data could have many feature subsets that are equally good boundaries. In our work, we design a new multiple-Markov-boundary-based predictive model, Galaxy, to identify the precursors to heavy precipitation event clusters and predict heavy rainfall with a long lead time. Our model identified the cold surges along the coast of Asia as an essential precursor to the surface weather over the United States, a finding which climate experts later corroborated.

Second, chaotic behavior exists in many nonlinear Spatio-temporal systems. A reliable solution for these systems must handle their complex space-time dynamics and sensitive dependence on initial and boundary conditions. We propose a new recurrent architecture(error trajectory tracking) and an accompanying training regime(Horizon Forcing) for prediction in chaotic systems. By better evaluating the consequences of local errors when modeling nonlinear systems, the proposed model can push the time horizon of reliable predictions further into the future.

# ACKNOWLEDGMENTS

This journey would not have been possible without the dedicated support and encouragement from friends, family, and colleagues.

I would like to first deliver my sincere gratitude to my adviser, Professor Wei Ding. She first saw my potential to make this journey, guiding me along the way and constantly pushing me to enhance my quality for discovering and solving research problems. She has always inspired me to believe that, even in difficult times, all things are possible. I would also express my appreciation to Professor Ping Chen for his guidance during the long process from idea to experiment to paper. My great thanks to Professor and Dissertation Committee member Shafiqul Islam of Tufts University for providing the real-world data and expertise on which all our theories and models depend. I must also thank Professor and Committee member Nurit Haspel for her clear and precise explanations in the course Analysis of Algorithms. Her suggestions and comments influenced me a lot. Great thanks to Professor and Committee member Dan Simovici for his expertise and guidance and all the other Computer Science Professors who shared their knowledge and insights during my coursework.

I want to express my great thankfulness to the rest of the Artificial Intelligence and Knowledge Discovery Labs members. Especially to Dr. Matthew Almeida. The experience of cooperating on Horizon Forcing papers and countless discussions in research and personal aspects is one of my invaluable assets. Also, great thanks for helping me with the proofreading. To Dr. Tong Wang, Dr. Kaixun Hua, The positive learning experience from our machine learning study group has become one of my precious memory. I also want to thank Dr. Dawei Wang, Dr. Yang Mu, Dr.

Joseph Paul Cohen, Dr. Henry Lo, and Dr. Yahui Di for sharing their research experience. Many thanks to Dr. Shaohua Jia, Dr. Jipeng Qiang, Dr. Tingting Lu, Hefei Qiu, Olga Andreeva, Tianyu Kang, Zihan Li, Chengjie Zheng, Dr. Yi Ren, Dr. Siyuan Gong, Patrick Flynn, and many others. I appreciate their support and friendship during my Ph.D. study.

My special thanks go to my parents (Shengcheng Zhuang, Cailian Zhang) and my wife's family for their long-standing support, patience, comprehension, and encouragement throughout the entire journey of my Ph.D. study.

Finally, I express my most significant appreciation to my wife, Rui Lu, for her persistent understanding and consideration. I thank her for accompanying me on this journey through both the good and the tough times.

# TABLE OF CONTENTS

CHAPTER Page

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Research Questions

Space and time are omnipresent in measurements in many fields, including climate science, social science, epidemiology, transportation, criminology, and earth science. Many data collection methods are designed to record each measurement's spatial and temporal information in the data because of the Spatio-temporal nature of real-world processes studied in these fields. Massive Spatio-temporal data provides more possibilities for research in these fields. Forecasting the long-term future is one of the most urgent needs in Spatio-temporal data analysis. Many practical applications, such as disaster prevention, dynamic traffic management, and resource pre-allocation, require longer to prepare and provide emergency support. High-dimensional Spatio-temporal phenomena are more challenging than traditional prediction problems because they deal with nonlinear time correlations and dynamic and complex spatial correlations. The challenge becomes extremely arduous in the long-term prediction since tiny errors can traverse complex correlations and lead to the butterfly effect of error propagation which makes the predictive ability at each upcoming space-time position rapidly lost. At present, achieving effective long-term prediction of Spatio-temporal phenomena is still challenging in data mining and machine learning.

This thesis focuses on two considerations that affect the performance of long-term prediction on Spatio-temporal data sets. The first is high predictability and interpretability feature subsets from the original high dimensional Spatio-temporal feature space. Long term predictions always need to consider a deluge of features over space

and time. However, it is almost impossible that all these features are necessary for model building. Redundant and irrelevant features may reduce the model's generalization ability, Interpretability, and accuracy and increase its complexity. Therefore, searching for the most meaningful sub-feature-set becomes very important. In causal analysis, the optimal sub-feature-set that contains all the target feature information is called the Markov Boundary. However, the existing feature selection methods often focus on identifying a single Markov Boundary when real-world data could have many feature subsets that are equally good boundaries. A single Markov Boundary may not cover all the information of the target feature. We expect to identify multiple Markov boundaries containing all the target feature's predictability and interpretability information for constructing forecasting model to simplify models for more straightforward interpretation and enhanced generalization by reducing overfitting. The detail is presented in Chapter 3.

The second is handling the complex space-time dynamics and sensitive dependence on initial conditions in Spatio-temporal data. Spatio-temporal correlations are highly complex, chaotic, dynamic, and sensitive dependent on initial conditions. Especially in the long-term prediction, tiny errors in the initial conditions will soon lead to exponential differently, which means the predictive ability is rapidly lost. We expect to push the time horizon at which reliable predictions can be made further into the future by modeling the evolution of the initial errors.. This work is presented in Chapter 4.

## 1.2 Contributions of this thesis

This thesis focuses on two considerations for long-term forecasting on big Spatio-temporal data, specifically a technique for identifying the most relevant and meaningful features from the original Spatio-temporal feature space and a method for modeling complex space-time dynamics with sensitive dependence on initial and boundary

conditions. I provide a summary of the contributions here, with a more thorough description is given in the Conclusions chapter. In Chapter 3:

- We define the notion of Partial Minimal Target Explanation to represent the most relevant and meaningful features to the target instances derived from the same subpopulation. We also design an algorithm to detect PMTE using equivalent feature detection.

- We define the notion of Galaxy space to represent the most meaningful feature sets with global faithfulness of the overall population of the target feature. We also design and implement the Galaxy algorithm to discover the Galaxy space from the mixed distribution and then do forecasts by its ensemble predictive power. In our empirical experiments, our algorithm outperforms state-of-the-art ensemble methods under dimensional feature space.

- We apply Galaxy to study historical precipitation data in the Des Moines river basin. Our empirical study includes 5,313,600 features over 67 years of data. Our model identified the cold surges along the coast of Asia as an essential precursor to the surface weather over the United States, a finding which climate experts later corroborated.

In Chapter 4:

- We design a new recurrent architecture for error trajectory tracing (ETT) to simulate temporal dynamics. It allows the model to optimize its parameters based on the true, longer-term consequences of minor initial prediction errors and how the trajectories beginning at the predicted states evolve forward to the time horizon.

- We present Horizon Forcing, a new training regime for optimizing our ETT models. It begins with a short-term error evaluation and then focuses on the

long term as training progresses. By doing this with shared weights, we further improve the single-step error (a proxy for the system's transition function) because minimizing long-term error necessitates controlling short-term error in chaotic systems.

- We extensively validate our method on three well-studied chaotic systems with known dynamics and a set of real-world time series prediction tasks.

In summary, this thesis presents two major concerns for long-term prediction problems on Spatio-temporal data and offers a novel approach to use in each case. Methods are validated with experiments on real-world data sets, including one meteorological dataset, three classics, chaotic systems, and four real-world time series prediction tasks with chaotic characteristics.

# CHAPTER 2

# BACKGROUND

## 2.1 Notation

- an italic lowercase letter to denote an instance (e.g. $x$)
- an italic capital Greek letter to denote a feature (e.g. $X$)
- a boldface, capital Greek letter to denote a feature set (e.g. $\mathbf{X} = (X_1, ..., X_k) \in \mathbb{R}^k$)
- a backslash to denote difference between feature sets (e.g. $\mathbf{X} \backslash \{X_i\} = \{X_1, ..., X_{i-1}, X_{i+1}, ..., X_k\}$).
- $\mathbb{D}$ to denote a data set.
- $Y$ to denote the target feature.
- $P$ to denote a probability distribution.
- $G$ to denote a graph of the Bayesian network

## 2.2 Long-Term Prediction Problem

**Definition 2.2.1.** ***Long-term Prediction:*** *A long-term prediction problem can be specified as a function that learns to map inputs, given historical observations over space and time, to corresponding outputs for multiple future time steps.*

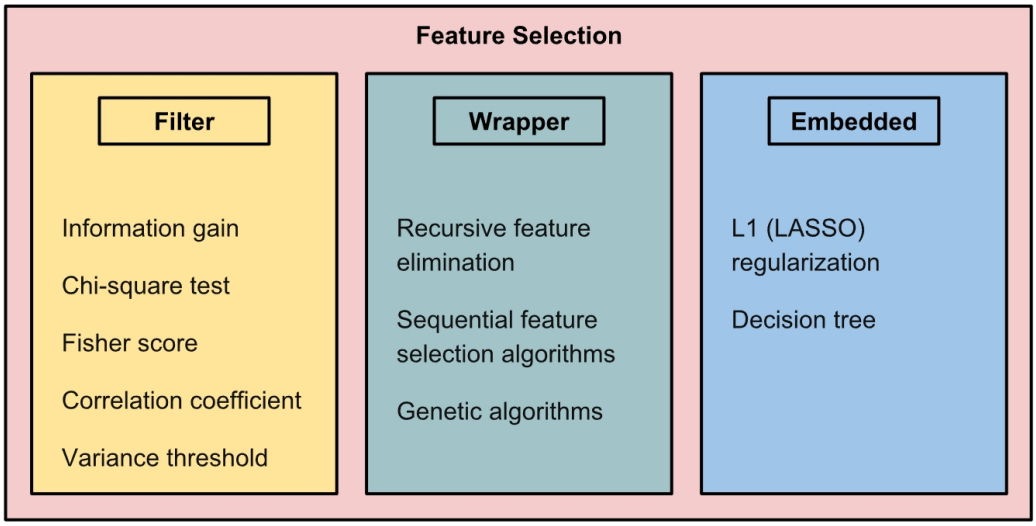



Figure 2.1: Example feature selection methods

## 2.3 Feature Selection Algorithms

Features are measurable attributes or characteristics of a phenomenon, usually appearing as columns in a data table. Large-scale data sets always contain a large number of features. However, it is almost rare that all the features in the data set are necessary for model building. Feature selection has always been a superior technique to interweave the feature generation process with the feature testing process to avoid generating features that are unlikely to be helpful. It continuously updates the subset of predicted features through correlation analysis of newly emerging features, target features, and selected features to remove irrelevant and redundant features and select the most relevant features. There are three general classes of feature selection algorithms are introduced as follows, some example feature selection methods can be found in Figure2.1:

- **Wrapper methods:** Wrapping methods select features by iterating and trying a different subset of features until the optimal subset is reached.

  **Advantages:**

  - Simple.

  - Interacts with the predictive model.

  **Disadvantages:**

  - Large calculation time for high-dimensional data.
  - Increasing overfitting risk when there are not many data points.
  - Model dependent selection.

- **Embedded methods:** Embedded methods embed feature selection as part of the model creation process. This usually results in the selection process being completed together with the model adjustment process.

  **Advantages:**

  - Interacts with the predictive model.
  - Lower computational complexity than wrapper methods.
  - Models feature dependencies.

  **Disadvantages:**

  - Model dependent selection.

- **Filter methods:** Filter methods select features based on correlations with the target feature. These methods are particularly effective in computation time and robust to overfitting.

  **Advantages:**

  - Fast and Scalable.
  - Independent of the predictive model.
  - Robust to overfitting.

  **Disadvantages:**

  - Ignores interaction with the predictive model.

## 2.4 Causal Inference

Causal inference is the field of artificial intelligence, which aims to reveal the causal relationship between features from observational data[61]. Combining causal discovery in feature selection can well identify meaningful feature subsets. In this section, we introduce causal reasoning and Markov boundary theory.

### 2.4.1 Basic Properties of Probability Distributions

The following theorem can be used for the theoretical analysis of probability distribution and the proof of the correctness of the Markov boundary algorithm. . It was proposed by [63] and Its proof is given in the book[54].

**Theorem 1.** *Let* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, *and* $\mathbf{D}$ *be any four subsets of features from* $\mathbf{X}$. *The following five properties hold in any joint probability distribution* $P$ *over features* $\mathbf{X}$.

1. *Symmetry:* $\mathbf{A} \perp \mathbf{B} \mid \mathbf{C} \Leftrightarrow \mathbf{B} \perp \mathbf{A} \mid \mathbf{C}$
2. *Decomposition:* $\mathbf{A} \perp (\mathbf{B} \cup \mathbf{D}) \mid \mathbf{C} \Rightarrow \mathbf{A} \perp \mathbf{B} \mid \mathbf{C}$ *and* $\mathbf{A} \perp \mathbf{D} \mid \mathbf{C}$
3. *Weak union:* $\mathbf{A} \perp (\mathbf{B} \cup \mathbf{D}) \mid \mathbf{C} \Rightarrow \mathbf{A} \perp \mathbf{B} \mid (\mathbf{C} \cup \mathbf{D})$
4. *Contraction:* $\mathbf{A} \perp \mathbf{B} \mid \mathbf{C}$ *and* $\mathbf{A} \perp \mathbf{D} \mid (\mathbf{C} \cup \mathbf{B}) \Rightarrow \mathbf{A} \perp (\mathbf{B} \cup \mathbf{D}) \mid \mathbf{C}$
5. *Self-conditioning:* $\mathbf{A} \perp \mathbf{C} \mid \mathbf{C}$

*If* $P$ *is faithful to* $G$, *then* $P$ *satisfies the above five properties and following two properties:*

1. *Intersection:* $\mathbf{A} \perp \mathbf{B} \mid (\mathbf{C} \cup \mathbf{D})$ *and* $\mathbf{A} \perp \mathbf{D} \mid (\mathbf{C} \cup \mathbf{B}) \Rightarrow \mathbf{A} \perp (\mathbf{B} \cup \mathbf{D}) \mid \mathbf{C}$
2. *Composition:* $\mathbf{A} \perp \mathbf{B} \mid \mathbf{C}$ *and* $\mathbf{A} \perp \mathbf{D} \mid \mathbf{C} \Rightarrow \mathbf{A} \perp (\mathbf{B} \cup \mathbf{D}) \mid \mathbf{C}$

### 2.4.2 Markov Boundary Theory

In this subsection we begin by revisiting the concepts of Markov blanket and Markov boundary and theoretically characterize distributions with multiple Markov boundaries of the same target feature.

**Definition 2.4.1.** ***Markov Blanket:*** *A Markov blanket* $\mathbf{M}$ *of the target feature* $Y \in \mathbf{X}$ *in the joint probability distribution* $P$ *over feature set* $\mathbf{X}$ *is a set of features conditioned on which all other features are independent of* $Y$*, that is,*

$$\forall A \in (\mathbf{X} \setminus \mathbf{M} \setminus \{Y\}), Y \perp A \mid \mathbf{M} \tag{2.1}$$

According to definition of Markov Blanket, the set of all features $\mathbf{X}$ excluding $Y$ is one of $Y$'s Markov blankets. But it may have redundant or unrelated variables. Therefore, the smallest Markov blanket may be more critical.

**Definition 2.4.2.** ***Markov Boundary:*** *A sub feature set* $\mathbf{M}$ *is called a Markov boundary of* $Y$ *iff it is the smallest Markov blanket of* $Y$*.*

**Theorem 2.** *If a joint probability distribution* $P$ *over features* $X$ *is faithful to* $G$*, that is, it satisfies the seven properties in theorem1, then for each* $Y \in \mathbf{X}$ *, there exists a unique Markov boundary of* $Y$*.*

According to theorem 2, if a joint probability distribution $P$ that is faithful to $G$, there is a unique Markov boundary in this distribution. However, the faithfulness condition is hard to hold due to the common response issue and confounding issue caused by lurking feature, which makes the Markov boundary of the target feature not unique in common situations.

- **Common response** occurs when changes in both feature $A$ and feature $B$ are caused by a lurking feature $H$. As shown in Figure2.2 (b), the effect of $H$ produces the association between $A$ and $B$.

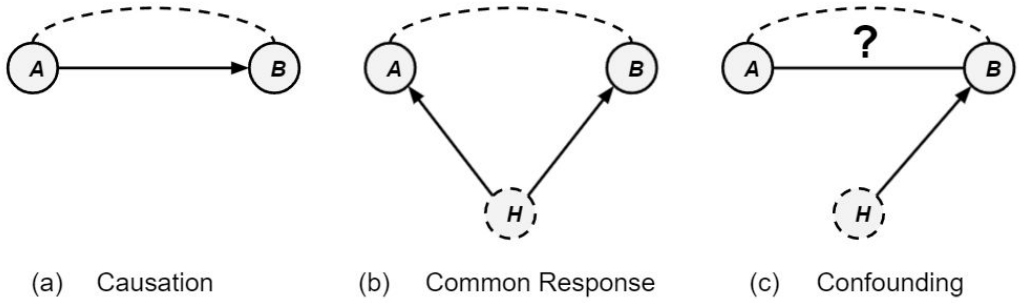



Figure 2.2: Some explanations of an observed association. The dashed lines show an association. The solid arrows show a cause-and-effect link. a: Causation: Changes in $A$ cause change in $B$; b: Common response: Changes in both $A$ and $B$ are caused by a lurking feature $H$; c: Confounding: The effect of $A$ on $B$ is confounded by a lurking feature $H$.

- **Confounding** occurs when the a explanatory feature $A$ and a lurking feature $H$, have collective causation effect on feature $B$, but the difference between the contribution of these two features cannot be distinguished, Figure2.2 (c).

**Definition 2.4.3.** ***Equivalent information:*** *Two subsets of features* $\mathbf{B} \subseteq \mathbf{X}$ *and* $\mathbf{C} \subseteq \mathbf{X}$*, and* $\mathbf{B} \neq \mathbf{C}$*, They contain equivalent information about target feature* $Y$ *iff the following conditions hold:*

$$Y \not\perp \mathbf{B}, Y \not\perp \mathbf{C}, Y \perp \mathbf{B} \mid \mathbf{C}, \text{ and } Y \perp \mathbf{C} \mid \mathbf{B} \tag{2.2}$$

**Lemma 1.** *If* $\mathbf{M}$ *is a Markov boundary of target feature* $Y$*,* $\mathbf{B} \subset \mathbf{M}$*, and there is a subset of features* $\mathbf{C}$*,* $\mathbf{B} \neq \mathbf{C}$*, such that* $\mathbf{B}$ *and* $\mathbf{C}$ *contain equivalent information about* $Y$*, then* $(\mathbf{M} \setminus \mathbf{B}) \cup \mathbf{C}$ *is also a Markov boundary of* $Y$*.*

Consider a Bayesian network shown in Figure 2.3. If feature $A$ and $B$ have hidden common response feature $H$, $A$ and $B$ may have equivalent information of target feature $Y$.

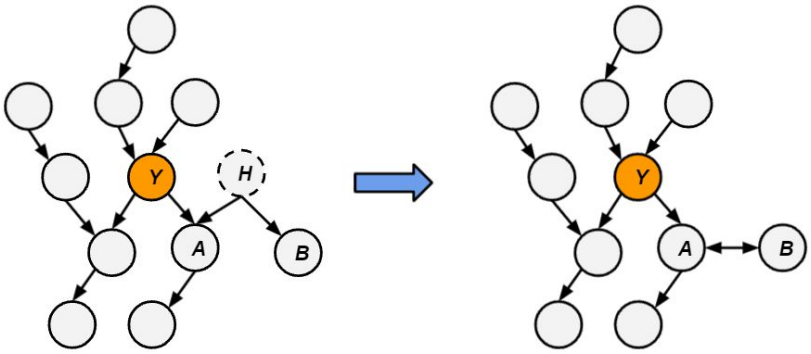



Figure 2.3: An example of equivalent information caused by common response, where $H$ is a lurking feature, and two features connected by a bi-directed path have equivalent information of target feature.

## 2.5 Learning Temporal Dynamics via Deep Neural Networks

Due to the flexible structure, deep learning models are increasingly used in learning temporal dynamics. Precisely, Recurrent Neural Network(RNN)[23], one of the deep learning models, models the temporal dynamics of time-lapse sequences by recurrent neural connections. However, standard RNN has an issue where the model's ability to remember important information about a previous time step decays as the model looks at more and more inputs. Long Short-Term Memory (LSTM)[30], a variation on the traditional RNN, addresses this by introducing the input and forget "gates," neurons with weights that allow the network the learn how much it should remember or forget about the history, relative to its input at the current step. Gated Recurrent Unit (GRU)[16] is a further modification of the LSTM architecture, which combines the forget and input gates of an LSTM network into a single update gate. Often, this leads to a simpler recurrent network with performance very close to that of the more complex LSTM architecture.

# CHAPTER 3

# IDENTIFICATION OF MULTIPLE INTERPRETABLE PREDICTOR SETS FOR ENSEMBLE LEARNING

## 3.1 Introduction

With a rapid increase in the availability of spatio-temporal climate data and growing popularity of data mining techniques [72], qualitative understanding between the meteorological variables and the climate phenomena has become a major objective of current meteorology. This makes interpretability has a great need in the design of predictive models.

However, the climate data is always high-dimensional and spatio-temporal correlated, and so the relationships among the meteorological variables are very complicated - especially in spatial-temporal studies of numerous variables simultaneously [38, 31]. With time and space increasing, the number of elements potentially contributing to a meteorological event grows sharply. This makes identifying the causes of climate phenomena from large spatial-temporal scale meteorological features extremely difficult. For example, explaining phenomena five days ahead is typically less reliable than explaining phenomena for the next day. This occurs since small changes may likely influence observed weather events as time advances (butterfly effect), but such small changes can be easily overlooked due to the high degree of spatial and temporal correlations among the features as their magnitude decreases, so it is difficult to analyze how a multitude of tiny events will impact observed weather as time moves forward.

On the other hand, climate phenomena are the result of the interactions and operations of atmospheric physical effects on multiple spatial-temporal scales. This

means the climate events occurred in different regions or different times may have different explanatory patterns. For example, compared with precipitation during the cold season, warm season precipitation generally occurs on smaller spatial-temporal scales with large gradients in precipitation amounts. We cannot use the same meteorological features' influence to explain all precipitation events. Thus identifying explanatory patterns in all perspective are of significant interest for understanding the causes of climate phenomena.

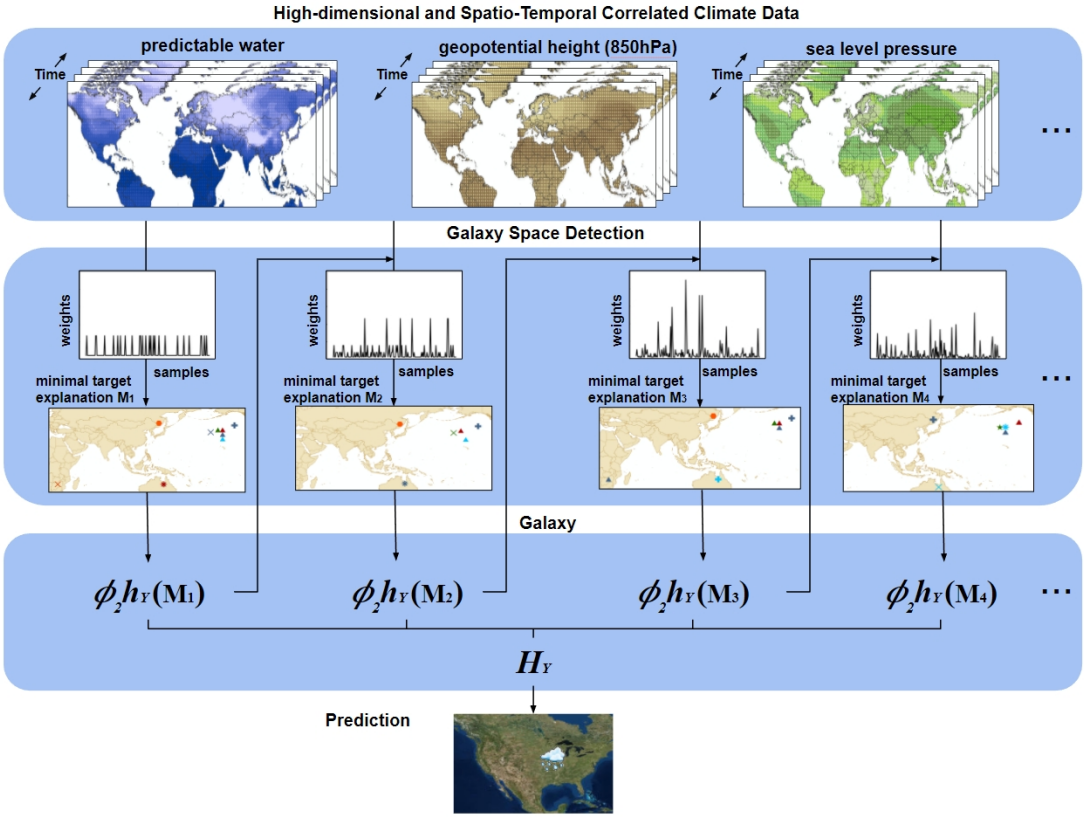



Figure 3.1: An example of Galaxy for precipitation forecasting. Galaxy detects the Galaxy space, the explanations for overall target population, by sequentially detecting minimal target explanation of every individual subpopulation within the overall population, and then forecasting target by their ensemble predictive power.

In this paper, we propose a new interpretable predictive model Galaxy (Figure3.1) which can detect explanatory patterns in all perspective of target climate phenomena from large spatial-temporal scale meteorological features and forecast by their ensem-

ble predictive power. Our work is not only able to give interpretable explanations on high-dimensional spatio-temporal climate data, but also provide the preconditions for scientists for further studies. Thus, our main contributions are as follows:

- **Minimal Target Explanation:** We define the notion of minimal target explanation to represent the explanation that locally faithful to the target instances derived from the same subpopulation.
- **Galaxy Space:** We define the notion of Galaxy space to represent the explanation with global faithfulness of the overall population of the target feature.
- **Galaxy:** We design and implement the Galaxy algorithm, to discover the Galaxy space from mixture distributed feature data, and then forecast by its ensemble predictive power. In our empirical experiments, our algorithm outperforms state-of-the-art ensemble methods under dimensional feature space.
- **Interpretable on Real-world scenario:** We apply Galaxy to study historical precipitation data in the Des Moines river basin. Our empirical study includes 5,313,600 features over 67 years of data. We are able to understand precipitation forecasting in the area.

The rest of this paper is organized as follows. Section 3.2 reviews related work. Section 3.3 presents the Galaxy space, Galaxy detection algorithms, and as theoretical analysis of Galaxy. Section 3.4 discusses our empirical studies on synthetic data and a real-world precipitation data set, and showcase the explanatory patterns. We conclude the paper in section 3.5.

## 3.2 Related Work

To provide high-quality explanations for observed weather events, we need to look for the features which are most likely to influence them. However, the high-dimensionality and high degree of spatio-temporal correlations are serious challenges

of climate data. Although the existing dimension reduction methods are able to address the high dimensionality by reporting a subset of features which are strongly contribute on prediction, detecting the features most influent on observed weather events is still facing the following challenges.

- Scale amplification: Weather systems are very sensitive to changes in initial conditions. So many small perturbations in air motion could compound to result in large changes over longer time frames.

- Error magnification and analysis: Because the system is so sensitive, measurement error in monitoring devices can lead to errors in analysis.

Markov boundary based feature selection is the state-of-the-art dimension reduction technology using causal inference[77, 75, 10, 74], and a Markov boundary of the target feature can be tread as the knowledge needed to predict the behavior of the target. However, a unique Markov boundary ideally exists for targets in datasets under a strong faithfulness assumption[54, 3, 50], which is often violated in real-world data because of the occurrence of hidden variables, hypothesis test errors and some fake relevance of pure chances, so multiple Markov boundaries exist almost in all situations[63]. Which one can best explain observed weather events remains an open research problem.

On the other hand, the climate phenomena are the result of the interactions and operations of atmospheric physical effects on multiple spatial-temporal scales, the observed climate events are likely derived from mixture populations, which the climate events occurred in different regions or different times may derived from different subpopulations[60]. This makes one pattern may not be interpretable to all observed climate events. We are interested in how to detect the explanations for climate events of the mixture populations.

In this paper, we discuss a new predictive model with high interpretability to facilitate climate scientists to better understand causes of climate events.

## 3.3 Galaxy Space

### 3.3.1 Galaxy Space

The Galaxy space is designed for global faithfulness and efficient interpretability. Formally, suppose we have a dataset $\mathbb{D}$ that includes $n$ instances. Each of the instance is in the form of $(\mathbf{X}, Y)$, where $\mathbf{X} = (X_1, ..., X_k) \in \mathbb{R}^k$ and $Y \in \mathbb{R}$, and the $n$ instances of $Y$ are derived from $m$ subpopulations. Then the overall conditional distribution of $Y$ can be represented as the mixture of subpopulation conditional distributions:

$$P(Y \mid \mathbf{X}) = \sum_{i=1}^{m} \phi_i P_i(Y \mid \mathbf{X}), \tag{3.1}$$

where $\phi$ is the mixture component weight, $P_i(Y \mid \mathbf{X})$ presents the conditional probability distribution of Y of the $i^{th}$ subpopulation, and $P(Y \mid \mathbf{X})$ is the overall conditional population distribution of $Y$.

**Definition 3.3.1.** ***Target Explanation (TE):*** *A feature set* $\mathbf{M} \subseteq \mathbf{X}$ *is said to be a target explanation of* $P(Y \mid \mathbf{X})$ *if and only if:*

$$P(Y \mid \mathbf{X}) = P(Y \mid \mathbf{M}) \tag{3.2}$$

By Definition 3.3.1, a non-minimal target explanation can be trivially produced by adding redundant or irrelevant features into itself. Only minimal target explanations are of interest in this paper. If a feature subset $\mathbf{M} \subseteq \mathbf{X}$ is a minimal target explanation, it is more efficiently than $\mathbf{X}$ to interpret the instances derived from $P(Y \mid \mathbf{X})$.

**Definition 3.3.2.** ***Minimal Target Explanation (MTE):*** *A target explanation* $\mathbf{M}$ *is said to be a minimal target explanation if and only if no proper subset of* $\mathbf{M}$ *satisfies the definition of target explanation.*

Since a **MTE** of $P(Y \mid \mathbf{X})$ is the minimal explanation of $P(Y \mid \mathbf{X})$, it does not have any redundant or irrelevant feature. Then back to the mixture conditional distributions representation, we define partial target explanation as follows.

**Definition 3.3.3.** ***Partial Target Explanation (PTE):*** *If* $P(Y \mid \mathbf{X})$ *can be represented as mixture of subpopulation conditional distributions* $\sum_{i=1}^{m} \phi_i P_i(Y \mid \mathbf{X})$*, then we say a target explanation of* $P_i(Y \mid \mathbf{X})$ *is a partial target explanation of* $P(Y \mid \mathbf{X})$*.*

We also define a **MTE** of $P_i(Y \mid \mathbf{X})$ as a partial minimal target explanation **(PMTE)** of $P(Y \mid \mathbf{X})$. Then, we say **PMTE** is **locally faithful**, i.e. it is efficiently interpretable to the instances derived from a subpopulation of $Y$. Thus, for the overall population of $Y$, we define the Galaxy space of $Y$ as follows.

**Definition 3.3.4.** ***Galaxy Space*** $\mathbb{G}$***:*** *If* $P(Y \mid \mathbf{X})$ *can be represented as mixture of subpopulation conditional distributions* $\sum_{i=1}^{m} \phi_i P_i(Y \mid \mathbf{X})$*, then we say* $\prod_i^m \boldsymbol{M}_i$ *is a Galaxy space* $\mathbb{G}$ *of* $Y$ *if and only if every* $\boldsymbol{M}_i$ *corresponds a* ***MTE*** *of* $P_i(Y \mid \mathbf{X})$*.*

Based on the Definition 3.3.4, a Galaxy space $\mathbb{G}$ of $Y$ is a set of **PMTE**s of $P(Y \mid \mathbf{X})$. Each **PMTE** provides a local partical minimal target explanation, and the complete set of **PMTE** is able to interpret the entire instances of $Y$, i.e. **globally faithful**. In order to look for a Galaxy space, we need to first detect **PMTE**.

### 3.3.2 Partial Minimal Target Explanation (PMTE) Detection

Detecting a **PMTE** of $P(Y \mid \mathbf{X})$ is to look for the "smallest" explanation that is locally faithful to a subpopulation of $Y$. Here we can further decompose it into two

problems: (1) Detecting the "smallest" explanation **MTE** of a subpopulation of $Y$ and (2) Identifying the instances from $\mathbb{D}$ which belong to this subpopulation.

To address the first problem, we utilize the approaches in causal inference. In the domain of causal discovery, a Bayesian network[54] is a standard tool for modeling the conditional dependencies of the features, and a Markov boundary of a target feature corresponds to a local causal neighborhood of it and consists of all its direct causes, effects, and causes of the direct effects. This means that knowledge of the values of the Markov boundary features should render all other features superfluous for predicting $Y$[4]. In faithful joint distributions of $(\mathbf{X}, Y)$, there exists a unique Markov boundary of $Y$[65]. However, in real-world data, the faithfulness condition may be violated by hidden variables, hypothesis test errors and some fake relevance of pure chances. This makes the Markov boundaries of the target variable not unique[63]. In order to define a unique minimal target explanation, we first state the definition of optimal predictor and link it with the concept of target explanation, then we detect the minimal target explanation using optimal predictor on the Markov boundaries of the target feature.

**Definition 3.3.5.** ***Optimal Predictor****[63]: Given a data set $\mathbb{D}$, a learning algorithm $h_Y$, and a performance metric $T$ to assess the learner's model, a feature subset $\mathbf{M} \subseteq \mathbf{X}$ is an optimal predictor of $Y$ if it maximizes the performance metric $T$ for predicting $Y$ using learner $h_Y$ in the data set $\mathbb{D}$.*

The following theorem states the link between the optimal predictor and the target explanation.

**Theorem 3.** *If a conditional probability distribution $P(Y \mid \mathbf{X})$ can be estimated accurately by maximizing a performance metric $T$ on a learning algorithm $h_Y$, then $\mathbf{M} \subseteq \mathbf{X}$ is a target explanation of $P(Y \mid \mathbf{X})$ if and only if it is an optimal predictor of $P(Y \mid \mathbf{X})$.*

*Proof of Theorem 3*

- Prove a **TE** of $P(Y \mid \mathbf{X})$ is an optimal predictor of $P(Y \mid \mathbf{X})$: If $\mathbf{M} \subseteq \mathbf{X}$ is a target explanation of $P(Y \mid \mathbf{X})$, then $P(Y \mid \mathbf{X}) = P(Y \mid \mathbf{M})$ and this implies that $T$ will be maximized on learning algorithm $h_Y$, therefore, $\mathbf{M}$ is an optimal predictor of $P(Y \mid \mathbf{X})$.

- Prove an optimal predictor of $P(Y \mid \mathbf{X})$ is a **TE** of $P(Y \mid \mathbf{X})$: Suppose $\mathbf{M} \subseteq \mathbf{X}$ is an optimal predictor of $P(Y \mid \mathbf{X})$ but it is not a target explanation, so $P(Y \mid \mathbf{X}) \neq P(Y \mid \mathbf{M})$, and this implies $T_{h_Y(\mathbf{M})} > T_{h_Y(\mathbf{X})}$. By Definition 3.3.1, $\mathbf{X}$ is always a target explanation, thus it is also an optimal predictor of $P(Y \mid \mathbf{X})$. Therefore, the following should hold: $T_{h_Y(\mathbf{M})} = T_{h_Y(\mathbf{X})}$. This is contradiction. Therefore, $\mathbf{M}$ is a target explanation.

By the Definition 3.3.1 and Theorem 3, we get Corollary 3.7 to address the second problem of Identifying the instances from $\mathbb{D}$ which belong to this subpopulation.

**Corollary 1.** *If conditional probability distribution can be estimated accurately by maximizing a performance metric $T$ on a learning algorithm $h_Y$, then the instances predicted correctly by $h_Y$ are derived from the same distribution.*

Now we can use Theorem 3 as the criterion for detecting the **MTE** of $P_i(Y \mid \mathbf{X})$.

$$\begin{aligned} \mathbf{M}_i \quad &= \underset{\mathbf{M}' \in f_Y(\mathbf{X})}{\operatorname{argmax}} \; T_{h_Y(\mathbf{M}')} \\ \text{s.t.} \quad & P_i(Y \mid \mathbf{X}) = h_Y(\mathbf{M}_i) \end{aligned} \tag{3.3}$$

Here $f_Y : \mathbb{R}^k \to \mathbb{R}^d$ is a Markov boundaries detection algorithm, $d \leq k$, $h_Y : \mathbb{R}^d \to \mathbb{R}$ is a learning algorithm for predicting $Y$, and $T$ is a performance metric to assess $h_Y$ (the bigger the value, the better the performance). For $f_Y$, we utilize the idea mentioned in [63], which firstly detects a Markov boundary $\mathbf{M}$ from $\mathbf{X}$, and then tries to replace part of $\mathbf{M}$ by its equivalent feature sets explored from the residual features. Finally a **PMTE** of $P(Y \mid \mathbf{M})$, which is the **MTE** of $P_i(Y \mid \mathbf{M})$, can be detected by choosing the optimal Markov boundary $\mathbf{M}_i$ which maximizes the

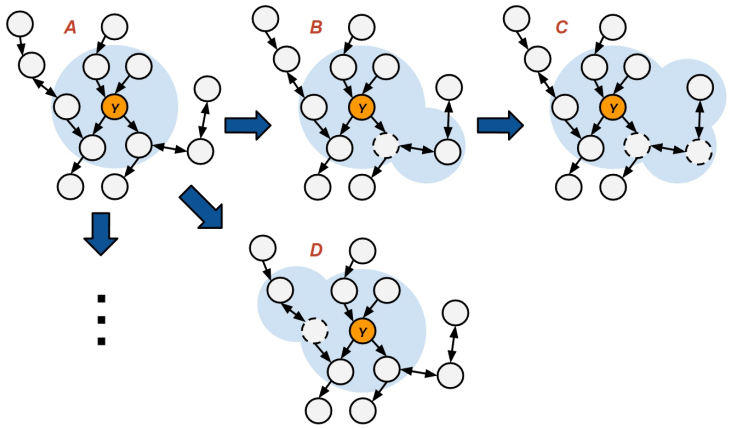



Figure 3.2: Multiple Markov boundaries detection via equivalent information exploration. A, B, C, D are Markov boundaries of $Y$ in a Bayesian network. A is the Markov boundary detected from the original feature space. B, C, D are Markov boundaries generated by replacing part of features in A by equivalent features explored from the residuals.

performance metric $T$ for predicting $Y$ using learner $h_Y$. The detail is illustrated in Fig 3.2 and Algorithm 1.

### 3.3.3 Galaxy Space Detection

The probability distribution of $Y$ in the overall population is represented as a mixture distribution, then detecting the Galaxy space of $Y$ is actually looking for the explanations which globally faithful to the mixture distribution of $Y$. It can be implemented by detecting every subpopulation's **MTE** in the mixture distribution of $Y$ via Theorem 3. Since every **PMTE** in the Galaxy space is locally faithful to a subpopulation of $Y$, the overall explanation of $Y$ can be represented as the ensemble of the Galaxy space. Thus, we give the definition of Galaxy predictor and then link it with the concept of Galaxy space.

**Definition 3.3.6.** ***Galaxy Predictor****: Given a data set $\mathbb{D}$, we say a family of feature subsets $\prod_i^m \mathbf{M}_i$, where $\mathbf{M}_i \subseteq \mathbf{X}$, is a Galaxy predictor of $Y$ if it maximizes the performance metric $T$ for predicting $Y$ using an ensemble learning algorithm $H_Y$.*

**Algorithm 1:** Partial Minimal Target Explanation (**PMTE**) Detection.

**Input:**

- data set $\mathbb{D}$ for features $\mathbf{X}$; target feature $Y$; Markov boundary detection algorithm $f_Y$; learning algorithm $h_Y$; performance metric T;

**Output:**

- $\mathbf{M}$, a partial minimal target explanation of $Y$.
- $h_Y(\mathbf{M})$, a trained learning algorithm on $\mathbf{M}$.

**begin**

$\mathbf{M}'_{init} = \text{empty}$ `/* Initialize new Markov boundary with an empty set */`

$\mathbf{M}'$, $\mathbf{R} = f_Y(\mathbf{M}'_{init}, \mathbf{X})$ `/* Detect` $1_{st}$ `Markov boundary M' and residual features` $\mathbf{R}$ `from` $\mathbf{X}$ `on` $\mathbb{D}$ `*/`

$\mathbf{M} = \mathbf{M}'$

$Performance = T_{h_Y(\mathbf{M}')}$

**for** $\forall \mathbf{S} \subset \mathbf{M}'$ **do**

  $\mathbf{R}_{new} = \mathbf{R}$

  $\mathbf{M}'_{init} = \mathbf{M}' \backslash \mathbf{S}$ `/* Initialize new Markov boundary as` $\mathbf{M}' \backslash \mathbf{S}$ `*/`

  **repeat**

    $\mathbf{M}'_{new}$, $\mathbf{R}_{new} = f_Y(\mathbf{M}'_{init}, \mathbf{R}_{new})$ `/* Replacing` $\mathbf{S}$ `by exploring its equivalent features from` $\mathbf{R}_{new}$ `*/`

    **if** $T_{h_Y(\mathbf{M}'_{new})} > Performance$ **then**

      $\mathbf{M} = \mathbf{M}'_{new}$

      $Performance = T_{h_Y(\mathbf{M}'_{new})}$

  **until** $\mathbf{R}_{new}$ *is empty*

Return $\mathbf{M}$, $h_Y(\mathbf{M})$

We call the ensemble learning algorithm $H_Y$ on Galaxy predictor as Galaxy. The following theorem provides the link between the Galaxy predictor and the Galaxy space.

**Theorem 4.** *A family of feature subsets $\prod_i^m \mathbf{M}_i$, where $\mathbf{M}_i \subseteq \mathbf{X}$, is a Galaxy space of $Y$ if and only if it is an Galaxy predictor of $Y$.*

*Proof of Theorem 4*

- Prove a Galaxy space of $Y$ is a Galaxy predictor of $Y$:

  If a family of feature subsets $\prod_i^m \mathbf{M}_i$, where $\mathbf{M}_i \subseteq \mathbf{X}$, is a Galaxy space of $Y$, then by Definition 3.3.4, every $\mathbf{M}_i$ corresponds a **PMTE** of $P_i(Y \mid \mathbf{X})$ in $\sum_{i=1}^m \phi_i P_i(Y \mid \mathbf{X})$. This implies that $\mathbf{M}_i$ is an optimal predictor of $P_i(Y \mid \mathbf{X})$ by maximizing the performance metric $T$ on a learning algorithm $h_Y$. Therefore, $\prod_i^m \mathbf{M}_i$ is a Galaxy predictor and Galaxy is presented as $H_Y = \sum_{i=1}^m \phi_i h_Y(\mathbf{M_i})$.

- Prove a Galaxy predictor of $Y$ is a Galaxy space of $Y$:

  Suppose $\prod_i^m \mathbf{M}_i$ is a Galaxy predictor of $Y$, but it is not a Galaxy space of $Y$ and Galaxy is presented as $H_Y = \sum_{i=1}^m \phi_i h_Y(\mathbf{M_i})$, so there is at least one $\mathbf{M}_j \in \prod_i^m \mathbf{M}_i$ is not a **PMTE**. This implies that $\mathbf{M}_j$ is not an optimal predictor. so there exist an optimal predictor to make the performance of Galaxy better. This is contradict to that $\prod_i^m \mathbf{M}_i$ is a Galaxy predictor. Therefore $\prod_i^m \mathbf{M}_i$ is a Galaxy space of $Y$.

Based on Theorem 4 and Definition 3.3.6, we can detect a Galaxy space of $Y$ via an ensemble learning algorithm, Galaxy, as follows.

1. Detect a **PMTE** via Theorem 3 on weighted instances in $\mathbb{D}$ and then calculate the misclassification rate $\epsilon$.

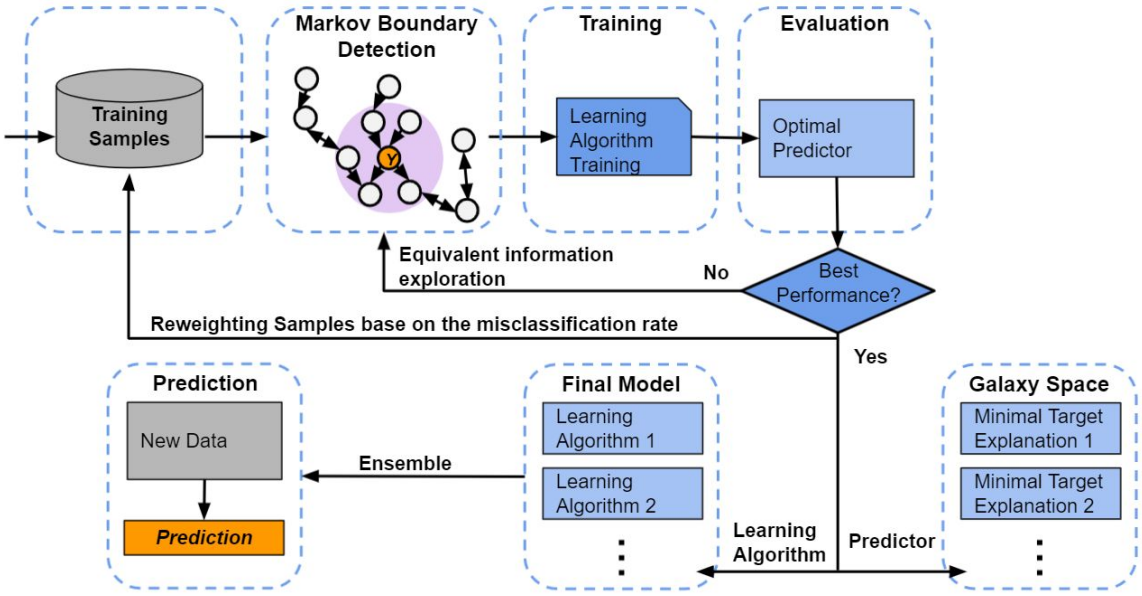



Figure 3.3: The Framework of the Galaxy Model.

2. Compute mixture component weight.

$$\phi_i = \log\left(\frac{1-\epsilon}{\epsilon}\right). \tag{3.4}$$

3. Strengthen the misclassified instances by re-weighting the misclassified instances.

$$w_j = w_j e^{\epsilon \mathbb{I}(h_Y(x_j), y_j)}. \tag{3.5}$$

   where $w_j$ is the weight of the instance $(x_j, y_j)$, $\mathbb{I}$ is the predicting error of $(x_j, y_j)$, where $\mathbb{I} = 0$ if the prediction is correct, otherwise 1.

4. Repeat steps 1 - 3 until the misclassification rate $\epsilon$ lower than a threshold $\delta$.

The Galaxy algorithm, to discover the Galaxy space from mixture distributed feature data, is explained in Algorithm 2 and Fig 3.3.

**Algorithm 2:** The Galaxy algorithm, to discover the Galaxy space from mixture distributed feature data and forecast via its ensemble predictive power.

**Input:**

- data set $\mathbb{D}$ includes $n$ instances; target feature $Y$; Markov boundary detection algorithm $f_Y$; learning algorithm $h_Y$; performance metric T;

**Output:**

- Galaxy space $\mathbb{G}$.
- Galaxy $H_Y$, an trained ensemble algorithm.

**begin**

$\mathbb{G} =$ empty `/* Initialize` $\mathbb{G}$ `with an empty set */`

`/* Initialize the instances' weights` $\mathbf{W}$ `*/`

$\mathbf{W} = \prod_{j=1}^{n} w_j$, where $w_j = \frac{1}{n}$

$\mathbb{I}(h_Y(x_j), y_j)$ `/* Predicting error of the instance` $(x_j, y_j)$`, where` $\mathbb{I} = 0$ `if the prediction is correct, otherwise 1 */`

$i = 1$

**repeat**

$\mathbf{M}_i$, $h_Y(\mathbf{M}_i) =$ PMTE Detection($\mathbb{D}$, $Y$, $f_Y$, $h_Y$, $T$)

`/* calculate the weighted misclassification rate` $\epsilon$`. */`

$\epsilon = \frac{\sum_{j=1}^{n} w_j \mathbb{I}(h_Y(x_j), y_j)}{\sum_{j=1}^{n} w_j}$ $\phi_i = \log\left(\frac{1-\epsilon}{\epsilon}\right)$ `/* calculate mixture weight` $\phi$`. */`

**for** $w_j \in \mathbf{W}$ **do**

$w_j = w_j \exp(\epsilon \mathbb{I}(h_Y(x_j), y_j))$

$i = i + 1$

**until** $\epsilon < \delta$

Return $\mathbb{G} = \prod_i \mathbf{M}_i$, $H_Y = \sum_i \phi_i h_Y(\mathbf{M_i})$

## 3.4 Experimental Evaluation

In this section, we present experiments to evaluate the effectiveness and utility of explanations of Galaxy on synthetic data with different dimensionalities and a real-world precipitation data set. In particular, we address the following questions:

- **Q1. Effectiveness on highly correlated data:** How effective can Galaxy work on highly correlated data?
- **Q2. Interpretation of target explanations:** Are the target explanations detected by Galaxy on real-world data interpretable?

We implemented Galaxy in Python; all experiments were carried out on a 3.0 GHz Intel(R) Xeon(R) E5-2687 Linux server, 1007 GB RAM, running Ubuntu 16.04.2 LTS.

### 3.4.1 Synthetic data generation

In order to simulate highly correlated data that represent data collected in real-world climate applications, we generate a $d$-dimensional synthetic data set using classification data generator in Python package scikit-learn[9] with high redundancy and noise.

Table 3.1: Average F-measure on different dimensional synthetic datasets.

| Dimensionality | Random Forest | AdaBoost | Gradient Boosting | Multilayer Perceptron | **Galaxy (ours)** |
|---|---|---|---|---|---|
| 450 | 0.812 | 0.794 | 0.791 | 0.819 | **0.82** |
| 550 | 0.777 | 0.786 | 0.842 | 0.845 | **0.847** |
| 650 | 0.761 | 0.755 | 0.792 | 0.832 | **0.841** |
| 750 | 0.671 | 0.643 | **0.716** | 0.703 | 0.714 |
| 850 | 0.707 | 0.701 | 0.691 | 0.706 | **0.723** |
| 950 | 0.672 | 0.587 | 0.612 | 0.711 | **0.803** |
| 1050 | **0.776** | 0.773 | 0.781 | 0.726 | 0.772 |
| 1150 | 0.753 | 0.733 | 0.744 | 0.742 | **0.804** |
| 1250 | 0.662 | 0.591 | 0.652 | 0.643 | **0.683** |
| 1350 | 0.764 | 0.734 | 0.758 | 0.802 | **0.824** |

### 3.4.2 Experiment settings

We demonstrate the effectiveness of Galaxy by comparing its F-measure($\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$) against a bench of candidate methods: Random Forest, AdaBoost, Gradient Boosting, and multilayer perceptron. Simulated data were generated with feature counts $d$ ranging from 450 to 1,350 in increments of 100 features. Each of the comparison methods was run against each of the subsets of the overall dataset after feature reduction. All computations were performed on the same hardware and datasets.

### 3.4.3 Q1. Effectiveness on highly correlated data

We report the F-measure for each classifier on the different-dimensional datasets, averaged by 10-fold cross-validation, in Table 3.1. We can see that Galaxy outperforms others most often (8 wins, 2 losses). These results indicate that Galaxy achieve the satisfying predictive power while detecting the Galaxy space.

### 3.4.4 The Precipitation Data Set

Table 3.2: Meteorological variables for precipitation forecasting.

| Name | Level(hPa) |
|---|---|
| Zonal Wind | 200, 500, 850 |
| Meridional Wind | 200, 500, 850 |
| Geopotential Height | 200, 500, 850 |
| Temperature | 200, 500, 850 |
| Relative Humidity | 700, 925 |
| Specific Humidity | 850 |
| Pressure Vertical Velocity | 700 |
| Sea Level Pressure | - |
| Precipitable Water | - |

The real-world dataset we used for the experiments is a subset of the NCEP/NCAR Reanalysis dataset[37] and includes 9 meteorological variables collected at different

vertical levels in the atmosphere (Table 3.2). All the variables are chosen by our domain scientists collaborators based on their physical relevance for precipitation analysis. By convention, atmospheric pressure (in units of hectopascals or hPa) is used as the vertical coordinate with the 1000hPa surface located near the surface and 200hPa near the top of the troposphere. According to the theory of quasi-geostrophic and baroclinic[28], we specially choose 200hPa, 500hPa, and 850hPa zonal winds(i.e. east-west) because they are a proxy for the location and strength of the jet stream which requires wind shear (strong change in wind speed with height) to develop. And the information of the location of the jet stream exhibits persistence on scales much longer than individual storm events. Moreover, 200hPa, 500hPa, and 850hPa meridional (i.e. North-South) winds are chosen because they are extremely important for the transport of heat and moisture from the tropics into the mid-latitudes. The geopotential height at 200hPa, 500hPa, and 850hPa are chosen because the 500hPa field will contain information about Rossby wave propagation, which is a natural phenomenon in the atmosphere and oceans of planets that largely owe their properties to rotation, and the comparison with 200hPa and 850hPa fields allows us to infer where large-scale rising motion (and therefore precipitation) is likely to take place. On the other hand, the temperature at 200hPa, 500hPa, and 850hPa are chosen because the moisture transport is needed to maintain the precipitation while the advection of temperature is crucial for strengthening (weakening) temperature gradients and the production (destruction) of fronts, which are important in producing vertical (i.e. rising) motion. And specific humidity at 850hPa, relative humidity at 700hPa and 925hPa are chosen because the amount of water in the upper troposphere was thought to be negligible. The pressure vertical velocity at 700hPa, precipitable water (total water vapor integrated from the surface to the top of the atmosphere) and sea level pressure (atmospheric pressure at surface corrected to sea level) are also important in producing precipitation.

The total number of variables in all levels is 18. All these meteorological variables are sampled at the spatial domain of $0°E$ to $375.5°E$ and $90°N$ to $20°S$ with a resolution of $2.5° \times 2.5°$ (totally 5,904 locations) and a daily temporal resolution. We pick the samples collected during the rainy season (March to November) during the years 1951-2017. The target feature is the historical spatial average precipitation (the mean of daily precipitation totals from 23 stations) of the Des Moines River basin in Iowa from the same time period.

In the experiments, We set the lead time as 5 days, "look ahead" period as 10 days. For example, to explain rainfall situations at today ($Day_0$) in the study area, we will look for the explanatory features in the time period from $Day_{-14}$(14 days ago) to $Day_{-5}$(five day ago). The precipitation data set presents two particularly difficult characteristics:

- **Extremely high dimensionality:** Each sample has $5,313,600$ features (18 variables $\times$ 5,904 locations $\times$ 10 days)
- **High intra-dataset correlation:** Meteorological variables presented at different levels, locations, and days are highly correlated. Different meteorological variables may also correlated.

### 3.4.5 Q2. Interpretation of target explanations

We run Galaxy on this extremely high dimensionality data set and finally got 15 **PMTE**s, which the minimum size is 4, the maximum size is 13. The top four weighted **PMTE**s are illustrated in Figure 3.4.

The **PMTE**s in Figure 3.4 (A) and Figure 3.4 (D) identifies a geopotential height anomaly over Eastern Europe at 11 days before. This is consistent with a deepening trough over Ural Mountains. Troughs deepening over the Urals are often triggers of wave trains across Asia (the so-called Silk Road pattern) that eventually end up propagating across the Pacific.

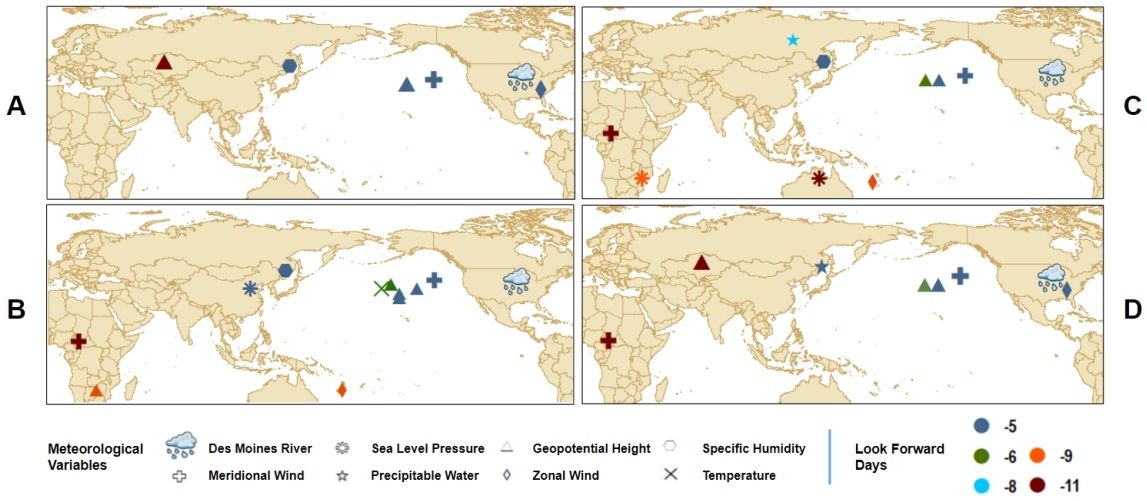



Figure 3.4: Four PMTEs of the precipitation at Des Moines river basin detected by Galaxy.

The **PMTE** in Figure 3.4 (B) includes meridional wind, specific humidity, and upper level (200hPa) geopotential height fields near the east coast of Asia. Many studies have identified cold surges along the Asian coast as important precursors to surface weather over the United States[18, 19]. The surge of cold air and deepening trough typically result in a strengthened jet stream and generate a Rossby wave that propagates across the North Pacific and breaks along the west coast of North America. The mechanism implied by the **PMTE** is that the North-South winds are transporting dry (and presumably cold) air from the north into the middle latitudes around Japan. This pattern results in the deepening of an upper-level trough (negative anomaly in the 200hPa Geopotential Height field) and strengthening of the mid-latitude jet stream. The strengthening of the jet stream in the **PMTE** (i.e. an upper level zonal (u) wind field being chose) was not observed, but it is implied. The upper-level geopotential height anomalies along the west coast of North America on day -5 imply a large, pre-existing upper-level ridge along the west coast of North America[25, 59] that is also very consistent with expectations of strong precipitation

over the central U.S. This pattern suggests a "forcing" for a wave train setting up along the east coast and a pre-existing ridge along the west coast.

## 3.5 Conclusions

This study proposes a new interpretable model, Galaxy, to identify efficient explanations of subpopulations(PMTE) within an overall population of the target feature. We provide the atheoretical framework and implementation details of Galaxy. Furthermore, we use the predictive ensemble power of all PMTEs to make long-term forecasting on the target feature. Finally, our empirical study on the synthetic and real data demonstrates Galaxy's superb performance on prediction and interpretation.

# CHAPTER 4

# WIDENING THE TIME HORIZON: PREDICTING THE LONG-TERM BEHAVIOR OF CHAOTIC SYSTEMS

## 4.1 Introduction

In many physical, biological, and human systems the governing equations are known with high confidence, but the analytic solutions may not exist and reliable numerical solutions are often prohibitively expensive because of nonlinearity, feedback, and sensitive dependence on initial and boundary conditions [48, 7, 64, 76, 70]. Developing reliable numerical solutions that can integrate short length scales and fast time scales is a long-standing problem. Examples include weather forecasting, space-time prediction of virus, and forecasting of the stock market. As this class of governing equations arise in so many varied applications, means to improve the ability to make meaningful predictions of their states at future times are of great practical importance; even a small increase in model performance with respect to applications like long-lead weather prediction for flooding [76, 70] would have wide-ranging societal benefit.

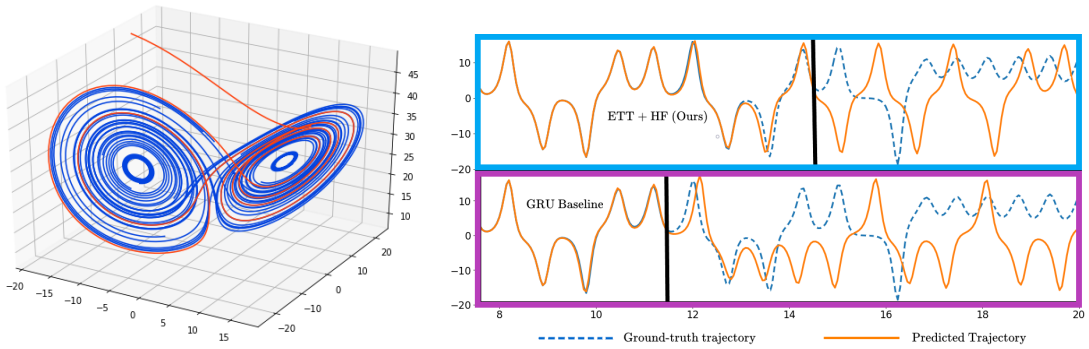
Moreover, many of these highly complex dynamic systems exhibit chaotic behaviors that are highly sensitive to initial and boundary conditions. A small observational error—even truncation error caused by binary representation of values tens of digits past the decimal—can grow exponentially in time. As Lorenz [49] aptly noted, for this class of nonlinear systems, approximately close present states do not necessarily map to approximately close states in the future. More importantly, a small predictive error (from a point in a system's phase space, $\mathbf{u_{t_0}}$, to $\mathbf{u_{t_0}} + \mathbf{e}$, where $\mathbf{e}$ is a small error vector) at a time step $t_0$ may result in exponentially larger future error than that

resulting from a larger initial error (in magnitude) in another direction (from $\mathbf{u_{t_0}}$ to $\mathbf{u_{t_0}} - 2\mathbf{e}$, for example) at the same time step.

Many methods from both statistics and machine learning have been proposed in last few decades [12, 33, 67, 32]. However, under these conditions, reliable prediction of state past a certain time horizon remains extremely challenging, depending on the parameterization of the system and the lead time of the prediction. Our research goal is to push out the time horizon at which reliable predictions can be made as far as possible with new developments from machine learning, and our study has obtained promising results as shown in figure 4.1. More specifically, while there can be theoretical bounds for the accuracy of future prediction given an initial discrepancy of some size for certain systems (e.g. [64] gives an analysis for the Lorenz '63 system), important performance gain is possible to achieve by training a model to avoid local mistakes that result in dramatic changes to future phase space trajectories. In doing so, we avoid the greatest sources of error, and keep the predicted trajectories as close to the ground truth as possible and the predictions relevant in practice. Any such progress could represent a significant step forward in real-world problem domains.

In this work, we present a new deep learning method for prediction in chaotic systems: it takes the form of a recurrent architecture set up for error trajectory tracing and an accompanying training regime, Horizon Forcing, which allows a neural network to both model the system's state transition function and use it to trace the evolution of its mistakes. By using both in tandem a model is able to properly evaluate the consequences of local mistakes as the underlying system evolves toward the time horizon. We make the following contributions:

- We introduce a new recurrent architecture for error trajectory tracing (ETT) in chaotic systems. It is designed to improve the prediction of such systems by allowing the model to optimize its parameters based on the true, longer-term




(a) Lorenz '63 system, a classic example of chaotic systems

(b) Ground-truth and predicted trajectories

Figure 4.1: The time horizon at which reliable predictions on the Lorenz system can be made is pushed further into the future by Horizon forcing. (a) The Lorenz attractor, a representative chaotic system, with an example test trajectory pictured in orange. Trajectories move quickly from their initial (randomly generated; see §4.4.1.2) state to the attractor and follow a chaotic orbit around the two butterfly wings. (b) A test set example ground-truth (blue) and predicted (orange) trajectories for our Horizon-Forced model (top) and a baseline Teacher Forcing model (bottom). We show only the state X coordinate (y axis) for readability. The x axis (time) is in lyapunov times. It is clear that predictions from the baseline model begin to have significant divergence from truth just before 12 lyapunov times and completely lose the correct trajectory around 13.5. Our model shows good agreement out to 15 lyapunov times (defined to be the inverse of the largest lyapunov exponent, the amount of time to accrue error $e$. In the Lorenz system, this is $\approx \frac{1}{0.906}$, or 1.103).

consequences of small initial prediction errors, how the trajectories beginning at the predicted states evolve forward to the time horizon.

- We present Horizon Forcing, a new training regime for optimizing our ETT models. We optimize our cost function based on short-term predictions first, then shift focus to the long term as training progresses. By doing this with shared weights we further improve the single-step error (a proxy for the system's transition function) because minimizing long-term error necessitates controlling short-term error in chaotic systems.

- We extensively validate our method on three well-studied chaotic systems with known dynamics and a set of real-world time series prediction tasks.

## 4.2 Related Work

Enormous effort has gone into the understanding and modeling of chaotic nonlinear dynamical systems using observed data [6]. Examples of system identification techniques based on input-output pairs go back decades: in the 1960s, efforts based on Wiener kernels and Kalman filters [41, 29], in the 1980s, nonlinear autoregressive (NARMAX) methods [6, 42, 14], in the 1990s, neural network approaches [13, 22]. Earlier this century, [8] and [58] used evolutionary methods with symbolic regression to attempt to match up derivatives estimated from data with known derivative functions.

With the explosion of interest in deep learning and proliferation of powerful computing hardware, focus has shifted to the use of powerful models to recreate complex chaotic dynamics. Some recent works [57, 68] directly model the derivatives of the system such that the model learns the vector field over the phase space; other works (including this one) attempt to model trajectories through phase space in a sequence-to-sequence formulation [53, 71]. Below we examine current deep neural methods for chaotic system prediction:

### 4.2.1 Recurrent Approaches

In [68], the authors use a series of LSTM architectures to estimate the value of the vector field at a given point in phase space and integrate it forward to generate the next sequence prediction. [51] uses iterative prediction of points with a feed-forward network trained with Jacobian regularization. [55] build a recurrent network out of residual network cells [26], which have been shown to have similar properties to PDEs [44]. In [71], a new architecture is presented that features a stacked LSTM

encoder-decoder architecture with an"inhibitor" layer that sits above it and relates the next-step state prediction $u_{m+1}$ to each previous state $u_0...u_m$ via a vector of weights $a_0...a_m$, in a manner analogous to an attention layer over previous time-steps.

### 4.2.2 Reservoir Computing

In reservoir computing and echo state networks [36], a recurrent neural network is randomly wired (and called the *reservoir*) and not updated during training [46]. Instead, the chaotic system is modeled as a linear combination of the static nonlinear components of the reservoir [20, 11]. In echo state networks, the reservoir is carefully initialized such that the *echo state property* is ensured, which states that the influence of a past state $u_i$ on future states $u_j$ should approach 0 as $j \to \infty$. In [53], the authors use a combined knowledge-based and reservoir computing method to predict future state evolution of chaotic systems. They show that future predictive performance can be improved by including knowledge of the equations of the system in the model as an additional component, and simulate varying the confidence with which the equations are known by modifying the value of the chaotic parameter $\rho$ of the Lorenz system (see §4.4.1.2 for details) in the knowledge-based component of the model by up to 10% (while holding the true value to be the same), and demonstrate that performance depends on a very accurate estimation of the system parameters.

### 4.2.3 Teacher Forcing in Deep Recurrent Neural Networks

Recurrent deep neural network architectures are typically trained with *teacher forcing*, which replaces the network's own predictions of the value of the input sequence at time step $t$, ($\hat{y}_t$) with the ground-truth sequence values ($y_t$) when predicting the value of the next time step ($\hat{y}_{t+1}$); this prevents errors that compound over time from harming network convergence while training. However, at inference ground-truth values are not available, so the predictions $\hat{y}_t$ must be used, reintroducing the danger of error accumulating over time. Scheduled Sampling [5] proposes to

fix this method by replacing the teacher-forced ground truth inputs with the model's predicted values with a certain probability ($\epsilon_i$, where $i$ indexes the epoch) that is annealed $1 \rightarrow 0$ over the course of training via some choice of an annealing function, effectively weaning the model off of the ground-truth inputs. However, [35] show that this sampling method yields a biased estimator - the loss function induced by Scheduled Sampling is not minimized at the true input distribution. In [40], the authors introduce *Professor Forcing*, which uses an adversarial discriminator to minimize the difference between a model being run in teacher-forced mode and free-running (or inference) mode, by using the outputs of each mode and a function of some of the internal states of the model. This method is difficult to train in practice, requiring the optimization of several hyperparameters [47]. Other methods include Zoneout [39], which is a dropout-like training procedure that instead of setting activations to 0, randomly sets a set of activations to be identical to those at the previous time step. This is meant to regularize the transition dynamics; we do not find this to be an appropriate method for a chaotic system because sensitivity is necessary to fit the system's ground truth.

### 4.2.4 Transformer Architectures

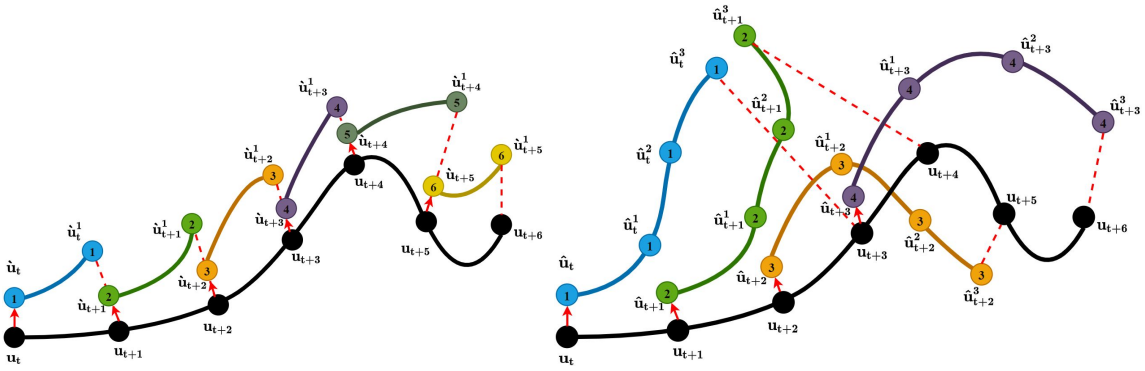
Transformer architectures [67] have revolutionized language modeling in recent years [17] and are now being applied to other domains with success as well (e.g., [52]). [32]'s music transformer applies this architecture to a music generation task, motivated by transformers' ability to continue to generate coherent predictions over long sequences. They introduce an efficient method for encoding relative position information in the model and apply a mask to the sequence, eliminating the model's ability to attend to future time-steps. This formulation is well-suited for application to chaotic sequence prediction, so we use their architecture as our baseline for transformer models in our experiments.

### 4.2.5 The Broad Learning System and Extreme Learning Machines

In [24], the authors use a manifold constraint to make a Broad Learning System (BLS) [12] model better suited for learning manifolds and reconstructing attractors in chaotic systems. The BLS model is a single-layer architecture designed to learn on a series of "wide" concatenated feature maps over the data, which are themselves mapped to additional *enhancement nodes* via a second mapping function to provide nonlinearity. The enhancement nodes and the feature nodes are concatenated and the final output weights are determined by a ridge or lasso regression to the ground-truth labels. The structured manifold iteration of their method, SM-BLS, uses feature maps that are sparse linear approximations of the transformation matrix that maps the time series to a coordinate system determined by taking the graph Laplacian of a nearest-neighbor graph created from the time series using a Gaussian Radial Basis Function.

The SM-BLS method [24] authors additionally find that extreme learning machines [33] are competitive with their method. ELMs [33] are single-layer feedforward models where the first weight matrix (which multiplies the input) and biases are initialized randomly and untrained; the post-activation weights are analytically solved by calculating the pseudoinverse of the output matrix.

Comparing with these existing methods, our approach has two major advantages: (1) it assumes no prior knowledge of the equations describing the underlying system, and (2) it models how its mistakes evolve through the system's dynamics and accumulate over time. To the best of our knowledge, no other work has attempted (2) without assuming prior knowledge of system dynamics.



(a) Calculation of Lyapunov Exponent (b) Error to time horizon

Figure 4.2: Motivation for Lyapunov Horizon Loss. $\grave{\mathbf{u}}_\mathbf{t}$ indicates that a point is due to a perturbation of $\mathbf{u}_t$; $\hat{\mathbf{u}}_t$ is a prediction of $\mathbf{u}_t$ (separation represents modeling error). The superscript at each node is the number of steps taken to reach it on its trajectory, and the subscript is the index of the time step in the original ground-truth series at which the error trajectory began. For example, a label of $\hat{\mathbf{u}}_t^3$ means that the labeled point is a prediction of the location of $\mathbf{u}_{t+3}$, and that the point's trajectory evolved starting from a prediction of $\mathbf{u}_t$ ($\hat{\mathbf{u}}_t$). (a) The lyapunov exponent is estimated from an orbit through phase space by creating an initial perturbation (red arrow) from a starting point ($u_t$), yielding a new point ($\grave{u}_t$) and then computing the logarithm of the ratio of the distance between the $t+1$ terms ($\grave{u}_t^1$ and $u_{t+1}$) and the initial perturbation. At the next iteration, a new point is generated a distance $d_{t+1}$ along the line between the $t+1$ terms and the process is repeated. (b) The inspiration for the proposed error trajectory tracing (ETT) architecture. When a small mistake is made at $u_t$ ($u_t$ is estimated to be $\hat{u}_t$, red arrow), the cost of that error after three steps of evolution is $\|\hat{u}_t^3 - u_{t+3}\|$: the orbit from $\hat{u}_t$ evolves along the blue path to $\hat{u}_t^3$ at the time horizon, while the true state at that time should be the state the same number of time steps ahead, but on the true orbit: $u_{t+3}$. Similarly, the cost of a small error at $u_{t+1}$ (predicting its value to be $\hat{u}_{t+1}$) is $\|\hat{u}_{t+1}^3 - u_{t+4}\|$, as the system evolves along the green orbit from $\hat{u}_{t+1}$ to $\hat{u}_{t+1}^3$ at the time horizon. We do the same analysis for errors at each time step up to the horizon.

## 4.3 Methods: Error Trajectory Tracing and Horizon Forcing

Because reducing local errors at individual time-steps doesn't necessarily optimize the model toward correct long(er)-term predictions of the state sequence, a better way to evaluate local predictive error is needed. See fig. 4.2(b), and refer to Table 4.1 for a description of the notation used. Typically, the model parameters would be

Table 4.1: Notations used in Lyapunov Horizon Loss Derivation

| Notation | Meaning |
|---|---|
| $\mathbf{u}_t$ | Ground-truth state vector at time $t$. |
| $\grave{\mathbf{u}}_t$ | State vector resulting from a small perturbation of $\mathbf{u}_t$. |
| $\hat{\mathbf{u}}_t$ | Model's predicted value of $\mathbf{u}_t$. |
| $\mathbf{u}_t^k$ | State vector resulting from $k$ applications of $F$ to $\mathbf{u}_t$; i.e., $\mathbf{u}_t^k = F^{(k)}(\mathbf{u}_t)$ |
| $\mathbf{u}_j^{(i),k}$ | State vector resulting from $k$ applications of $F$ to $\mathbf{u}_j$ on the $i$th sequence in the dataset. |

updated solely based on, for example, the mean squared error between $\mathbf{u_t}$ and $\hat{\mathbf{u}}_\mathbf{t}$ (the norm of the red arrow between them). We propose to add an additional penalty to the cost of that initial error, by further observing the evolution of that mistake at a time horizon, defined to be a number of time steps in the future chosen as a hyperparameter before modeling (in the figure, this is three steps). This allows us to not only observe the initial error $\|\mathbf{u_t} - \hat{\mathbf{u}}_\mathbf{t}\|$, but the degree to which that initial deviation results in increased error at the time horizon, $\|\mathbf{u_{t+3}} - \hat{\mathbf{u}}_\mathbf{t}^\mathbf{3}\|$. We refer to the objective function incorporating this penalty as the lyapunov horizon loss, and our architecture as performing error trajectory tracing (ETT).

### 4.3.1 Problem formulation

In this work, we study nonlinear systems defined by a set of differential equations that evolve deterministically. We use $x$ or $x(t)$ to denote a function relating states and time and $\dot{x}$ notation for derivatives when using continuous time; we use $\mathbf{u_t}$, $\mathbf{u_{t+1}}$ for discrete-time systems and continuous systems that have been discretized for modeling (e.g. the Lorenz system).

In the most general form, we have:

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \theta). \tag{4.1}$$

where f is a function that determines the system as a function of the state vector $\mathbf{x}(t)$, and is parameterized by a vector $\theta$. Each possible state of the system $\mathbf{x_0}$ is represented as a point in the phase space of the system. From each $\mathbf{x_0}$, the system evolves deterministically along a trajectory in phase space. We can define a transition function, $F$, to discretize the system to a time-step ($\Delta t$) as

$$\begin{aligned} F(\mathbf{x_i}) &= \mathbf{x_i} + \int_t^{t+\Delta t} f\left(\mathbf{x}(\tau)\right) d\tau \\ \mathbf{u_{t+1}} &= F(\mathbf{u_t}) \end{aligned} \tag{4.2}$$

[57] Given a sequence of states in a discretized system $\mathbf{u_0}, \mathbf{u_1}, ..., \mathbf{u_m}$, where each successive state is a repeated application of $F$, our goal is to predict the evolution of the system, represented by the sequence of future states $\mathbf{u_{m+1}}, \mathbf{u_{m+2}}, ...$, where $\mathbf{u_{m+1}} = F(\mathbf{u_m}), \mathbf{u_{m+2}} = F(F(\mathbf{u_m}))$, $\mathbf{u_m} = F^{(m)}(\mathbf{u_0})$ and so on. We assume a training dataset, $\mathbb{D}$, of $N$ trajectories of length $T$:

$$\mathbb{D} = \left\{ \left( \mathbf{u}_0^{(i)}, F^{(1)}(\mathbf{u}_0^{(i)}), ..., F^{(j)}(\mathbf{u}_0^{(i)}), ..., F^{(T)}(\mathbf{u}_0^{(i)}) \right) \right\}$$

$i \in 1...N, j \in 1...T$. We assume no prior knowledge of the system's dynamics apart from the fact it is determined by transition function $F$. In the next section, we discuss how to use this formulation to develop a loss that can penalize error growth over a time horizon using the lyapunov exponent.

### 4.3.2 Lyapunov horizon loss

In chaos theory, the *maximal lyapunov exponent* is a measure of the chaotic nature of a system [64, 62]. It can be understood as the expected logarithm of the growth

rate of unit errors near a strange attractor (specific to the attractor, which depends on the basin of attraction of initial condition $\mathbf{u}_0$): a negative value of the lyapunov exponent indicates stability (as the growth rate is not exponential), while a positive exponent indicates exponential increase in small deviations over time.

$$\lambda_{max} = \mathbb{E}\left[\log\left(\frac{\|\grave{\mathbf{u}}_t^1 - \mathbf{u}_{t+1}\|}{\|\grave{\mathbf{u}}_t - \mathbf{u}_t\|}\right)\right] \tag{4.3}$$

where $\mathbf{u}_t$ and $\grave{\mathbf{u}}_t$ are two nearby points in the phase space of the system (see figure 4.2(a)) and $\mathbf{u}_{t+1}$ and $\grave{\mathbf{u}}_t^1$ are the points they transition into after one iteration. In practice, an attractor's $\lambda_{max}$ is estimated numerically by computing a huge number of terms ($10^9$ or more, in some cases [62]).

Rather than estimating the degree of chaotic expansion of an entire attractor, as $\lambda_{max}$ does, our models improve predictive performance by using local estimations of the error growth: by selecting a time horizon in the near future (by a number of time steps, $k$) and tracking how errors at each time step on the trajectory evolve from intermediate states ($\mathbf{u}_{t_i}$) to the time horizon ($\mathbf{u}_{t_i+k}$)(fig. 4.2(b)). By modeling how trajectories starting at a predicted state evolve, we can penalize errors that grow into large deviations at the time horizon more harshly than errors of the same magnitude that stay close to the true trajectory by adding a lyapunov horizon penalty into the training loss:

$$\begin{aligned}\lambda_{t_i} &= \log\frac{\|\hat{\mathbf{u}}_{t_i}^k - \mathbf{u}_{t_i+k}\|}{\|\hat{\mathbf{u}}_{t_i} - \mathbf{u}_{t_i}\|} \\ &= \log\frac{\left\|d_{t_i}^k\right\|}{\|d_{t_i}\|}\end{aligned} \tag{4.4}$$

where $k$ is the number of time steps to the horizon, and we define $d_k := \|\hat{\mathbf{u}}_{t_i}^k - \mathbf{u}_{t_i+k}\|$ as the distance from the predicted state to the true state, $k$ steps in the future from

a given $t$, which we will specify explicitly when not clear from context. The network parameters $\theta$ will be optimized incorporating this lyapunov horizon loss as follows:

$$\begin{aligned}
\theta^* &= \underset{\theta}{\operatorname{argmin}}\, \mathcal{L} \\
&= \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[ \sum_{j=1}^{T} (\hat{\mathbf{u}}_j^{(i)} - \mathbf{u}_j^{(i)})^2 + \sum_{j=1}^{T-k} \lambda_j^{(i)} \right] \\
&= \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[ \sum_{j=1}^{T} (\hat{\mathbf{u}}_j^{(i)} - \mathbf{u}_j^{(i)})^2 + \sum_{j=1}^{T-k} \log \frac{\|\hat{\mathbf{u}}_j^{(i),k} - \mathbf{u}_{j+k}^{(i)}\|}{\|\hat{\mathbf{u}}_j^{(i)} - \mathbf{u}_j^{(i)}\|} \right] \\
&= \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[ \sum_{j=1}^{T} (d_j^{(i)})^2 + \sum_{j=1}^{T-k} \log \frac{\left\| d_j^{(i),k} \right\|}{\left\| d_j^{(i)} \right\|} \right] \\
&= \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[ \sum_{j=1}^{T} (d_j^{(i)})^2 + \sum_{j=1}^{T-k} \left( \log \left\| d_j^{(i),k} \right\| - \log \left\| d_j^{(i)} \right\| \right) \right]
\end{aligned} \tag{4.5}$$

In the last line of (4.5), the first term is the squared error at time step $j$ on data sequence $i$. The second summation is the local estimation of the error growth rate: the logarithm of the $k$-step error divided by the current-step error. In (4.6), we modify this objective, removing the $-\log \left\| d_j^{(i)} \right\|$ term (as its square is already being optimized to 0 by the first term, leaving $\log \left\| d_j^{(i),k} \right\|$) as the dominating factor) and the logarithm of $\left\| d_j^{(i),k} \right\|$ (as the logarithm makes the magnitude of that term small and difficult to optimize and its removal will not change the value of $\theta^*$ ), defining the new set of optimal parameters to be:

$$\theta^* \equiv \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[ \sum_{j=1}^{T} (d_j^{(i)})^2 + \sum_{j=1}^{T-k} \left\| d_j^{(i),k} \right\| \right] \tag{4.6}$$

where $i$ indexes the $N$ trajectories in the training set, $j$ indexes the $T$ time steps, and $k$ represents the number of time steps from the start step to the horizon step. In the next section we will present a novel deep neural network model called Error Trajectory Tracing for optimal lyapunov horizon loss minimization.

### 4.3.3 Model architecture: Error Trajectory Tracing

We implement the structure of the system in fig. 4.2 (b) as a deep recurrent model using Gated Recurrent Unit (GRU, [15]) cells as building blocks, in such a way that we can calculate and backpropagate the lyapunov horizon loss. We empirically find GRU to be superior to LSTM and feedforward units across hyperparameter settings, so consider only those models here.

**Training.** Recurrent neural networks are usually trained using Teacher Forcing, whereby during training the model receives the ground truth $\mathbf{u}_t$ as input at time $t$ instead of the prediction made by the previous cell, $\hat{\mathbf{u}}_t$. This Teacher Forcing process can help to minimize the one-step prediction error and reduce the vanishing gradient effects that arise when predictions from one state are used as input for the next over many time steps [5].

A GRU cell can be formulated as:

$$\begin{aligned} z_t &= \sigma(W_z \mathbf{u}_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r \mathbf{u}_t + U_r h_{t-1} + b_r) \\ s_t &= \phi_h(W_h \mathbf{u}_t + U_h(r_t \odot h_{t-1}) + b_h) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot s_t \\ \hat{\mathbf{u}}_{t+1} &= W_u h_t + b_u \end{aligned} \tag{4.7}$$

where $z_t$ is the update gate, $r_t$ is the reset gate, and $h_t$ is the hidden state. Our output prediction per cell, $\hat{\mathbf{u}}_{t+1}$, is the output of a linear layer on top of the hidden state and $W_u$ is a weight matrix with rows corresponding to the number of variables in the chaotic system and columns corresponding to the dimensionality of the hidden state.

Parameters $\{W_z, W_r, W_h, W_u, U_z, U_r, U_h, b_z, b_r, b_h, b_u\}$ are updated using backpropagation. Taking $W_z$ as an example (the process is similar for the others), the updates can be computed as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}_t}{\partial W_z} &= \frac{\partial \mathcal{L}_t}{\partial h_t}\frac{\partial h_t}{\partial W_z} \\ &= \frac{\partial \mathcal{L}_t}{\partial h_t}\sum_{i=1}^{t}\left(\frac{\partial h_t}{\partial h_i}\frac{\overline{\partial h_i}}{\partial W_z}\right) \\ &= \frac{\partial \mathcal{L}_t}{\partial h_t}\sum_{i=1}^{t}\left(\left(\prod_{j=i}^{t-1}\frac{\partial h_{j+1}}{\partial h_j}\right)\frac{\overline{\partial h_i}}{\partial W_z}\right)\end{aligned} \tag{4.8}$$

where $\overline{\partial h_j}/\partial W_z$ is the gradient of $\partial h_j$ with respect to $W_z$ while taking $\partial h_{j-1}$ as a constant [43]. And $\partial h_t/\partial h_{t-1}$ is,

$$\begin{aligned}\frac{\partial h_t}{\partial h_{t-1}} &= \frac{\partial h_t}{\partial s_t}\frac{\partial s_t}{\partial h_{t-1}} + \frac{\partial h_t}{\partial z_t}\frac{\partial z_t}{\partial h_{t-1}} + \overline{\frac{\partial h_t}{\partial h_{t-1}}} \\ &= \frac{\partial h_t}{\partial s_t}\left(\frac{\partial s_t}{\partial r_t}\frac{\partial r_t}{\partial h_{t-1}} + \frac{\partial s_t}{\partial h_{t-1}}\right) + \frac{\partial h_t}{\partial z_t}\frac{\partial z_t}{\partial h_{t-1}} + \overline{\frac{\partial h_t}{\partial h_{t-1}}}\end{aligned} \tag{4.9}$$

This use of the chain rule to propagate gradients through previous time steps is called backpropagation through time.

**Inference.** The goal of the trained GRU is to generate predictions of the future time steps (after $m$ steps), and since $\mathbf{u}_t$ is no longer available (having not occurred yet), the model must use its own prediction, $\hat{\mathbf{u}}_t$, from the previous time steps in order to produce $\hat{\mathbf{u}}_{t+1}$. At inference, equation 4.5 then becomes,

$$\begin{aligned} z_t &= \sigma(W_z\hat{\mathbf{u}}_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r\hat{\mathbf{u}}_t + U_r h_{t-1} + b_r) \\ s_t &= \phi_h(W_h\hat{\mathbf{u}}_t + U_h(r_t \odot h_{t-1}) + b_h) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot s_t \\ \hat{\mathbf{u}}_{t+1} &= W_u h_t + b_u \end{aligned} \tag{4.10}$$

This process must repeat $T-m+1$ times; this can lead to small prediction deviations in the early inference steps sharply increasing as the number of inference steps grows

further into the future.

**Error Trajectory Tracing.** To force the GRU cells to learn how the error evolves over time and reduce prediction deviations at later steps, we build the Error Trajectory Tracing architecture as follows (See figure 4.3): first, we build a bottom (zero) layer in the form of a standard GRU RNN. But at each step in this layer, we transfer the hidden state and output to two other cells: the next cell in the zero layer (represent hidden state ($h_t$) and another cell (the next level up on the diagram) representing the next state in the trajectory starting at the prediction (the first step on the error trajectory, $\hat{\mathbf{u}}_t$). We then extend each of these towers of GRU cells until we reach the $k_{th}$ (horizon) layer. Each of these GRU "towers" traces the evolution of the trajectory starting with the initial 1-step predictive error from the original trajectory. Each successive GRU cell in the tower represents an additional application of the transition function $F$. Using backpropagation through each GRU "tower", $\partial h_t/\partial h_{t-1}$ can be updated according to eq. 4.11 (terms differing from standard backpropagation through time are indicated from below with brackets):

$$\begin{aligned}
\frac{\partial h_t}{\partial h_{t-1}} &= \frac{\partial h_t}{\partial s_t}\frac{\partial s_t}{\partial h_{t-1}} + \frac{\partial h_t}{\partial z_t}\frac{\partial z_t}{\partial h_{t-1}} + \overline{\frac{\partial h_t}{\partial h_{t-1}}} \\
&= \frac{\partial h_t}{\partial s_t}\left(\frac{\partial s_t}{\partial \hat{\mathbf{u}}_t}\overline{\frac{\partial \hat{\mathbf{u}}_t}{\partial h_{t-1}}} + \frac{\partial s_t}{\partial r_t}\left(\overline{\frac{\partial r_t}{\partial h_{t-1}}} + \frac{\partial r_t}{\partial \hat{\mathbf{u}}_t}\overline{\frac{\partial \hat{\mathbf{u}}_t}{\partial h_{t-1}}}\right) + \overline{\frac{\partial s_t}{\partial h_{t-1}}}\right) \\
&+ \frac{\partial h_t}{\partial z_t}\left(\overline{\frac{\partial z_t}{\partial h_{t-1}}} + \frac{\partial z_t}{\partial \hat{\mathbf{u}}_t}\overline{\frac{\partial \hat{\mathbf{u}}_t}{\partial h_{t-1}}}\right) + \overline{\frac{\partial h_t}{\partial h_{t-1}}}
\end{aligned} \tag{4.11}$$

All of the GRU cells in this architecture share weights. These shared weights in the GRU cells are our model's parameterization of $F$; by tuning them with both the horizontal GRU and the forward evolutions of the error from each step, we are able to use this forward-looking training method to tune our representation to have strong

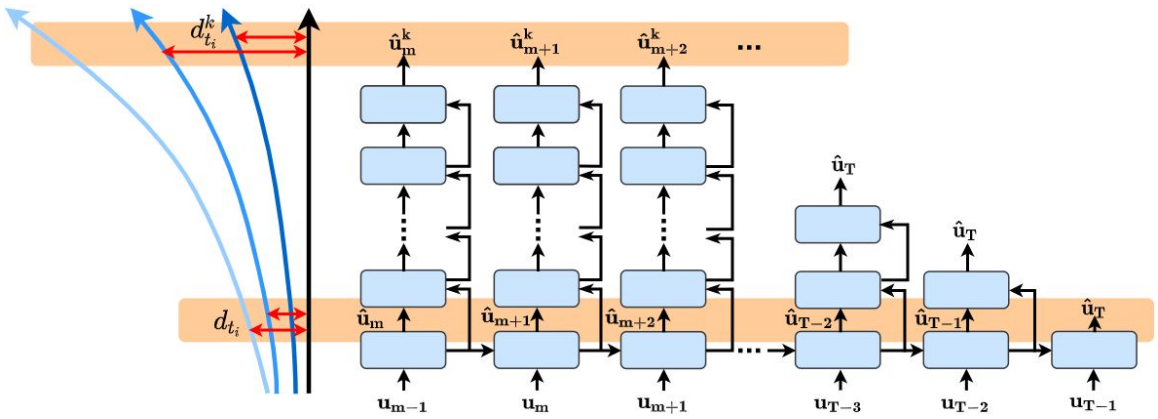



Figure 4.3: Our error trajectory tracing architecture for time-horizon prediction. Each rounded rectangle represents a GRU cell, all of which share weights. The trajectories on the left represent modeled trajectories (blue) against the ground-truth (black), illustrating the improvement resulting from training on a model's own predictions from teacher forcing to horizon forcing. Each tower models the future evolution of the chaotic system from the prediction made after a single step (lower orange-shaded box). Our first-stage training minimizes $d_{t_i}$ (from the longer red arrow at the bottom of the figure to the arrow immediately above it). The upper orange-shaded box represents optimization at the time horizon ($k+1$ steps); the later training stage minimizes $d^k_{t_i}$ (from the longer red arrow at the top of the figure to the arrow immediately above it), the time horizon loss.

short-and-long-term performance. In next section we will present a novel training procedure for optimal training of our ETT architecture.

### 4.3.4 A novel training procedure: Horizon Forcing

We apply a novel training approach to our ETT architecture. See Figure 3 and Algorithm 3.

1. First, we train the $0_{th}$ layer (the bottom orange-shaded box in Fig. 3) using standard teacher forcing to reach a strong one-step-ahead baseline. When this layer is well trained, the divergence $d_j^{(i)}$ (in eq. 4.5) will decrease and the prediction trajectory will be tightened in to the ground truth trajectory. This will reduce one-step prediction error, the difference $\|\hat{\mathbf{u}}_j^{(i)} - \mathbf{u}_j^{(i)}\|$ in eq. 4.5.

**Algorithm 3:** Horizon Forcing.

**Input:**

- number of steps $n$;
- step size $k$;

**Output:**

- trained predictive model$hf$.

**begin**

$D \leftarrow$ Data() /* Load $\mathbb{D}$ as def. in §4.3.1 */
$bsl \leftarrow$ BaseModel() /* Initialize Baseline */
$bsl$.fit($D$) /* Train Baseline */
$hf \leftarrow$ HFModel($bsl, k$) /* Add ETT architecture */
$hf$.fit($D$) /* Train model for Horizon $k$ */
**for** $i$ = *2 to* $n$ **do**
  /* Initialize $hf_{i*k}$ using $hf_{(i-1)*k}$ as a baseline */
  $hf \leftarrow$ HFModel($hf, i * k$)
  $hf$.fit($D$) /* Train model for horizon $k * i$ */
Return $hf$ /* Return model for horizon $n * k$ */

2. After the $d_j^{(i)}$ training task converges, we start to train the horizon ($k_{th}$) layer (the top orange-shaded box in Fig. 3). This further reduces the $(k+1)$ step prediction error $d_j^{(i),k}$ during inference, optimizing out the large future errors that result from the small errors made by the converged one-step model.

3. Optionally, step 2 can be repeated to further extend the horizon by using the resulting model from step 2 as a new baseline. As outlined in algorithm 3, this process yields a model with a horizon of $n * k$, where $n$ is the number of times step 2 has been completed (e.g., in our experiments, we train HF10 with $n = 2$ and $k = 5$).

A benefit of our Horizon Forcing approach is that when we train the horizon ($k_{th}$) layer, all of the GRU cells will be updated because their weights are shared. All outputs and states in the ETT architecture are updated during each training session,

so minimizing the $(k+1)$ step prediction error will continue minimizing the one step prediction error, which is difficult to achieve when training a standalone GRU.

## 4.4 Experiments

**Models.** We validate the utility of the Horizon Forcing method on a deep recurrent network architecture by using $k = 5$ and $n \in [1, 2, 3, 4]$, yielding a teacher-forcing model (1-step ahead), and Horizon-Forced models with horizons at 5, 10, 15, and 20 time-steps ahead for all experiments. The RNN architecture is a single-layer GRU with 256 latent units. We compare our Horizon Forcing training regime with Teacher Forcing and Scheduled Sampling [5] regimes optimizing a standard GRU network with the same cell dimensionality. We run Scheduled sampling under two experimental settings: 1) where we generate a full prediction vector and sample from it at each time step with probability $1 - \epsilon_i$ (which leaves each sampled prediction independent) and where we allow for early stopping identically to other methods (called Scheduled Sampling - Early Stopping or SSES in our results table) and 2) where errors are computed sequentially (and thus are dependent) and we enforce that the entire schedule of sampling probabilities must be completed at training (and which we call Scheduled Sampling - Full Schedule or SSFS). We run all SSFS models with an inverse sigmoid decay schedule, which dramatically outperformed linear or exponential schedules during validation. We also benchmark against BLS (§4.2.5), ELM (§4.2.5), and a Music Transformer (§4.2.4). Our BLS model has 100 latent units (5 windows and 20 latent units per window) and 31 enhanced units; ELM has 500 latent units; the music transformer has four stacked encoders, each with four attention heads (64 latent units per head). To avoid the influence of training hyperparameters, we use 100 sequence time steps as input across all experiments and only tune the learning rate, $r$.

All experiments were performed on a private Linux GPU server with 48 CPU cores, 1TB RAM, and 8 Nvidia 1080ti GPUs (each with ∼11GB memory). Each model was trained using a single GPU, and was implemented in Tensorflow 2.0. We use a batch size of 30 trajectories throughout all experiments. The maximum epochs are set to 200 but early stopping is employed to stop training if no improvement is made after 15 epochs. We also reduce the learning rate by 10% after 5 epochs of no improvement.

**Datasets.** We study datasets corresponding to several chaotic systems and real-world time series: the well-known Rössler (§4.4.1.1) and Lorenz (§4.4.1.2) systems, as well as a Lotka-Volterra Ecosystem and a suite of real-world time series compiled in [21] to evaluate attractor reconstruction (§4.4.2).

**Evaluation and metrics.** Because input data is not available during inference and the models need to use the prediction from the previous time step as the input of the next step in order to continue making predictions, small deviations from the ground-truth in the early inference steps tend to increase sharply as the predicted horizon extends. Furthermore, we observe that in some problem domains the error can accumulate to such a degree that past a certain time step, any resemblance a predicted sequence has with the true sequence (as quantified by an evaluation metric) is simply coincidence or luck and no longer the result of having a useful long-term representation of the system's dynamics.

For this reason, in this study we employ a set of commonly-used evaluation metrics that compare a single predicted step with a single ground-truth step (see table 4.2), and monitor their change over time. Because we value a model's ability to make useful predictions for as long as possible, we evaluate the competing methods by the expected amount of predictions (number of time steps into the future) that can be made before an error of a certain magnitude (as measured by metric $M$) occurs. We

Table 4.2: Table of metrics for forecasting analysis

| Metric | Expression | Symbols: Expectation & Effective Range |
|---|---|---|
| RMSE | $\gamma\left(\mathbf{u}_j^{(i)}, \hat{\mathbf{u}}_j^{(i)}\right) = \sqrt{\frac{1}{D}\sum_{d=1}^{D}\left\|\hat{\mathbf{u}}_{j,d}^{(i)} - \mathbf{u}_{j,d}^{(i)}\right\|^2}$ | $\mathbb{E}_\gamma; \mathbb{P}_\gamma$ |
| MNE | $\zeta\left(\mathbf{u}_j^{(i)}, \hat{\mathbf{u}}_j^{(i)}\right) = \frac{1}{D}\sum_{d=1}^{D}\frac{\left\|\hat{\mathbf{u}}_{j,d}^{(i)} - \mathbf{u}_{j,d}^{(i)}\right\|}{\left\|\mathbf{u}_{j,d}^{(i)}\right\|}$ | $\mathbb{E}_\zeta; \mathbb{P}_\zeta$ |
| SMAPE | $\mu\left(\mathbf{u}_j^{(i)}, \hat{\mathbf{u}}_j^{(i)}\right) = \frac{1}{D}\sum_{d=1}^{D}\frac{\left\|\hat{\mathbf{u}}_{j,d}^{(i)} - \mathbf{u}_{j,d}^{(i)}\right\|}{\left\|\hat{\mathbf{u}}_{j,d}^{(i)}\right\| + \left\|\mathbf{u}_{j,d}^{(i)}\right\|}$ | $\mathbb{E}_\mu; \mathbb{P}_\mu$ |

refer to this as the model's *effective range*. This is computed as follows: we take a (possibly multidimensional) input sequence and predict the future trajectory for a set number of time steps. We compute the per time-step error (using $M$) for each step on the predicted trajectory, and then compute an average error per time-step over all test set sequences (see figure 4.4 for examples). We take the first (minimum $j$) step where the error (or percent error, depending on choice of $M$) is greater than a given threshold—that step is considered the length of time for which the average error trajectory is valid, and we report that value as the effective range:

$$\mathbb{P}_M = \min_j \left\{ j \;\middle|\; \frac{1}{N}\sum_{i=1}^{N} M\left(\mathbf{u}_j^{(i)}, \hat{\mathbf{u}}_j^{(i)}\right) > \delta_M \right\} \tag{4.12}$$

Here $\mathbb{P}_M$ is the effective range as measured by metric $M$ for the model making predictions $\hat{\mathbf{u}}_t^{(i)}$.

$\delta_M$ denotes the threshold being used. A balance must be struck with respect to this value: if $\delta_M$ is chosen to be too low, the predicted ranges will be short and noisy and fail to differentiate between methods that perform well and those that perform

poorly. If $\delta_M$'s value is taken to be too large, then predicted ranges will be long and also fail to differentiate strong from weak methods: in some cases, it's possible that a high $\delta_M$ could result in the threshold never being reached for a given sequence, in which case that sequence's contribution to the predicted range would be its length, greatly increasing the range value in a manner that is not meaningful for evaluation. Therefore, we seek a low threshold that isn't immediately crossed in practice. We find that among the datasets we use and methods we compare, setting $\delta_M$ to be the average performance of the best and second-best methods meets our criteria and allows us to avoid the use of arbitrarily selected thresholds. We report the $\delta_M$ values used for each dataset in table 4.3, along with the number of variables and inference steps used.

We compute effective range with respect to three metrics: *Root Mean Squared Error* (RMSE), *Mean Normalized Error* (MNE), and *Symmetric Mean Absolute Percent Error* (SMAPE), the computation of which we detail in Table 4.2; $j$ is the time step at which the metric is being calculated and $D$ is the number of dimensions at time step $j$, indexed by $d$.

We also report the expectation with respect to each metric, computed as follows:

$$\mathbb{E}_M = \frac{1}{N \times T} \sum_{i=1}^{N} \sum_{j=1}^{T} M\left(\mathbf{u}_j^{(i)}, \hat{\mathbf{u}}_j^{(i)}\right) \tag{4.13}$$

### 4.4.1 Known Chaotic Systems

#### 4.4.1.1 Rössler System

In 1976, O.E. Rössler presented a system related to the Lorenz '63 system, that exhibited chaotic dynamics with only a single nonlinearity ($zx$) in the defining equations [56]:

Table 4.3: Table of experimental settings for chaotic systems forecasting

| Data set | Vars | Inference steps | $\delta_\gamma$ | $\delta_\zeta$ | $\delta_\mu$ |
|---|---|---|---|---|---|
| Rössler | 3 | 1300 | 1.863 | 0.1705 | 0.082 |
| Lorenz | 3 | 200 | 3.1065 | 0.228 | 0.1105 |
| L-V Ecosystem | 10 | 150 | 1.243 | 0.066 | 0.0305 |
| Accelerometer | 3 | 300 | 0.2935 | 0.3635 | 0.182 |
| Roaming Worm | 5 | 160 | 2.265 | 0.8575 | 0.4615 |
| Gait Force | 1 | 300 | 158.6875 | 0.501 | 0.2565 |
| Electricity | 1 | 300 | 121.54 | 0.219 | 0.0595 |

$$\begin{aligned} \dot{x} =& - y - z \\ \dot{y} =& x + ay \\ \dot{z} =& b + z(x - c) \end{aligned} \tag{4.14}$$

Unlike the Lorenz attractor (figure 4.1(a)), the Rössler system has only a single spiral, which, as trajectories cycle around it, stretches out (into the Z dimension) and folds over onto itself, creating a Cantor set of layered surfaces [64]. For certain parameterizations, the flow is stable and non-periodic but all trajectories are unstable. [56].

**Data Generation.** We use the parameters $a = 0.1$, $b = 0.1$, and $c = 18$, which is dense and chaotic. We generate $X$, $Y$, and $Z$ using SciPy's ODE solver with a step size of $\Delta t = 0.05$, and divide the sequence into individual training sequences by striding the original sequence with an offset of five time steps. This yields 6,400 training sequences and 1,001 testing sequences. During training, we further split the train set into 80% train and 20% validation.

#### 4.4.1.2 Lorenz '63 system

The Lorenz '63 system is a dynamical system presented by Edward Lorenz in [45] as a means to study some of the chaotic aspects of the atmosphere in tractable equations. The system represents a simplified model of convection in a fluid layer between two surfaces held at a constant temperature differential, and offers some desirable characteristics [69]:

- The system is fully deterministic. However, errors accumulate exponentially with time, making perfect predictions possible only for short time windows.
- For certain parameterizations (see below), there is no periodicity, and solutions never resolve to a single stationary state.

The system is parameterized by three values, $\sigma, \rho$, and $\beta$; $\sigma$ is the *Prandtl number*, the ratio of kinematic diffusivity to thermodynamic conductivity; $\rho$ is proportional to the *Rayleigh number* of the fluid, indicative of the degree of turbulence in the system and dependent on the magnitude of temperature differential; $\beta$ is representative of the dimensions of the system, in that it is a function of the ratio of the distance between the surfaces and the (assumed) period of the field function and temperature variation (see [73] for details).

The system has three variables, $X, Y$, and $Z$, which are related by the following system of differential equations (dots denote the derivative of a variable with respect to time):

$$\begin{aligned} \dot{x} =& \sigma(y - x) \\ \dot{y} =& x(\rho - z) - y \\ \dot{z} =& xy - \beta z \end{aligned} \tag{4.15}$$

When generating data, we set $\sigma, \rho$, and $\beta$ to $10, 28$, and $\frac{8}{3}$. These values were presented in the original '63 paper as settings that yield chaotic dynamics, and are

commonly used in literature. The most common modification of these parameters is changing the value of $\rho$, which governs the degree to which the system is chaotic [1].

**Data Generation.** We employ SciPy's ODE solver using the fourth-order Runge-Kutta method and the system in equation 4.15 to generate $X, Y$ and $Z$ using a $\Delta t$ of 0.05 for each step. We use a stride of five to divide the sequence into 8,500 training examples and 1,201 testing examples, with a further 80%-20% train-validation split.

#### 4.4.1.3 Lotka-Volterra Ecosystem

We also benchmark against a 10-variable (five competing species, five resources) Generalized Lotka-Volterra Ecosystem simulated by [21] using a community matrix with parameters provided in [34], where the authors argue that chaotic dynamics that arise from the system could be an explanation for the observed diversity in oceanic phytoplankton being higher than theory predicts possible. We split the available data into 5,200 training sequences and 1,001 testing sequences using a stride of three, and again use an 80%-20% train-validation split. The horizon-forced models outperform the other methods in both expectation and effective range, and have the strongest prediction performance as shown by the average error sequences in Figure 4.5 until approximately 100 time steps in the future, when all models become similar.

### 4.4.2 Real-World Datasets

[21] compiles a suite of real-world data time series for evaluation of chaotic attractor reconstruction. While they study a problem formulation different from our own (determining the chaotic attractor of a system from one of its components, e.g., the sequence of $X$ values in the Lorenz system), the collection of datasets they provide offer a valuable framework on which to benchmark our approach. We use the fol-

lowing data sets, with each training sequence being 100 time steps, split on sequence identifier (entire sequences are assigned to be in exactly one of train or test):

- **Roaming Worm**. [2] provides a time series that tracks the evolution of the curvature of worm *C. Elegans*; the dimensionality of each step has been reduced to five with PCA. We use a stride of four to section the dataset into 5,000 training sequences (of which 20% are reserved for validation) and 1,649 testing sequences.

- **Accelerometer**. [66] contains accelerometer readings from a gait database for a study participant (id 3, following [21]) walking with a smartphone. This dataset contains three variables, which record acceleration with respect to $X$-, $Y$-, and $Z$-axes. We use 5,500 train samples and 801 test samples, generated with a stride of two.

- **Gait Force**. Dataset includes gait force measurements for a walking subject (id 2), collected from [27]. Univariate. We use 5,600 train samples and 1,023 test samples, with a stride of nine.

- **Electricity**. Time series of the mean power usage (in kilowatts) by 321 clients of a Portuguese power company from 2011 to 2014, sampled every 15 minutes. Part of the UCI Machine Learning Repository [1], preprocessed by [21]. We used a stride of 23 time steps to generate 5,000 train samples and 1,081 test samples.

### 4.4.3 Ablation Study: Horizon-Forced Models and Baseline GRU

We first present evidence of Horizon Forcing's efficacy over the baseline, a GRU model trained with Teacher Forcing. Its results across all datasets as measured by

[1] download available at
http://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

RMSE (other metrics available in Table 4.5) are provided in Table 4.4. As we expect, each Horizon-Forced model outperforms the baseline model across all datasets.

Table 4.4: Performance comparison results between Horizon Forcing Models and Baseline

| Data sets | Metrics | Baseline | HF5 | HF10 | HF15 | HF20 |
|---|---|---|---|---|---|---|
| *Lorenz* | $\mathbb{E}_\gamma$ | 3.944 | 3.206 | 3.007 | **2.975** | 3.053 |
| | $\mathbb{P}_\gamma$ | 107 | 121 | 128 | **129** | 127 |
| *Rössler* | $\mathbb{E}_\gamma$ | 9.679 | **1.790** | 1.936 | 1.904 | 1.903 |
| | $\mathbb{P}_\gamma$ | 107 | **757** | 752 | 751 | 756 |
| *L-V Ecosystem* | $\mathbb{E}_\gamma$ | 1.915 | 1.318 | 1.168 | 1.120 | **1.103** |
| | $\mathbb{P}_\gamma$ | 48 | 73 | 81 | 83 | **85** |
| *Roaming Worm* | $\mathbb{E}_\gamma$ | 3.215 | 2.333 | 2.302 | 2.239 | **2.212** |
| | $\mathbb{P}_\gamma$ | 29 | 53 | 55 | 58 | **60** |
| *Accelerometer* | $\mathbb{E}_\gamma$ | 0.333 | 0.294 | 0.293 | 0.281 | **0.260** |
| | $\mathbb{P}_\gamma$ | 92 | 123 | 124 | 138 | **208** |
| *Gait Force* | $\mathbb{E}_\gamma$ | 202.483 | 162.578 | **154.797** | 155.979 | 157.724 |
| | $\mathbb{P}_\gamma$ | 93 | 136 | **144** | 143 | 140 |
| *Electricity* | $\mathbb{E}_\gamma$ | 159.618 | 123.850 | 119.230 | 112.004 | **98.447** |
| | $\mathbb{P}_\gamma$ | 111 | 121 | 123 | 163 | **207** |

### 4.4.4 Ablation Study: Choice of Horizon

Here, we present a comparison of our model over choices of time horizon with $k$ and $n$ in Algorithm 3 set to $k$=5 and $n \in [1, 2, 3, 4]$, resulting in models trained with respect to horizons at 5, 10, 15, and 20 time steps. Results can be found in Table 4.4.

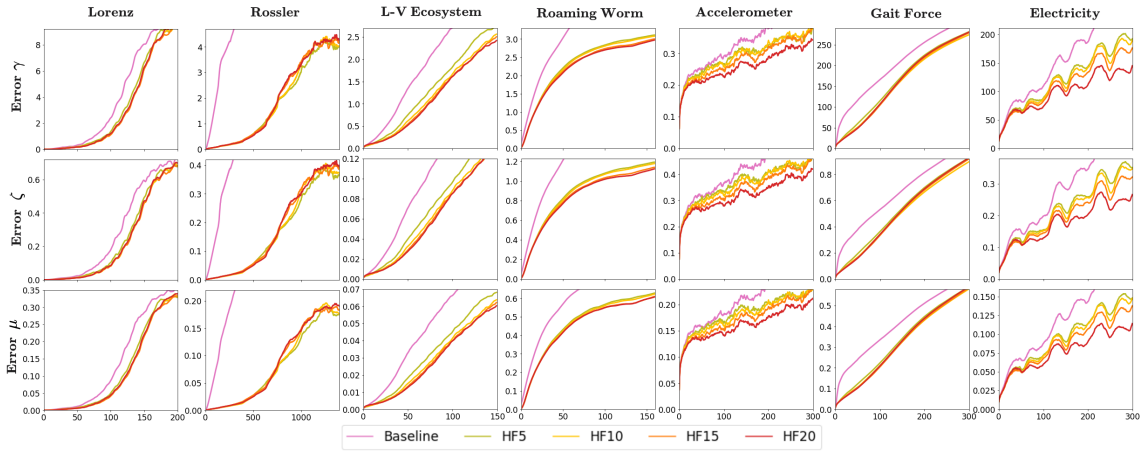



Figure 4.4: Average value of each error metric ($y$ axis) by time step ($x$ axis) and model for each of our evaluation systems. Curves that stay closer to the bottom right of each plot (high time step, low error) are best. We see that the Horizon-Forced models outperform the GRU baseline, and improve with longer horizon in most cases.

We can see that HF models are robust to the choice of training metric (not so with all methods; see §4.4.5)—with few exceptions, when an HF model is best in one metric it is best in all metrics.

In Figure(4.4), generally, training out to a longer time horizon improves performance; on 4 of 7 datasets, HF20 is the best model. When it is not, we see that the results are fairly close across horizons (effective ranges within 1-2% of the other models) but on those datasets where there are large differences the longer horizon dramatically outperforms the others.

### 4.4.5 Benchmarking

Expected values of metrics and their effective ranges are presented in Table 4.5, and full error sequences (average error over all test sequences at each time step) are shown in Figure 4.5. Lower is better for all expected values and higher is better for all predicted ranges. Horizon-Forced models are consistently better than the alternative methods, and the average error sequences (Fig. 4.5) show that in some cases (the BLS model on gait force, notably) strong performance in Table 4.5 from another

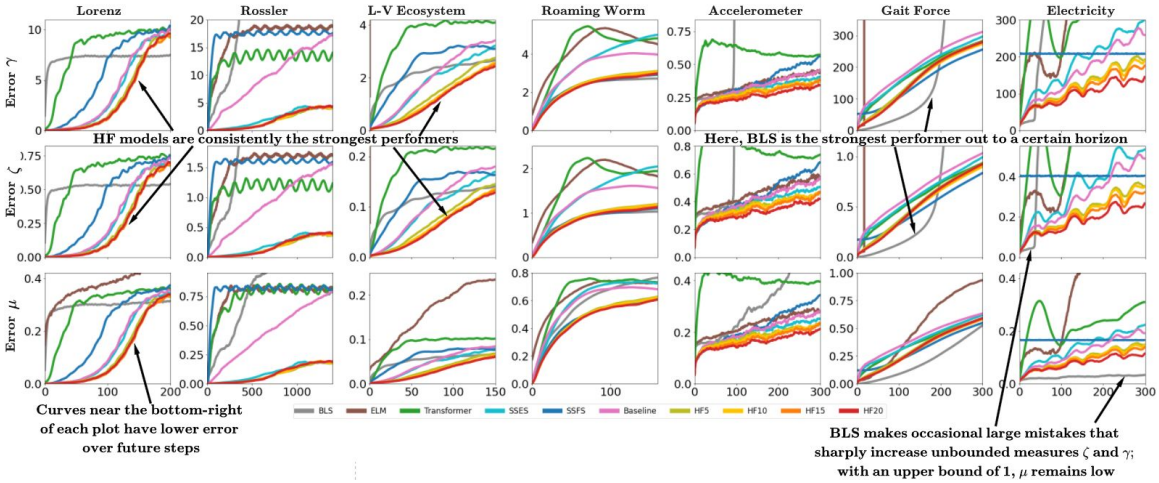



Figure 4.5: Error curves illustrating benchmark error accumulation over time for each dataset, model type, and metric. Curves that stay closer to the bottom right of each plot (high time step, low error) are best. We see that the Horizon-Forced models consistently have lower error over a longer time than alternative methods (HF curves are beneath others). Note that some combinations of method and dataset are favored by some metrics; we report a set of metrics to get a complete picture. For example, BLS sees a sharp increase in loss at 200 time-steps when predicting the Gait Force dataset, which is smoothed out by $\mu$ (SMAPE). We do not observe this behavior in the RNN-based models, highlighting the need for multiple evaluation metrics. On the Electricity dataset, BLS has good baseline performamce but makes occasional very large mistakes that result in a spike of the average in the unbounded error measures; as SMAPE has an upper bound at 1, the importance of these mistakes are limited and the error remains low.

model is actually indicative of a tradeoff with HF. The BLS model on gait force has very strong initial predictive performance but has a sharp increase in average error around 200 time-steps in the future (this is smoothed out by SMAPE, which has BLS as the clear best performer); if predictions beyond that horizon were necessary in practice, Horizon-Forced models would still be preferred. We can also see SMAPE strongly favoring another method (again BLS) on the electricity dataset—BLS has very low expected SMAPE and the clear lowest SMAPE average error sequence in Figure 4.5 (bottom row, far right plot), but this is actually the result of a small number of predicted values with very large error magnitude. The inclusion of the predicted value in the denominator of SMAPE upper bounds its value at 1, which

limits the impact of the occasional error spikes on the expected SMAPE, resulting in improved performance by that metric. With its strong performance by SMAPE and weak performance when measured with RMSE or Normalized Error, the value of the model in such a situation would depend on the details of the task and may be cause for concern in practice.

We additionally find that Scheduled Sampling run to its full schedule (SSFS) performs worse than the version that we allow to stop when validation loss stops decreasing (SSES) on four of the seven datasets. SSFS outperforms SSES on the Roaming Worm dataset (remaining lower than the HF models) and on the Gait Force dataset, where its results are excellent (only BLS is better) and HF underperforms. SSFS also fails to properly converge on the Electricity dataset, which we believe is due to the that dataset being a stable time series with the presence of "spikes" in usage that can lead to high predictive error (or not) at random based on sampling chance.

Table 4.5: Performance comparison results between Horizon Forcing and Benchmarks

| Data set | Models | Metrics | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathbb{E}_\gamma$ | $\mathbb{P}_\gamma$ | $\mathbb{E}_\zeta$ | $\mathbb{P}_\zeta$ | $\mathbb{E}_\mu$ | $\mathbb{P}_\mu$ |
| *Lorenz* | BLS | 7.201 | 2 | 0.519 | 2 | 0.295 | 2 |
| | ELM | – | 0 | – | 0 | 0.38 | 0 |
| | Transformer | 8.206 | 17 | 0.610 | 16 | 0.294 | 17 |
| | SSES | 3.806 | 110 | 0.279 | 109 | 0.135 | 110 |
| | SSFS | 5.619 | 73 | 0.409 | 71 | 0.201 | 73 |
| | Baseline | 3.944 | 107 | 0.289 | 105 | 0.141 | 106 |
| | HF5 | 3.206 | 121 | 0.235 | 121 | 0.114 | 123 |

*Continued on next page*

Table 4.5 – *Continued from previous page*

| Data set | Models | Metrics | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathbb{E}_\gamma$ | $\mathbb{P}_\gamma$ | $\mathbb{E}_\zeta$ | $\mathbb{P}_\zeta$ | $\mathbb{E}_\mu$ | $\mathbb{P}_\mu$ |
| | HF10 | 3.007 | 128 | 0.221 | 127 | 0.107 | 127 |
| | HF15 | **2.975** | **129** | **0.219** | **128** | **0.106** | **128** |
| | HF20 | 3.053 | 127 | 0.224 | 126 | 0.108 | 126 |
| | BLS | – | 30 | – | 29 | 0.777 | 26 |
| *Rössler* | ELM | 16.917 | 3 | 1.545 | 3 | 0.758 | 2 |
| | Transformer | 12.828 | 13 | 1.147 | 13 | 0.742 | 14 |
| | SSES | 2.201 | 624 | 0.201 | 620 | 0.096 | 635 |
| | SSFS | 17.043 | 15 | 1.556 | 15 | 0.798 | 15 |
| | Baseline | 9.679 | 107 | 0.880 | 124 | 0.437 | 120 |
| | HF5 | **1.790** | **757** | **0.163** | **768** | **0.079** | **766** |
| | HF10 | 1.936 | 752 | 0.178 | 755 | 0.085 | 754 |
| | HF15 | 1.904 | 751 | 0.174 | 758 | 0.084 | 752 |
| | HF20 | 1.903 | 756 | 0.175 | 764 | 0.084 | 757 |
| | BLS | 2.147 | 11 | 0.113 | 11 | 0.054 | 11 |
| *L-V Ecosystem* | ELM | – | 0 | – | 0 | 0.159 | 0 |
| | Transformer | 3.548 | 9 | 0.187 | 9 | 0.088 | 9 |
| | SSES | 1.747 | 50 | 0.092 | 50 | 0.043 | 50 |
| | SSFS | 2.475 | 22 | 0.131 | 22 | 0.061 | 22 |
| | Baseline | 1.915 | 48 | 0.101 | 48 | 0.047 | 48 |
| | HF5 | 1.318 | 73 | 0.070 | 73 | 0.032 | 73 |
| | HF10 | 1.168 | 81 | 0.062 | 81 | 0.029 | 81 |
| | HF15 | 1.120 | 83 | 0.059 | 84 | 0.028 | 83 |

*Continued on next page*

Table 4.5 – *Continued from previous page*

| Data set | Models | Metrics | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathbb{E}_\gamma$ | $\mathbb{P}_\gamma$ | $\mathbb{E}_\zeta$ | $\mathbb{P}_\zeta$ | $\mathbb{E}_\mu$ | $\mathbb{P}_\mu$ |
| | HF20 | **1.103** | **85** | **0.058** | **85** | **0.027** | **85** |
| | BLS | 2.228 | 52 | 0.843 | 52 | 0.565 | 42 |
| *Roaming Worm* | ELM | 4.283 | 14 | 1.745 | 12 | 0.648 | 18 |
| | Transformer | 4.164 | 22 | 1.687 | 22 | 0.651 | 23 |
| | SSES | 3.449 | 32 | 1.383 | 31 | 0.591 | 33 |
| | SSFS | 2.240 | 53 | 0.838 | 56 | 0.462 | 55 |
| | Baseline | 3.215 | 29 | 1.280 | 27 | 0.576 | 32 |
| | HF5 | 2.333 | 53 | 0.885 | 54 | 0.465 | 57 |
| | HF10 | 2.302 | 55 | 0.872 | 56 | 0.458 | 59 |
| | HF15 | 2.239 | 58 | 0.843 | 59 | 0.449 | **61** |
| | HF20 | **2.212** | **60** | **0.831** | **61** | **0.448** | **61** |
| | BLS | 43.834 | 80 | 59.822 | 66 | 0.312 | 64 |
| *Accelerometer* | ELM | 0.353 | 65 | 0.457 | 45 | 0.231 | 45 |
| | Transformer | 0.592 | 4 | 0.769 | 3 | 0.399 | 3 |
| | SSES | 0.312 | 113 | 0.388 | 111 | 0.196 | 91 |
| | SSFS | 0.362 | 80 | 0.439 | 88 | 0.222 | 79 |
| | Baseline | 0.333 | 92 | 0.413 | 92 | 0.207 | 89 |
| | HF5 | 0.294 | 123 | 0.366 | 123 | 0.184 | 123 |
| | HF10 | 0.293 | 124 | 0.361 | 125 | 0.180 | 124 |
| | HF15 | 0.281 | 138 | 0.345 | 138 | 0.173 | 140 |
| | HF20 | **0.260** | **208** | **0.320** | **214** | **0.162** | **209** |
| | BLS | 996.947 | **189** | 3.159 | **186** | **0.199** | **195** |
| *Gait Force* | | | | | | | |

*Continued on next page*

Table 4.5 – *Continued from previous page*

| Data set | Models | Metrics | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathbb{E}_\gamma$ | $\mathbb{P}_\gamma$ | $\mathbb{E}_\zeta$ | $\mathbb{P}_\zeta$ | $\mathbb{E}_\mu$ | $\mathbb{P}_\mu$ |
| | ELM | – | 18 | – | 18 | 0.497 | 93 |
| | Transformer | 180.482 | 109 | 0.576 | 112 | 0.364 | 80 |
| | SSES | 188.083 | 105 | 0.602 | 107 | 0.381 | 77 |
| | SSFS | **149.072** | 159 | **0.466** | 167 | 0.310 | 126 |
| | Baseline | 202.483 | 93 | 0.651 | 90 | 0.410 | 58 |
| | HF5 | 162.578 | 136 | 0.514 | 139 | 0.328 | 115 |
| | HF10 | 154.797 | 144 | 0.488 | 147 | 0.314 | 120 |
| | HF15 | 155.979 | 143 | 0.495 | 145 | 0.314 | 122 |
| | HF20 | 157.724 | 140 | 0.500 | 143 | 0.318 | 120 |
| | BLS | – | 41 | 27.871 | 39 | **0.026** | **300** |
| *Electricity* | ELM | 497.149 | 6 | 0.818 | 9 | 0.425 | 1 |
| | Transformer | 330.938 | 9 | 0.652 | 10 | 0.229 | 5 |
| | SSES | 184.876 | 73 | 0.338 | 41 | 0.139 | 23 |
| | SSFS | 208.360 | 0 | 0.403 | 0 | 0.165 | 0 |
| | Baseline | 159.618 | 111 | 0.292 | 114 | 0.122 | 29 |
| | HF5 | 123.850 | 121 | 0.223 | 123 | 0.096 | 61 |
| | HF10 | 119.230 | 123 | 0.215 | 124 | 0.093 | 63 |
| | HF15 | 112.004 | 163 | 0.201 | 166 | 0.088 | 65 |
| | HF20 | **98.447** | **207** | **0.177** | **209** | 0.078 | 100 |

### 4.4.6 Discussion

We have shown our method effective for use in a number of simulated chaotic systems and real-world datasets. Here, we discuss the situations in which our approach is most valuable.

HF/ETT is designed to iteratively optimize over error trajectories every $k$ steps in the future (i.e., optimizing 1-step, $k$-step, $2k$-step error, etc), which is ideal for systems with exponential error growth and where trajectories starting at initially close points (say, a ground-truth state in a phase space and a good model prediction of that state) could diverge dramatically (or not) in response to small changes in their positions.

In such cases, creating an initial model of the dynamics (1-step error), then using that model to sample the error at increasingly distant horizons to enable the learning of the compounding effect (that is known to be present in chaotic systems) is an effective approach. However, systems could exist (especially non-chaotic systems) where the monotonic nature of the exponential divergence we picture in Figure 4.3 does not hold, and either 1) 1-step error is strongly correlated with future error and HF/ETT is not necessary or 2) the error evolves in such a way that sampling 1- and $k$-step errors are not able to capture the behavior of, say, $\frac{k+1}{2}$-step errors due to periodicity or other correlations within the data. We do expect this effect to be small in practice (especially for small values of $k$).

## 4.5 Conclusion

Chaotic systems are found across many fields of study, including climatology, biology, virology, and many others. Improved methods to model the dynamics of such systems have the potential to offer broad utility in a number of applications. Here, we have introduced a new recurrent architecture, error trajectory tracing, and accompanying training regime, Horizon Forcing, for prediction of chaotic systems;

instead of merely minimizing local error at each time-step, we monitor how those errors evolve at a time horizon so the model can optimize its parameters based on an estimation of the future cost of a small present error. By comparing with state-of-the-art machine learning methods, we validate our method on three widely studied simulated systems and show it to be highly effective in making predictions on chaotic systems with sensitive dependence on initial conditions, and further validate against time series data from a number of real-world problem domains.

# CHAPTER 5

# CONCLUSIONS AND FUTURE DIRECTIONS

Because of the high complexity, chaos, and uncertainty that come along with data distributed in space and time, long-term predictive modeling on big Spatio-temporal data is a major challenge across many fields of scientific study. In this thesis, we have targeted two different considerations for this challenge: a technique for identifying the most relevant and meaningful features from the original Spatio-temporal feature space, and a method for modeling complex space-time dynamics with sensitive dependence on initial and boundary conditions.

## 5.1 Conclusions

In Chapter 3, We first talked about predictability and interpretability under single distribution, and defined the notion of Partial Minimal Target Explanation(PMTE) to represent the most relevant and meaningful features to the target instances derived from the same population. We proposed a Multiple Markov Boundaries based algorithm to detect PMTE. Moreover, we further discussed the predictability and interpretability on mixed distribution, and defined the notion of Galaxy space to represent the most meaningful feature sets with global faithfulness of the overall population of the target feature. We proposed the Galaxy algorithm to discover the Galaxy space from the mixed distribution by a boosting process, and then do forecasts by its ensemble predictive power. In our empirical experiments, our algorithm outperforms state-of-the-art ensemble methods on high-dimensional feature spaces.

In Chapter 4, we discussed the challenges of forecasting long-term dynamics in nonlinear dynamical systems, including sensitive dependencies on initial conditions

and exponential error divergence. We design a new recurrent architecture for error trajectory tracing (ETT) to simulate temporal dynamics. It allows the model to optimize its parameters based on the true, longer-term consequences of minor initial prediction errors and how the trajectories beginning at the predicted states evolve forward to the time horizon. Moreover, We present Horizon Forcing, a new training regime for optimizing our ETT models. It begins with a short-term error evaluation and then focuses on the long term as training progresses. By doing this with shared weights, we can decrease the error divergence on long-term steps and further reduce the error from the initial condition. Finally, we extensively validate our method on three well-studied chaotic systems with known dynamics and a set of real-world time series prediction tasks.

## 5.2 Future Work

In this thesis, we have provided novel approaches for handling two major problems in long-term forecasting tasks on big Spatio-temporal data. But there are still several interesting areas for future work.

### 5.2.1 Distributed Galaxy

The main challenge of the Galaxy is the efficiency when dealing with larger and larger data sets, because the complexity of the combined problem increases exponentially, which will affect the time required to solve the problem and adversely affect the quality of the solution. It may be interesting to employ a distributed solution to solve this challenge.

### 5.2.2 Physics-informed Horizon Forcing

While Horizon Forcing is designed to improve model training for long-term prediction in high-dimensional Spatio-temporal data sets, it could be susceptible to overfitting when large amounts of high-quality, labeled training data are not available.

Therefore, the method could benefit from regularization based on physical information about the system to help the model training find an optimal solution on small-size observations.

# BIBLIOGRAPHY


[1] *Tearing Mechanisms: A3*. John Wiley and Sons, Ltd, 2007, ch. 8, pp. 323–342.

[2] Ahamed, Tosif, Costa, Antonio C., and Stephens, Greg J. Capturing the continuous complexity of behavior in c. elegans. *bioRxiv* (2019).

[3] Aliferis, Constantin F, Statnikov, Alexander, Tsamardinos, Ioannis, Mani, Subramani, and Koutsoukos, Xenofon D. Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *Journal of Machine Learning Research 11*, Jan (2010), 171–234.

[4] Aliferis, Constantin F, Tsamardinos, Ioannis, and Statnikov, Alexander. Hiton: a novel markov blanket algorithm for optimal variable selection. In *AMIA Annual Symposium Proceedings* (2003), vol. 2003, American Medical Informatics Association, p. 21.

[5] Bengio, Samy, Vinyals, Oriol, Jaitly, Navdeep, and Shazeer, Noam. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems* (2015), pp. 1171–1179.

[6] Billings, Stephen A. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[7] Boeing, Geoff. Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction. *Systems 4*, 4 (2016), 37.

[8] Bongard, Josh, and Lipson, Hod. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences 104*, 24 (2007), 9943–9948.

[9] Buitinck, Lars, Louppe, Gilles, Blondel, Mathieu, Pedregosa, Fabian, Mueller, Andreas, Grisel, Olivier, Niculae, Vlad, Prettenhofer, Peter, Gramfort, Alexandre, Grobler, Jaques, Layton, Robert, VanderPlas, Jake, Joly, Arnaud, Holt, Brian, and Varoquaux, Gaël. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), pp. 108–122.

[10] Campos, Cassio P de, and Ji, Qiang. Efficient structure learning of bayesian networks using constraints. *Journal of Machine Learning Research 12*, Mar (2011), 663–689.

[11] Carroll, Thomas L. Using reservoir computers to distinguish chaotic signals. *Physical Review E 98*, 5 (2018), 052209.

[12] Chen, CL Philip, and Liu, Zhulin. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE transactions on neural networks and learning systems 29*, 1 (2017), 10–24.

[13] Chen, Sheng, Billings, SA, and Grant, PM. Non-linear system identification using neural networks. *International journal of control 51*, 6 (1990), 1191–1214.

[14] Chen, Sheng, and Billings, Steve A. Representations of non-linear systems: the narmax model. *International journal of control 49*, 3 (1989), 1013–1032.

[15] Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[16] Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[17] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[18] Dole, Randall M. Persistent anomalies of the extratropical northern hemisphere wintertime circulation: Structure. *Monthly weather review 114*, 1 (1986), 178–207.

[19] Dole, Randall M, and Gordon, Neil D. Persistent anomalies of the extratropical northern hemisphere wintertime circulation: Geographical distribution and regional persistence characteristics. *Monthly Weather Review 111*, 8 (1983), 1567–1586.

[20] Gallicchio, Claudio, and Micheli, Alessio. Deep echo state network (deepesn): A brief survey. *arXiv preprint arXiv:1712.04323* (2017).

[21] Gilpin, William. Deep reconstruction of strange attractors from time series. *Advances in neural information processing systems* (2020).

[22] Gonzalez-Garcia, R, Rico-Martinez, R, and Kevrekidis, IG. Identification of distributed parameter systems: A neural net based approach. *Computers & chemical engineering 22* (1998), S965–S968.

[23] Graves, Alex, Liwicki, Marcus, Fernández, Santiago, Bertolami, Roman, Bunke, Horst, and Schmidhuber, Jürgen. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence 31*, 5 (2008), 855–868.

[24] Han, Min, Feng, Shoubo, Chen, CL Philip, Xu, Meiling, and Qiu, Tie. Structured manifold broad learning system: A manifold perspective for large-scale chaotic time series analysis and prediction. *IEEE Transactions on Knowledge and Data Engineering 31*, 9 (2018), 1809–1821.

[25] Harr, Patrick A, and Dea, Jonathan M. Downstream development associated with the extratropical transition of tropical cyclones over the western north pacific. *Monthly Weather Review 137*, 4 (2009), 1295–1319.

[26] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[27] Hebenstreit, F., Leibold, A., Krinner, S., Welsch, G., Lochmann, M., and Eskofier, B.M. Are relative gait phase durations speed dependent? *submitted* (2014).

[28] Held, Isaac M, Pierrehumbert, Raymond T, Garner, Stephen T, and Swanson, Kyle L. Surface quasi-geostrophic dynamics. *Journal of Fluid Mechanics 282* (1995), 1–20.

[29] Ho, BL, and Kálmán, Rudolf E. Effective construction of linear state-variable models from input/output functions. *at-Automatisierungstechnik 14*, 1-12 (1966), 545–548.

[30] Hochreiter, Sepp, and Schmidhuber, Jürgen. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[31] Hua, Kaixun, and Simovici, Dan A. Long-lead term precipitation forecasting by hierarchical clustering-based bayesian structural vector autoregression. In *Networking, Sensing, and Control (ICNSC), 2016 IEEE 13th International Conference on* (2016), IEEE, pp. 1–6.

[32] Huang, Cheng-Zhi Anna, Vaswani, Ashish, Uszkoreit, Jakob, Shazeer, Noam, Simon, Ian, Hawthorne, Curtis, Dai, Andrew M, Hoffman, Matthew D, Dinculescu, Monica, and Eck, Douglas. Music transformer. *arXiv preprint arXiv:1809.04281* (2018).

[33] Huang, Guang-Bin, Zhu, Qin-Yu, and Siew, Chee-Kheong. Extreme learning machine: theory and applications. *Neurocomputing 70*, 1-3 (2006), 489–501.

[34] Huisman, Jef, and Weissing, Franz J. Biodiversity of plankton by species oscillations and chaos. *Nature 402*, 6760 (Nov 1999), 407–410.

[35] Huszár, Ferenc. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101* (2015).

[36] Jaeger, Herbert, and Haas, Harald. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science 304*, 5667 (2004), 78–80.

[37] Kalnay, Eugenia, Kanamitsu, Masao, Kistler, Robert, Collins, William, Deaven, Dennis, Gandin, Lev, Iredell, Mark, Saha, Suranjana, White, Glenn, Woollen, John, et al. The ncep/ncar 40-year reanalysis project. *Bulletin of the American meteorological Society 77*, 3 (1996), 437–471.

[38] Kawale, Jaya, Chatterjee, Snigdhansu, Ormsby, Dominick, Steinhaeuser, Karsten, Liess, Stefan, and Kumar, Vipin. Testing the significance of spatio-temporal teleconnection patterns. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), ACM, pp. 642–650.

[39] Krueger, David, Maharaj, Tegan, Kramár, János, Pezeshki, Mohammad, Ballas, Nicolas, Ke, Nan Rosemary, Goyal, Anirudh, Bengio, Yoshua, Courville, Aaron, and Pal, Chris. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305* (2016).

[40] Lamb, Alex M, Goyal, Anirudh Goyal Alias Parth, Zhang, Ying, Zhang, Saizheng, Courville, Aaron C, and Bengio, Yoshua. Professor forcing: A new algorithm for training recurrent networks. In *Advances in neural information processing systems* (2016), pp. 4601–4609.

[41] Lee, YW, and Schetzen, M. Measurement of the wiener kernels of a non-linear system by cross-correlation. *International Journal of Control 2*, 3 (1965), 237–254.

[42] Leontaritis, IJ, and Billings, Stephen A. Input-output parametric models for non-linear systems part i: deterministic non-linear systems. *International journal of control 41*, 2 (1985), 303–328.

[43] Li, Minchen. A tutorial on backward propagation through time (bptt) in the gated recurrent unit (gru) rnn.

[44] Long, Zichao, Lu, Yiping, Ma, Xianzhong, and Dong, Bin. Pde-net: Learning pdes from data. *arXiv preprint arXiv:1710.09668* (2017).

[45] Lorenz, Edward N. Deterministic nonperiodic flow. *Journal of the atmospheric sciences 20*, 2 (1963), 130–141.

[46] Lukoševičius, Mantas, and Jaeger, Herbert. Reservoir computing approaches to recurrent neural network training. *Computer Science Review 3*, 3 (2009), 127–149.

[47] Martinez, Julieta, Black, Michael J, and Romero, Javier. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2891–2900.

[48] May, Robert M. Simple mathematical models with very complicated dynamics. *Nature 261*, 5560 (1976), 459–467.

[49] Mitchell, Melanie. *Complexity: A Guided Tour*. Oxford University Press, Inc., USA, 2009.

[50] Nielsen, Jens D., Kočka, Tomáš, and Peña, Jose M. On local optima in learning bayesian networks. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, USA, 2003), UAI'03, Morgan Kaufmann Publishers Inc., pp. 435–442.

[51] Pan, Shaowu, and Duraisamy, Karthik. Long-time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity 2018* (2018).

[52] Parmar, Niki, Vaswani, Ashish, Uszkoreit, Jakob, Kaiser, Łukasz, Shazeer, Noam, Ku, Alexander, and Tran, Dustin. Image transformer. *arXiv preprint arXiv:1802.05751* (2018).

[53] Pathak, Jaideep, Wikner, Alexander, Fussell, Rebeckah, Chandra, Sarthak, Hunt, Brian R, Girvan, Michelle, and Ott, Edward. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science 28*, 4 (2018), 041101.

[54] Pearl, Judea. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[55] Qin, Tong, Wu, Kailiang, and Xiu, Dongbin. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics 395* (2019), 620–635.

[56] Rössler, Otto E. An equation for continuous chaos. *Physics Letters A 57*, 5 (1976), 397–398.

[57] Rudy, Samuel H, Kutz, J Nathan, and Brunton, Steven L. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics 396* (2019), 483–506.

[58] Schmidt, Michael, and Lipson, Hod. Distilling free-form natural laws from experimental data. *science 324*, 5923 (2009), 81–85.

[59] Small, David, Atallah, Eyad, and Gyakum, John R. An objectively determined blocking index and its northern hemisphere climatology. *Journal of Climate 27*, 8 (2013), 2948–2970.

[60] Smith, James A, Villarini, Gabriele, and Baeck, Mary Lynn. Mixture distributions and the hydroclimatology of extreme rainfall and flooding in the eastern united states. *Journal of Hydrometeorology 12*, 2 (2011), 294–309.

[61] Spirtes, Peter, Glymour, Clark N, Scheines, Richard, and Heckerman, David. *Causation, prediction, and search*. MIT press, 2000.

[62] Sprott, Julien Clinton. *Chaos and time-series analysis*, vol. 69. Citeseer, 2003.

[63] Statnikov, Alexander, Lytkin, Nikita I, Lemeire, Jan, and Aliferis, Constantin F. Algorithms for discovery of multiple markov boundaries. *Journal of Machine Learning Research 14*, Feb (2013), 499–566.

[64] Strogatz, Steven H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

[65] Tsamardinos, Ioannis, and Aliferis, Constantin F. Towards principled feature selection: relevancy, filters and wrappers. In *AISTATS* (2003).

[66] Vajdi, Amir, Zaghian, Mohammad Reza, Farahmand, Saman, Rastegar, Elham, Maroofi, Kian, Jia, Shaohua, Pomplun, Marc, Haspel, Nurit, and Bayat, Akram. Human gait database for normal walk collected by smart phone accelerometer. *arXiv preprint arXiv:1905.03109* (2019).

[67] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.

[68] Vlachas, Pantelis R, Byeon, Wonmin, Wan, Zhong Y, Sapsis, Themistoklis P, and Koumoutsakos, Petros. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474*, 2213 (2018), 20170844.

[69] Wallis, G. Sparrow, c., the lorenz equations: Bifurcations, chaos, and strange attractors. berlin-heidelberg-new york, springer-verlag 1982. xii, 269 s., 91 abb., dm 54,—. us $ 21.60. isbn 3-540-90775-0 (applied mathematical sciences 41). *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 64*, 1 (1984), 71–71.

[70] Wang, Dawei, Ding, Wei, Yu, Kui, Wu, Xindong, Chen, Ping, Small, David L, and Islam, Shafiqul. Towards long-lead forecasting of extreme flood events: a data mining framework for precipitation cluster precursors identification. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), pp. 1285–1293.

[71] Wang, Rui, Kalnay, Eugenia, and Balachandran, Balakumar. Neural machine-based forecasting of chaotic dynamics. *Nonlinear Dynamics* (2019), 1–15.

[72] Wang, Tong, Chen, Ping, and Li, Boyang. Predicting the quality of short narratives from social media. *arXiv preprint arXiv:1707.02499* (2017).

[73] Wouters, Jeroen. A brief introduction to the lorenz'63 system.

[74] Yu, Kui, Wu, Xindong, Ding, Wei, and Pei, Jian. Towards scalable and accurate online feature selection for big data. In *Data Mining (ICDM), 2014 IEEE International Conference on* (2014), IEEE, pp. 660–669.

[75] Yu, Lei, and Liu, Huan. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (2003), pp. 856–863.

[76] Zhuang, Yong, Small, David L, Shu, Xin, Yu, Kui, Islam, Shafiqul, and Ding, Wei. Galaxy: Towards scalable and interpretable explanation on high-dimensional and spatio-temporal correlated climate data. In *2018 IEEE International Conference on Big Knowledge (ICBK)* (2018), IEEE, pp. 146–153.

[77] Zhuang, Yong, Yu, Kui, Wang, Dawei, and Ding, Wei. An evaluation of big data analytics in feature selection for long-lead extreme floods forecasting. In *Networking, Sensing, and Control (ICNSC), 2016 IEEE 13th International Conference on* (2016), IEEE, pp. 1–6.