ONLINE LEARNING ALGORITHMS FOR MINING TRAJECTORY DATA AND THEIR APPLICATIONS

By

Ding Wang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2021

ABSTRACT

ONLINE LEARNING ALGORITHMS FOR MINING TRAJECTORY DATA AND THEIR APPLICATIONS

By

Ding Wang

Trajectories are spatio-temporal data that represent traces of moving objects, such as humans, migrating animals, vehicles, and tropical cyclones. In addition to the geo-location information, a trajectory data often contain other (non-spatial) features describing the states of the moving objects. The time-varying geo-location and state information would collectively characterize a trajectory dataset, which can be harnessed to understand the dynamics of the moving objects. This thesis focuses on the development of efficient and accurate machine learning algorithms for forecasting the future trajectory path and state of a moving object. Although many methods have been developed in recent years, there are still numerous challenges that have not been sufficiently addressed by existing methods, which hamper their effectiveness when applied to critical applications such as hurricane prediction. These challenges include their difficulties in terms of handling concept drifts, error propagation in long-term forecasts, missing values, and nonlinearities in the data.

In this thesis, I present a family of online learning algorithms to address these challenges. Online learning is an effective approach as it can efficiently fit new observations while adapting to concept drifts present in the data. First, I proposed an online learning framework called `OMuLeT` for long-term forecasting of the trajectory paths of moving objects. `OMuLeT` employs an online learning with restart strategy to incrementally update the weights of its predictive model as new observation data become available. It can also handle missing values in the data using a novel weight renormalization strategy.

Second, I introduced the `OOR` framework to predict the future state of the moving object. Since the state can be represented by ordinal values, `OOR` employs a novel ordinal loss function to train its model. In addition, the framework was extended to `OOQR` to accommodate a

quantile loss function to improve its prediction accuracy for larger values on the ordinal scale. Furthermore, I also developed the OOR-ϵ and OOQR-ϵ frameworks to generate real-valued state predictions using the ϵ insensitivity loss function.

Third, I developed an online learning framework called JOHAN, that simultaneously predicts the location and state of the moving object. JOHAN generates its predictions by leveraging the relationship between the state and location information. JOHAN utilizes a quantile loss function to bias the algorithm towards predicting more accurately large categorical values in terms of the state of the moving object, say, for a high intensity hurricane.

Finally, I present a deep learning framework to capture non-linear relationships in trajectory data. The proposed DTP framework employs a TDM approach for imputing missing values, coupled with an LSTM architecture for dynamic path prediction. In addition, the framework was extended to ODTP, which applied an online learning setting to address concept drifts present in the trajectory data.

As proof of concept, the proposed algorithms were applied to the hurricane prediction task. Both OMuLeT and ODTP were used to predict the future trajectory path of a hurricane up to 48 hours lead time. Experimental results showed that OMuLeT and ODTP outperformed various baseline methods, including the official forecasts produced by the U.S. National Hurricane Center. OOR was applied to predict the intensity of a hurricane up to 48 hours in advance. Experimental results showed that OOR outperformed various state-of-the-art online learning methods and can generate predictions close to the NHC official forecasts. Since hurricane intensity prediction is a notoriously hard problem, JOHAN was applied to improve its prediction accuracy by leveraging the trajectory information, particularly for high intensity hurricanes that are near landfall.

ACKNOWLEDGEMENTS

Pursuing a PhD is a fantastic journey in my life. During this long journey, I met wonderful people and have many wonderful memories. I would like to sincerely acknowledge them for their support and help during my PhD study. This journey has greatly expanded my knowledge and will continue to influence my career life.

First and foremost, I would like to thank my advisor, Dr. Pang-Ning Tan for the continuous support of my PhD study and research. I got my undergraduate and master degrees in physics, but I always have a strong interest in computer science. Thanks to Dr. Tan for giving me the opportunity to join the field of computer science. He is a very knowledgeable, patient and responsible mentor, and he is willing to spend time communicating with students and discussing their researches. Under his guidance, I have learned how to conduct research correctly and how to identify and solve problems encountered in research. These knowledge will be the great wealth I have in my life. It is my honor to work with Dr. Tan, and I will always remember his guidance.

Second, I would like to thank my PhD committee members, Dr. Abdol-Hossein Esfahanian, Dr. Jiayu Zhou and Dr. Pouyan Nejadhashemi for their support and guidance throughout my PhD program. I had taken the course Algorithmic Graph Theory from Dr. Esfahanian and course Introduction to Machine Learning from Dr. Zhou. They are all very interesting and useful courses. I learned a lot from these courses and applied the knowledge to my research. Dr. Pouyan provided me with many suggestions on geographical perspectives. This is very helpful for my research, because my research focuses on the application of online algorithms in hurricane forecasting. I am very happy to work with everyone on my committee, and I am very grateful for their support and help in my research.

Furthermore, I would like to thank all my current and previous colleagues, especially, Boyang Liu, Jianpeng Xu, Xi Liu, Shuai Yuan, Qiaozi Gao, Shaohua Yang, Farzan Masrour, Tyler Wilson, Courtland VanDam and Zubin Abraham. We not only help each other in

research, but are also good friends in life.

In addition, this thesis is partially supported by the National Science Foundation under grant #IIS-1615612. I would also like to thank the Department of Computer Science and Engineering for providing me with fellowships during my thesis writing.

Last but not least, I want to thank my parents and wife for their encouragement, support and unconditional love to me. Without their support, I would not be able to accomplish my PhD degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Trajectory comes from modern Latin word *trajectorium*, which means the "path described by a body moving under the influence of given forces"[1]. Although the path of a moving object is continuous and contains infinite number of positions and time points, trajectory datasets typically contain only a subset of these positions measured at discrete time points.

1.1 Trajectory Dataset

The trajectory of a moving object is characterized by its spatial and temporal properties. The spatial property corresponds to its location and is typically denoted by a 2-dimensional (or 3-dimensional) vector, e.g., $\mathbf{x} = (x_i, y_i) \in \mathbb{R}^2$. The trajectory path of an object can thus be represented by a set of data points $P = \{p_0, p_1, ..., p_n\}$, where $p_i = (x_i, y_i, t_i)$ and t_i denote the i-th time step, as shown in Figure 1.1.

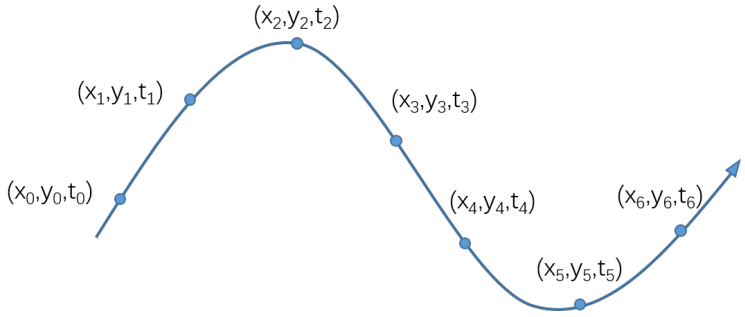


Figure 1.1: Example of a trajectory. Blue line is the trace of a moving object. Blue dots are the sample points.

In addition to its time-varying location information, a trajectory dataset often contains other (non-spatial) features describing the state of the moving object. For example, the state of a moving vehicle could be the aggressive level of its driver; the state of a tropical cyclone could be its intensity level; while the state of a pedestrian could be his/her walking speed

[1] https://www.etymonline.com/word/trajectory

and mood. Assuming the state and path of the moving object is recorded at discrete time steps $t_1, t_2, \cdots, t_T$, the overall trajectory data is represented as $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{s}_i, t_i)\}_{i=1}^{T}$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $\mathbf{s}_i \in \mathbb{R}^d$ are its respective location and state vectors at time t_i.

A trajectory dataset can be collected in many ways, depending on the application domain. In some applications, the trajectory data is recorded by the moving object itself using a Global Positioning System (GPS) sensor or other geo-positioning devices. For example, travellers can use their GPS-equipped cellphones to record the trajectory paths of their journeys. In other applications, the trajectories are recorded by external observers who monitored the paths traversed by the moving objects. For example, the paths of a hurricane are constantly monitored by the National Oceanic and Atmospheric Administration (NOAA)[2]. Another example is the vehicle trajectory data obtained from traffic monitoring cameras, where the trajectories for multiple vehicles can be collected at the same time.

1.2 Trajectory Data Mining

Trajectory data mining is an important task to discover useful information from trajectory datasets. A general framework for trajectory data mining is shown in Figure 1.2. After the data is collected, preprocessing is often needed to alleviate any data quality issues before applying data mining algorithms. Examples of data preprocessing steps include noise reduction, data compression, map matching, and trajectory segmentation. The processed dataset can then be used for subsequent trajectory mining tasks.

Similar to traditional data mining, trajectory mining tasks can be classified into several categories, such as clustering, classification, regression, pattern mining, and outlier detection. There are many applications that utilize the results of these trajectory mining tasks. For example, clustering of human mobility patterns can be used to study social events; frequent pattern mining on vehicle trajectories can help identify bottlenecks that lead to traffic jams; while outlier detection may reveal instances of fraudulent taxi driving activities. A more

[2]http://www.noaa.gov

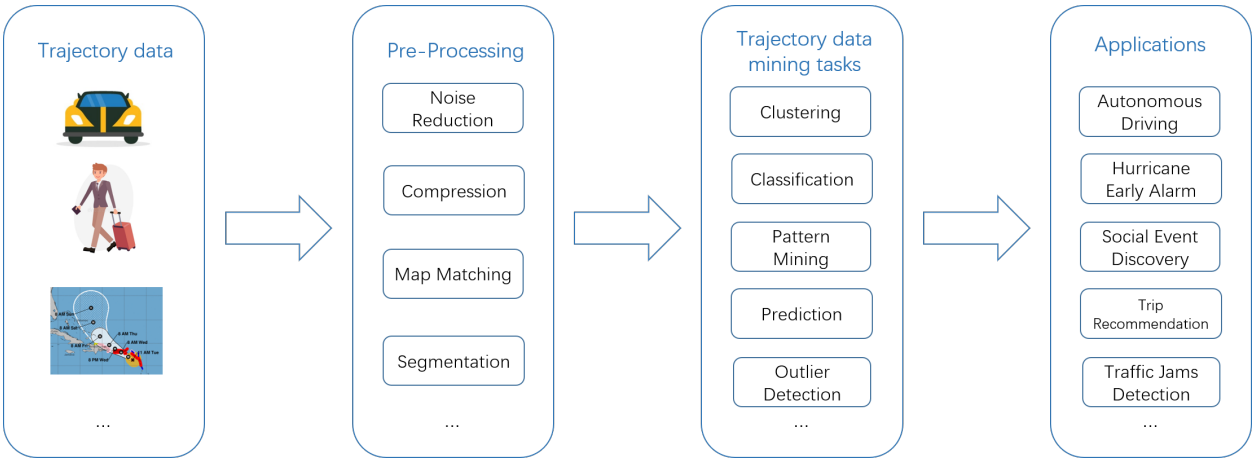


Figure 1.2: General framework of trajectory data mining.

comprehensive review on the different trajectory mining tasks and their applications is given in Chapter 2.

1.3 Application to Hurricane Prediction

Hurricanes are one of the most powerful storms on Earth that have the potential to cause devastating losses and destruction along their paths. Given their severe impact, accurate long-term forecasting of hurricane trajectory is critical to give ample time for emergency response teams to take appropriate actions that will minimize property damages and loss of human lives. In addition to its trajectory, hurricane intensity is another important aspect that must be accurately predicted since it is directly related to the potential destructive power of a hurricane. According to the National Hurricane Center (NHC), high-intensity hurricanes usually cause huge damages. To support emergency preparedness efforts, accurate long-term forecasting of the location and intensity of an impending hurricane is of upmost importance.

There has been growing interest in recent years to apply machine learning methods to the hurricane trajectory forecasting problem [59, 69, 3]. Lee and Liu [59] presented a Recurrent Neural Network (RNN) based approach to predict the hurricane trajectory paths while Kordmahalleh et al. [69] employed a sparse RNN with flexible topology for hurricane trajectory

prediction. Sheila et al. [3] used LSTM and Grid model to forecast the hurricane trajectory. However, there are several limitations to these machine learning approaches. First, they are mostly based on auto-regressive or recurrent neural network models, using only historical observation data. Due to the inherent error propagation problem [16] in such models, they are mostly suitable for short-range predictions. Second, due to the chaotic nature of the weather system and the varying conditions in the atmosphere and ocean temperature, the historical data alone may not be enough to train a reliable long-range forecasting model. Third, the previous methods are mostly designed for batch learning. Thus, they require the model to be re-trained from scratch whenever new observations become available. An online learning method is more appealing as it allows the model to be efficiently updated to fit the new observations while adapting to the concept drifts present in the data.

1.4 Research Challenges

This thesis focuses on the development of novel algorithms to address some of the fundamental challenges in predictive modeling of trajectory data. As proof of concept, the algorithms developed in this research will be applied to the hurricane prediction problem. Even though numerous trajectory mining algorithms have been developed, there are still notable challenges that have not been sufficiently addressed as listed below.

1. Trajectory data collection is often a continuous process, with new path and state information collected over time. However, due to the dynamically changing and non-stationary environment often encountered in many real-world applications, the presence of concept drift will degrade the performance of many existing trajectory prediction models. Therefore, a key challenge is to efficiently develop models that will adapt to changes in the data distribution by fitting the new observation data available.

2. Forecasting the future trajectory path of a moving object is inherently a multi-lead time prediction problem. Thus, the models trained for predicting the future locations or states of a moving object must take into account the inherent temporal autocorrelation

of the multi-lead time prediction task. However, current multi-lead time prediction models are susceptible to error accumulation problem, where the prediction error at a given time step may be propagated to its future time steps.

3. Missing values in the trajectory data can hamper the performance of trajectory prediction models. For example, in hurricane prediction, outputs from an ensemble of physics-based models can be used as input features for building a hurricane prediction model. However, structured missing patterns may exist in the data as the outputs from some physical models in the ensemble can be generated at different time intervals or are available only for certain hurricanes. Handling missing values in the trajectory data is a challenge that has yet to be sufficiently addressed.

4. The state of a moving object can be represented as both qualitative and qualitative attributes. For example, the intensity of a hurricane can be measured in both ratio and ordinal scales. Designing an algorithm that predicts the state of the moving object in both ratio and ordinal scales is a challenge.

5. In practice, some prediction tasks are more important than others. For example, accurate prediction of high-intensity hurricanes or hurricanes that will make landfall in densely populated areas are essential to minimize casualties and property damages. Since such trajectories tend to occur less frequently, this may lead to a class imbalance (or distribution skew) problem. Therefore, designing prediction models that consider the trade-off between prediction accuracy for different classes is another challenge that needs to be addressed.

6. Since the location and state of the moving object are often highly related, it is natural to consider learning their prediction models together. Designing a predictive modeling algorithm that can simultaneously predict the trajectory and qualitative state of a moving object is another challenge that has not been addressed in the literature.

7. Since the trajectory of a moving object is typically governed by nonlinear dynamical processes, developing models that account for nonlinear relationships in the data is another major research challenge.

1.5 Thesis Statement

Online learning is an ideal approach for modeling trajectory data in non-stationary environments, since its model can be efficiently updated to fit the new observations while adapting to concept drifts present in the data. In this research, I will develop a family of online learning algorithms for trajectory data that overcome the research challenges described in the previous section. Specifically, my thesis aims to address the following research questions.

RQ1: How to develop an online learning algorithm to generate multi-lead time forecasts for trajectory prediction task?

RQ2: How to extend the online multi-lead time forecasting algorithm to state prediction that consider both ordinal and ratio data types?

RQ3: How to adapt the online learning algorithm to make it pay more attentions on a subset of the trajectory data?

RQ4: How to combine the online learning algorithms for location and state predictions into a unified learning framework?

RQ5: How to develop an online deep learning model for trajectory prediction to handle nonlinearities in the data?

1.6 Thesis Contributions

In this thesis, a family of online learning algorithms were developed to address various challenges in trajectory and state prediction tasks. The algorithms were applied to hurricane

prediction task as proof of concept.

First, I developed a framework called OMuLeT for trajectory prediction by casting the task as a multi-lead time location prediction problem. OMuLeT employs an online learning with restart strategy to incrementally update the model parameters as new observation data become available. It can also handle the varying feature length problem due to missing values using a re-normalization strategy. OMuLeT was applied to predict the trajectories of hurricanes with a lead time ranging from 6 to 48 hours. Experimental results using the Atlantic and Eastern Pacific hurricane data showed that OMuLeT significantly outperforms various baseline methods, including the official forecasts produced by the U.S. National Hurricane Center (NHC).

Second, I extended the OMuLeT framework to predict the state of the moving object, where the state variable has an ordinal data type. Specifically, I proposed an online learning framework called OOR, which employs an ordinal loss function to predict the ordinal-valued target variable. The framework was subsequently be extended to OOQR to accommodate a quantile loss function in order to improve its prediction accuracy for the high/low ordinal values. The OOR/OOQR frameworks were applied to the hurricane dataset to predict the hurricane categories with lead times from 6 to 48 hours. Experimental results showed that OOR/OOQR outperformed various state-of-the-art online learning methods and can generate predictions close to the NHC official forecasts. The OOQR framework can further improve its accuracy in predicting high category hurricanes. Furthermore, using the ϵ insensitivity loss function, I also developed the OOR-ϵ and OOQR-ϵ frameworks to generate real-valued predictions of the hurricane intensities besides the category information. I've shown that leveraging real-valued intensity information with constraints on the ordinal categories into the learning formulation can further improve the accuracy of hurricane intensity prediction.

Third, I investigated the feasibility of learning a joint model for predicting the trajectory and state of a moving object simultaneously. A framework called JOHAN is proposed that utilizes quantile loss functions for both the location and state prediction tasks. The hyper-

parameters of the quantile loss functions were updated jointly in an online learning fashion. JOHAN was applied to the hurricane dataset with the goal of improving the performance of hurricane intensity prediction, particularly for high category hurricanes that are near to the coastal land. Experimental results demonstrate that JOHAN can further improve the hurricane intensity predictions by incorporating the location information. The results also verified that JOHAN has better performance in terms of identifying high category hurricanes with the potential to strike the land.

Finally, I proposed an LSTM-based approach called DTP for multi-lead time location prediction task. While existing RNN based approaches for trajectory prediction usually learn from historical observations only, they are mostly suitable for short-range predictions due to the inherent error propagation problem [16]. DTP aims to produce accurate long-range predictions by utilizing the multi-lead time location predictions generated by an ensemble of prediction models. A TDM (Temporal Decay Memory) was designed to handle the missing value problem in the data. The DTP framework was extended to an online learning approach known as ODTP to handle the concept drift issue present in trajectory modeling tasks. The proposed frameworks were applied to the hurricane dataset to generate predictions with a lead time up to 48 hours. Experimental results showed that ODTP can achieve better performance than DTP, and generally outperforms other linear baseline approaches.

1.7 Publications

The materials in this thesis are adapted from the following publications:

- Ding Wang, Boyang Liu, Pang-Ning Tan and Lifeng Luo, OMuLeT: Online Multi-Lead Time Location Prediction for Hurricane Trajectory Forecasting, *Proceedings of 34th AAAI Conference on Artificial Intelligence*, 2020 (Chapter 3)

- Ding Wang, Boyang Liu, Pang-Ning Tan, Online Learning Algorithm for Hurricane Intensity Prediction, *FEED20: Fragile Earth: Data Science for a Sustainable Planet*, 2020 (Chapter 4)

1.8 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2: Literature Review In this chapter, I reviewed the trajectory data mining algorithms and discussed their applications. Similar to the data mining algorithms, the trajectory data mining algorithms can be classified into several major categories, such as clustering, classification, pattern mining, prediction and outlier detection.

Chapter 3: Online Multi-Lead Time Trajectory Location Prediction In this chapter, I presented an online learning framework called `OMuLeT`, which was developed for online multi-lead time location prediction task. `OMuLeT` was applied to real-world hurricane dataset to generate hurricane trajectory forecasting in multi-lead times.

Chapter 4: Online Multi-Lead Time Trajectory State Prediction with Ordinal Data In this chapter, I presented a novel algorithm `OOR`, which is an extension of `OMuLeT` to handle ordinal data type for the trajectory state prediction. In addition, the framework can be further extended to `OOQR`, which accommodate a quantile loss function to improve its accuracy for high/low ordinal data. Both algorithm was applied to hurricane dataset to predict the hurricane intensity category, which is one of the most important state data for hurricanes. Furthermore, using ϵ insensitivity loss, `OOR-`ϵ and `OOQR-`ϵ frameworks can be developed to generate real-valued predictions of the hurricane intensities.

Chapter 5: Online Joint Prediction of Trajectory Location and State In this chapter, I investigated the advantages of learning the trajectory location and state predictions jointly and present a novel approach called `JOHAN`. It was applied to hurricane dataset with the goal of improving the model performance of the near land high category hurricanes.

Chapter 6: LSTM for Trajectory Location Prediction In this chapter, I proposed a LSTM based approach `DTP` for multi-lead time location prediction task. This framework was further extended to `ODTP`, which using online learning to address the concept drift issue in trajectory prediction tasks. Both frameworks were applied to hurricane dataset to generate hurricane trajectory forecasting in multi-lead times.

Chapter 7: Conclusion and Future Works In this chapter, I summarized the whole thesis and gave a comprehensive conclusion. I also discussed how to further expand the current research.

CHAPTER 2

LITERATURE REVIEW

Data mining is the process of extracting useful information from large datasets [31, 79]. Since its inception, data mining has been successfully applied to various domains, including trajectory data. The goal of trajectory data mining is two-fold: to build accurate prediction models and to discover interesting patterns from the trajectory data. The general framework for trajectory data mining is shown in Figure 2.1. This chapter reviews previous work related to trajectory data mining.

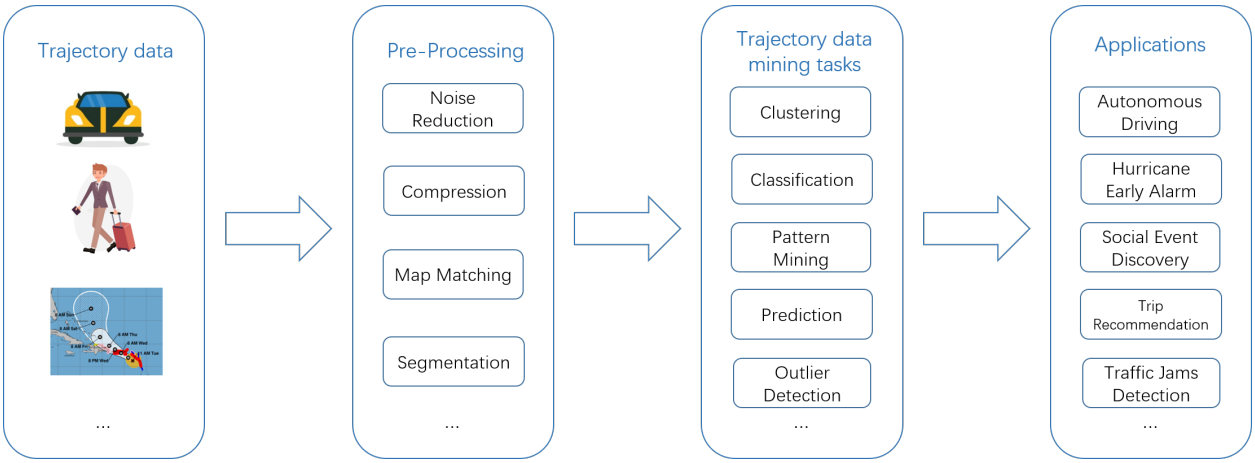


Figure 2.1: General framework of trajectory data mining.

2.1 Trajectory Data Preprocessing

The raw trajectory data often contain noise, missing values, and other imperfections. Thus, a preprocessing step is needed to improve the quality of the trajectory data for subsequent mining tasks. Some downstream tasks may also require additional preprocessing steps such as trajectory segmentation and map matching. In this section, I present some common approaches for preprocessing the trajectory data.

2.1.1 Data Cleaning

Raw trajectory data, such as the coordinates recorded by GPS devices, tend to be noisy. The error in the recorded positions can be as large as hundreds of meters, which may affect the results of the trajectory analysis. Data cleaning aims to filter noise from the trajectory data. There are numerous approaches available to clean the trajectory data. In order to improve the positioning accuracy of GPS data, a variety of filtering techniques can be applied, such as mean and median filters, Kalman and particle filters [67, 60]. For vehicle GPS data, a high density of GPS points indicates a high probability of being on the road, while a low density indicates that the vehicle has deviated far from the road. In this case, low density points are not as important and can be treated as outliers. This suggests the possibility of applying outlier detection algorithms to clean up the GPS dataset [89, 90, 91, 96, 98]. In additions, Yang et al. proposed a GPS data cleaning method that considers movement consistency [85].

2.1.2 Trajectory Compression

Trajectory datasets require considerable data storage as they contain the paths for a large number of objects or trajectories with high temporal resolutions. Massive computing power is also needed to process the data in subsequent mining tasks. Trajectory compression helps to reduce the size of the trajectory data while still accurately retains most of the information in the traces. The compression approach requires balancing the trade-off between maximizing compression ratio and minimizing errors. There are two categories of trajectory compression methods—offline compression [26, 47] and online compression [51, 66]. Offline compression aims to generate lower resolution trajectories with fewer number of sampled points from a trajectory dataset. Since many applications require trajectory compression in a timely fashion, online compression was employed to determine if a newly collected position should be retained in the trajectory dataset.

2.1.3 Map Matching

Additional side information is often needed to improve the performance of some trajectory mining tasks. For example, the modeling of vehicle trajectories can be improved by using information about the road network topology. Map matching converts trajectory sample points from a sequence of coordinates to a sequence of road segments. Different map matching algorithms have been developed, such as topological algorithms [42, 86] and probabilistic algorithms [71, 75, 74]. Topological algorithms consider the shape and connection of road network and map the GPS position sequence to the road network with lowest distance measure. Probabilistic algorithms consider multiple paths on road network and calculate the probabilities from the trajectory sample points. Probabilistic algorithms can handle noisy or low-sampling trajectory datasets.

2.1.4 Trajectory Segmentation

In many cases, we need to divide the trajectory into segments for further processing. Instead of using the entire trajectory, sub-trajectories enable us to mine richer information. In addition, processing trajectory segments can be less time consuming. A trajectory can be segmented based on its properties. There are three types of trajectory segmentation methods. The first type of the methods is based on time interval. If the time interval between two data points is larger than a threshold, the trajectory is separated into two parts. The second type of the methods is based on the shape of a trajectory [56]. If the movement direction of the trajectory changes beyond a certain threshold, the trajectory will split at that point. The third type of the methods is based on its semantic meaning [57, 92, 93]. For example, Yuan et al. [92] estimated the travel speed of a taxi based on the sub-trajectories of the taxi when it is driving. The stop points, which correspond to parked taxis, are used to split a trajectory into a set of trajectory segments.

2.2 Trajectory Data Mining Algorithms

Trajectory data mining algorithms are designed to discover useful knowledge from a trajectory dataset. Many of these algorithms are extensions of traditional data mining algorithms. Similar to traditional data mining, trajectory data mining approaches can be divided into several categories, such as trajectory clustering, trajectory classification, trajectory pattern mining and trajectory prediction. In this section, I will review the various trajectory data mining algorithms.

2.2.1 Trajectory Clustering

Clustering is the task of grouping together related objects such that objects in the same group are more similar to each other than to those in other groups. Clustering can be applied to find representative trajectories or movement patterns shared by different moving objects. Specifically, trajectories that share similar movement characteristics will be grouped together into the same cluster. A typical clustering algorithm requires specifying a similarity measure based on features extracted from the given objects. This can be challenging as the trajectories can have different lengths, locations, etc. Clustering can be performed either on the whole trajectory or on segments of the trajectory.

2.2.1.1 Clustering Trajectories

Many trajectory clustering algorithms are extensions of traditional clustering algorithms using similarity measures specifically defined for the trajectory data. For example, Birant and Kut [7] presented an extension of DBSCAN [30], which is a density-based clustering algorithm, known as ST-DBSCAN, to cluster spatio-temporal data. Gaffney and Smyth [33] introduced a probabilistic mixture regression based approach to cluster trajectories. Furthermore, they applied their method to cluster cyclone trajectories based on their overall directions [34].

2.2.1.2 Clustering Trajectory Segments or Points

Some trajectory clustering tasks requires clustering segments of a single trajectory instead of clustering multiple trajectories. For example, Lee et al. [56] designed a partition-and-group framework for clustering trajectory segments and discovering common sub-trajectories. Palma [72] introduced a spatio-temporal clustering approach to identify stops in the trajectory. According to their definition, a trajectory is composed of a set of stops and moves, with the stop being the most important part of the trajectory.

2.2.2 Trajectory Classification

Classification is the task of assigning objects into a set of pre-defined categories (labels or classes). Classification is a supervised learning task that requires a set of annotated labeled examples to be available for training a classification model. Traditional classification algorithms can be applied to the trajectory classification task by first extracting relevant features from the trajectories before training a classification algorithm on the extracted features. For example, Zheng et al. [97] partitioned the trajectory data into segments and extracted features such as direction change rate, velocity change rate, and stop rate from the segments. These features are then used to construct a decision tree model for classifying trajectories into their different transportation modes. Bolbol et al. [10] employed a support vector machine approach to classify GPS trajectory data into different transportation modes. They first identified the best discriminative features of the trajectory segments before training an SVM classifier to determine the transportation modes.

There are some common steps in trajectory clustering and classification. Both need to extract features from the trajectory data and then apply clustering or classification methods on the extracted features. In some cases, trajectory clustering can be performed as a prior step to trajectory classification. For example, Lee et al. [58] proposed a SVM based classification framework that utilizes trajectory segmentation and clustering to extract regions and sub-trajectory features for trajectory classification purposes.

2.2.3 Trajectory Pattern Mining

Trajectory pattern mining discovers frequently occurring movement patterns in a single trajectory or a group of trajectories. Trajectory pattern mining helps to better understand the characteristics of moving objects. There are different types of trajectory pattern mining tasks. The tasks can be mainly grouped into 3 categories, including periodic pattern mining, sequential pattern mining and group pattern mining.

2.2.3.1 Periodic Pattern Mining

Periodic pattern mining is applicable to trajectory data with periodic activity patterns, such as, the annual migration patterns of animals or the daily routes taken by a bus. Mining periodic patterns from the trajectory data is a challenge due to the complex periodic behaviors including multiple interleaving periods, noise, and outliers. Cao et al. [14] proposed an algorithm to discover frequent regions and iteratively combine them to obtain the complete periodic patterns. This approach is applied to spatio-temporal sequences, though it assumes that the periodicity is known by the user. Li et al. [63] suggested an alternative approach that can detect the periods in advance. First, they detected multiple periods in the movement using a method that combines Fourier transform with temporal auto-correlation. They then designed a probabilistic model to reveal the periodic behavior from movement sequences using hierarchical clustering.

2.2.3.2 Sequential Pattern Mining

Sequential pattern mining is the task of finding common subsequences shared by multiple trajectories. The common subsequence is a subset of locations that appears in multiple trajectories. The discovered sequential patterns enable us to better understand the relationship between different trajectories. An example of a sequential pattern is shown in Figure 2.2. The trajectories A, B, and C share a common subsequence D containing the path

$p_2 \xrightarrow{2\text{ hour}} p_4 \xrightarrow{1\text{ hour}} p_5$.

$$
\begin{aligned}
A &: p_1 \xrightarrow{1\text{ hour}} p_2 \xrightarrow{1.2\text{ hour}} p_3 \xrightarrow{0.8\text{ hour}} p_4 \xrightarrow{1\text{ hour}} p_5 \\
B &: p_2 \xrightarrow{1.2\text{ hour}} p_6 \xrightarrow{1\text{ hour}} p_4 \xrightarrow{1\text{ hour}} p_5 \xrightarrow{1\text{ hour}} p_7 \\
C &: p_1 \xrightarrow{1.2\text{ hour}} p_2 \xrightarrow{2.2\text{ hour}} p_4 \xrightarrow{0.5\text{ hour}} p_8 \xrightarrow{0.6\text{ hour}} p_5 \\
D &: p_2 \xrightarrow{2\text{ hour}} p_4 \xrightarrow{1\text{ hour}} p_5
\end{aligned}
$$

Figure 2.2: The trajectories A, B, C shared a common sequence as trajectory D with similar time interval.

The sequential patterns can be uncovered according to their spatial, temporal, or spatio-temporal properties, as shown in Figure 2.3. Some trajectory mining tasks consider only the spatial property, which is the sequence of locations. Cao et al. [13] proposed an algorithm to find sequential patterns. First, they transformed the original sequence into a list of sequence segments before detecting the frequent regions. The patterns are found by employing a substring tree structure approach that extends the Apriori technique. Giannotti et al. [37] introduced the notion of a trajectory pattern (T-Pattern) and developed several approaches to extract them from a given trajectory data.

$$
\begin{aligned}
A &: p_1 \longrightarrow p_2 \longrightarrow p_3 \longrightarrow p_4 \longrightarrow p_5 \\
B &: p \xrightarrow{1\text{ hour}} p \xrightarrow{2\text{ hour}} p \xrightarrow{2\text{ hour}} p \xrightarrow{1\text{ hour}} p \\
C &: p_1 \xrightarrow{1\text{ hour}} p_2 \xrightarrow{2\text{ hour}} p_3 \xrightarrow{2\text{ hour}} p_4 \xrightarrow{1\text{ hour}} p_5
\end{aligned}
$$

Figure 2.3: Trajectory A considers only spatial property; trajectory B considers only temporal property; while trajectory C considers both spatio-temporal properties.

2.2.3.3 Group Pattern Mining

Group pattern mining extracts the movement patterns for a group of objects moving together. There are different kinds of group patterns, such as flocks [45, 44, 6], convoys [49, 50] and

swarms [62]. These group patterns are the major categories of group patterns with examples shown in Figure 2.4.

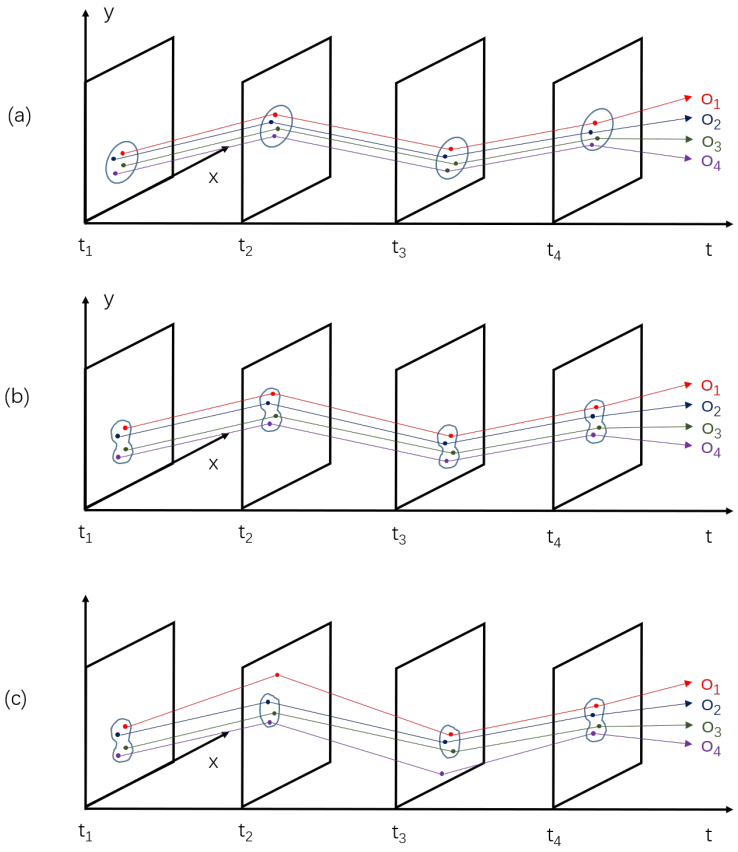


Figure 2.4: Example of a sequential group patterns: (a)flock, (b)convoy, (c)swarm.

A flock [45] is a group of objects that travel together for at least k consecutive time stamps and stay within a disc of radius r. One major concern is that the predefined disc of radius r may not represent the actual groups of moving objects in terms of their shapes and sizes. To address this issue, Jeung et al. [49, 50] defined a convoy pattern as a group of objects that travel together for at least k consecutive time stamps and stay within a shape generated by density-based clustering. The convoy pattern relaxes the disc shape assumption required by flock to any shapes formed by the moving objects. However, both flock and convoy patterns require the group of objects to move together for at least k consecutive time steps. Later, Li et al. [62] proposed an new group pattern, called swarm, which is defined as a group of objects that travel together for at least k possibly non-consecutive time stamps.

2.2.4 Trajectory Prediction

The goal of trajectory prediction is to forecast the future location of a moving object based on its previous trajectory paths. Asahara et al. [4] designed a method for predicting pedestrian movement using a mixed Markov-chain model (MMM). Specifically, they modeled the pedestrian's personality as an unobservable parameter and utilized the pedestrian's previous status. Mathew et al. [64] presented a method for predicting human mobility based on Hidden Markov Models (HMMs). They first clustered the location histories according to their characteristics and then trained an HMM for each cluster. Besides Markov models, the future trajectory path can be predicted by using trajectory pattern mining. For example, Monreale et al. [68] presented a method, called WhereNext, to predict the next location of a moving object with a decision tree built from the extracted trajectory patterns. Ying et al. [87] modeled user's movement behavior using historical trajectory patterns and estimated the probability of user's next location.

More recently, there has been considerable interests in applying deep learning approaches to the trajectory prediction task. Alahi et al. proposed a Long Short-Term Memory (LSTM) based approach to predict future trajectories of humans in crowded spaces. They built multiple LSTMs for pedestrians and a pooling layer to share the information between LSTMs. Kim et al. [52] designed an efficient LSTM based framework to predict vehicle trajectories. They employed LSTM to analyze the vehicle temporal behavior and to predict the future locations of the surrounding vehicles on a grid map. Lee and Liu [59] presented a RNN based approach to predict the feature location of hurricanes. Kordmahalleh et al. [69] used sparse RNN with flexible topology for hurricane trajectory predictions. To forecast the hurricane trajectory, they found the most similar hurricanes to the target hurricane and trained their RNN model with a genetic algorithm (GA). However, they were only interested in short-term forecasts (up to 6 hours).Sheila et al. [3] used Long Short-Term Memory(LSTM) and Grid model to forecast hurricane trajectory. They also focused on short-term forecasting up to 6 hours of lead time.

2.2.5 Trajectory Outlier Detection

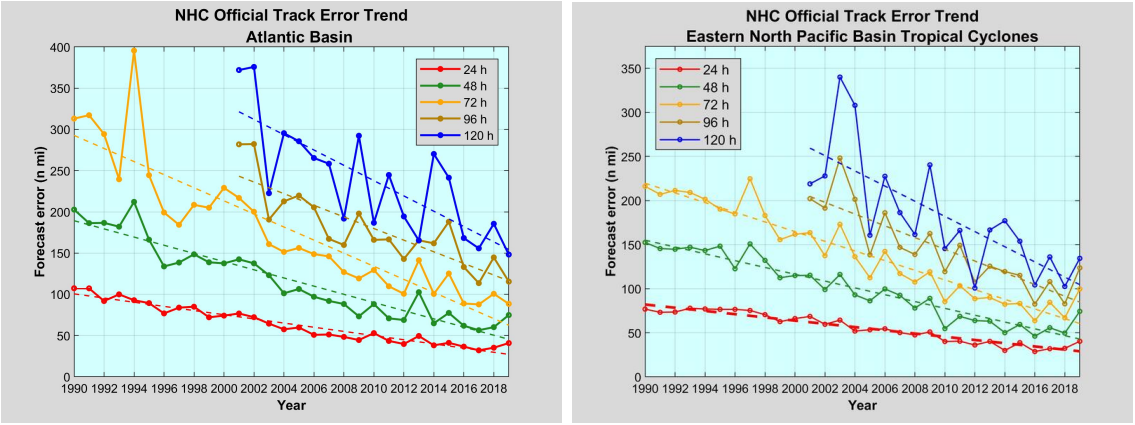
Outlier detection aims to find unusual trajectories or sub-trajectories that do not have similar patterns in the given trajectory dataset. Trajectory clustering or classification algorithms can be used for trajectory outlier detection. For example, Lee et al. [57] extended the partition-and-group framework they had developed in [56] for trajectory clustering to a trajectory outlier detection task. Li et al. [61] designed a rule-based classifier to detect abnormal moving objects. They extracted features from the trajectories to form a hierarchical feature space and trained a rule-based classifier to detect the outliers.

Trajectory outlier detection can be applied either to the whole trajectory [94] or sub-trajectories [57, 88]. Zhang et al. [94] presented an algorithm for discovering anomalous driving patterns from GPS traces of taxi rides. Yuan et al. [88] extracted features from trajectory segments and employed a distance measure to find outliers.

2.3 Trajectory Data Mining Applications

There are many important applications of trajectory data mining. For example, the GPS trajectories of taxis can be used to study city traffic dynamics. The study of the user's travel trajectory can be used to generate travel recommendations. In this thesis, I investigated the hurricane trajectory prediction task, which is one of the most important trajectory prediction tasks. I developed a set of trajectory prediction algorithms and applied them to real-world hurricane data. In this section, the recent applications for hurricane prediction tasks were discussed.

Hurricanes are strong tropical cyclones with wind speed exceeding 75 miles per hour that it can cause severe damages at landfall. Therefore, hurricane trajectory forecasting is crucial, which allows the civilians to have more time to prepare and evacuate. Hurricane trajectory forecasting predicts future positions of the hurricane. The hurricane trajectory forecasting relies on complex physical models, which are complex system contains mathematical formulations of physical processes. There are numbers of different physical models,



(a) (b)

Figure 2.5: Track errors of NHC official forecasts from year 1990 to 2019 [12]. Figure (a) shows the errors at Atlantic basin. Figure (b) shows the errors at Eastern North Pacific basin.

such as AVNO, AVNI, AEMN, AEMI[1]. Scientists are interested in improving the accuracy of trajectory forecasts for extended period of time. For example, forecast for upcoming days. However, it is difficult to improve the performance since these forecast models are sensitive to the complexity and nonlinearity of atmospheric system. NHC published the error trend of their official track forecasting error over past decades [12] as shown in Figure 2.5. With better satellite data and faster computers, the forecast performance has been slowly improved since the past decades.

Numerous data mining and machine learning approaches have been applied to the hurricane prediction problem. For example, Lee and Liu [59] presented a neural network-based approach to predict the features of hurricanes. Kordmahalleh et al. [69] used sparse RNN with flexible topology for hurricane trajectory predictions. To forecast the hurricane trajectory, they found the most similar hurricanes to the target hurricane and trained their RNN model with Genetic Algorithm (GA). Sheila et al. [3] utilized Long Short-Term Memory(LSTM) and Grid model to forecast hurricane trajectory. However, these approaches were mostly limited to short-term forecasts (of 6 hours) only. Accurate prediction of long-term forecasts is needed to give ample time for emergency preparation efforts. The National

[1]https://www.nhc.noaa.gov/modelsummary.shtml

Hurricane Center (NHC) of the National Oceanic and Atmospheric Administration (NOAA) generates official forecasts for impending hurricanes based on various models. These models can be categorized into four categories: dynamical, statistical, statistical-dynamical, and ensemble or consensus models. During the life time of a hurricane, NOAA provides its official forecasts every 6 hours, which include information such as longitude, latitude, maximum wind speed, and the central pressure.

CHAPTER 3

ONLINE MULTI-LEAD TIME TRAJECTORY LOCATION PREDICTION

Trajectory prediction is one of the most important tasks in trajectory data mining. The goal of this task is to infer the future locations of a moving object. Trajectory prediction is therefore equivalent to a multi-lead time location prediction task, which a challenging problem due to the inherent error propagation problem [16]. The presence of concept drift in the data will also result in outdated models, thereby degrading their predictive accuracy over time. In this chapter, I propose to address these challenges by developing an online learning algorithm to generate the anticipated trajectory path of the moving object. As proof of concept, the framework was applied to hurricane trajectory prediction, which is an important application with significant real-world implications.

Hurricanes are one of the most powerful storms on Earth that have the potential to cause devastating losses and destruction along their paths. For example, the Galveston Hurricane of 1900 is considered the deadliest hurricane in United States, responsible for at least 8000 deaths [9]. In 2005, Hurricane Katrina took away more than 1500 lives and caused at least $108 billion of property damages [9]. Given their severe impact, accurate long-range prediction of hurricane tracks is critical to give ample time for emergency response teams to take appropriate actions that will minimize property damages and loss of human lives. Towards this end, dynamical models such as NOAA's Hurricane Weather Research and Forecasting (HWRF) system and U.S. Navy Global Environmental Model (NAVGEM) have been widely used as the primary tool for hurricane forecasting. Although the skills of these models have improved steadily over the years, the forecast errors and variability among the model predictions still increase with lead time, as shown in Fig. 3.1.

Ensemble forecasting seeks to better represent the range of forecast uncertainties by combining outputs generated by multiple dynamical models. Each dynamical model can produce one or more ensemble member outputs by perturbing its initial conditions or model

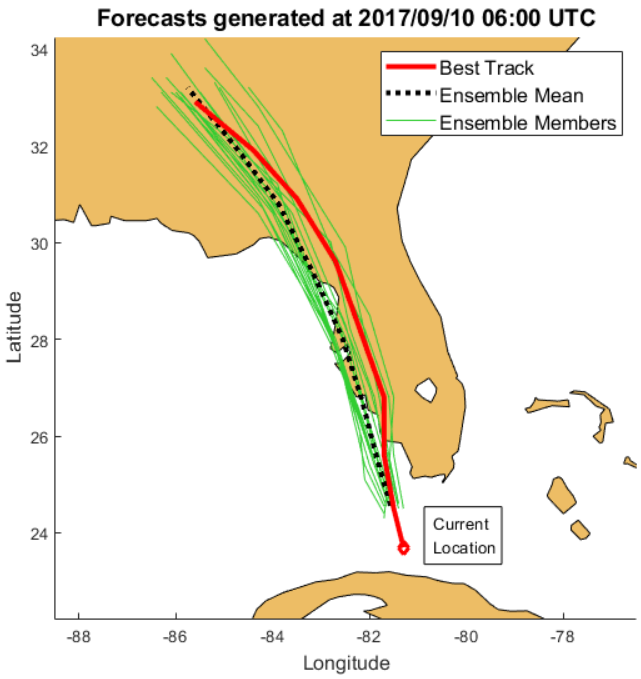


Figure 3.1: Hurricane track for the following 48 hours predicted by an ensemble of dynamical models for Hurricane Irma on September 10, 2017. The green lines represent the various forecasts produced by the ensemble members whereas the black dash line corresponds to the ensemble mean. The red line corresponds to the best track according to the National Hurricane Center.

parameters. The ensemble mean or median are commonly used as the deterministic forecast from the ensemble. These estimates assume that every member is equally skillful, and thus, their predictions should be weighted equally. Such an assumption may not be realistic due to the inherent differences in the way the ensemble member outputs are generated. Thus in an operational forecast environment when such ensemble forecasts are issued on a regular basis, the weight of each member must be established based on their accuracy in predicting the tracks. However, determining the appropriate weights is not a trivial task as the skills of the models may vary overtime. To overcome this challenge, the primary goal of this chapter is to develop an online trajectory forecasting framework that can dynamically update the weights of the ensemble members based on their past and recent performance when verified against observations.

			Ensemble member forecasts at different lead times, τ					
			$\tau = 1$ (24 hrs)			$\tau = 2$ (48 hrs)		
Hurricane h_i	Time t	NHC Best track, $\mathbf{y}^{i,t}$	AEMI $\mathbf{x}_1^{i,t,1}$	AEMN $\mathbf{x}_2^{i,t,1}$	CLP5 $\mathbf{x}_3^{i,t,1}$	AEMI $\mathbf{x}_1^{i,t,1}$	AEMN $\mathbf{x}_2^{i,t,2}$	CLP5 $\mathbf{x}_3^{i,t,2}$
SANDY	1	[12.7; -78.7]	[14.6; -77.8]	N/A	[13.4; -80.7]	[18.4; -76.4]	N/A	[14.2; -82.3]
	2	[12.9; -78.1]	[15.7; -77.7]	N/A	[14.1; -79.2]	[19.8; -76.8]	N/A	[15.6; -79.9]
	3	[14.0; -77.6]	[17.8; -76.9]	N/A	[16.0; -77.5]	[22.7; -76.5]	N/A	[18.0; -77.9]
IRMA	1	[16.4; -32.5]	[18.5; -35.3]	[18.8; -35.3]	N/A	[19.1; -39.7]	[19.2; -40.0]	N/A
	2	[17.1; -34.2]	[18.4; -38.4]	N/A	N/A	[18.2; -42.9]	N/A	N/A
	3	[17.9; -36.1]	[18.4; -40.2]	[18.7; -40.5]	N/A	[17.5; -44.4]	[17.6; -45.0]	N/A

Table 3.1: Example of NHC best track hurricane trajectory data along with the forecasts generated by an ensemble of dynamical models such as AEMI, AEMN, and CLP5 for Hurricanes Sandy and Irma at different lead times. N/A denotes missing values.

In the United States, the National Hurricane Center (NHC) is responsible for monitoring and providing official forecasts of the trajectory and intensity of hurricanes in the Atlantic and Eastern Pacific to the public. With a set of dynamical model forecasts as guidance, the official NHC forecasts are produced based on the experience and judgment of the forecasters. A secondary goal of this chapter is to investigate the feasibility of using an online learning approach to generate forecasts that are equally or more skillful than the official forecasts reported by NHC.

There has been growing research in recent years to apply data mining and machine learning methods to the hurricane trajectory forecasting problem [59, 69, 3]. However, there are several limitations to these approaches. First, they are mostly based on auto-regressive or recurrent neural network models, using only the historical observation data. Due to the inherent error accumulation problem [16] in such models, they are mostly suitable for short-range predictions and are ineffective for early warning systems. Second, due to the chaotic nature of the weather system and the varying conditions in the atmosphere and ocean temperature, the historical data alone may not be enough to train a reliable long-range forecasting model. By grounding the historical observations with multi-model ensemble forecasts from dynamical models, it may lead to more reliable predictions. Third, the previous methods are mostly designed for batch learning. Thus, they require the model to be re-trained from scratch whenever new observations become available. An online learning method is more

appealing as it allows the model to be efficiently updated to fit the new observations.

Designing an online learning algorithm for hurricane trajectory forecasting is a challenge for several reasons. First, the models trained for predicting the hurricane's location at different lead times must take into account the inherent autocorrelation along the trajectory. Furthermore, they are susceptible to the partially observed data problem described in [84]. For example, if the model is updated every six hours with new observation data and the forecast horizon (i.e., maximum lead time) is 48 hours, it is insufficient to revise only the latest model. Instead, we should also revise the older models for all lead times starting from 48 hours ago up to 6 hours ago. Otherwise, the errors from the older models will continue to propagate into future prediction. Another challenge is that the ensemble members available may vary from one hurricane to another (see Table 3.1). Due to the missing forecasts by some model members, the online algorithm must adaptively learn the weights in spite of the varying feature lengths. To address these challenges, we propose a novel framework called `OMuLeT` (**O**nline **Mu**lti-**Le**ad **T**ime Forecasting), which employs an online learning with restart strategy to incrementally update the weights of the ensemble members. OMuLeT can also handle the varying number of ensemble member forecasts by employing a novel weight renormalization scheme. Theoretical proofs are provided to justify the weight renormalization approach. Experimental results using the Atlantic and Eastern Pacific hurricane data showed that `OMuLeT` can improve the 48-hour lead time official forecast of NHC by more than 10%.

3.1 Related Work

Alemany et al. [3] categorized previous methods for hurricane trajectory forecasting into 4 types: (1) dynamical models [81, 22], (2) statistical models [24, 76], (3) statistical-dynamical models [80] and (4) ensemble or consensus models [95]. Dynamical models require powerful computers to solve the physical equations describing changes in the atmospheric system. In contrast, statistical models consider the relationship between the current and historical

trajectories. Statistical-dynamical models are hybrids of both techniques while ensemble models generate forecasts by merging the forecasts from a suite of dynamical and/or statistical models.

In addition to hurricane tracks, the multi-lead time trajectory prediction can be applied to other domains. For example, Asahara et al. [4] designed a method for predicting pedestrian movement using a mixed Markov-chain Model while Mathew et al. [64] presented a method for predicting human mobility based on hidden Markov models (HMMs). Trajectory pattern mining methods have also been developed for track prediction. For example, Monreale et al. [68] presented a method called WhereNext to locate the position of a moving object using a decision tree trained to fit features extracted from trajectory patterns. Ying et al. [87] uses historical trajectory patterns to estimate the probability of a user's next location. More recently, deep learning methods have been developed for trajectory prediction tasks. For example, Alahi et al. [2] proposed a Long Short-Term Memory (LSTM) based approach to predict human future trajectories in crowded spaces while Kim et al. [52] designed an LSTM based framework to predict vehicle trajectories.

There have also been some preliminary work on hurricane trajectory forecasting using deep learning. Lee and Liu [59] presented a Recurrent Neural Network (RNN) based approach to predict the hurricane trajectory paths while Kordmahalleh et al. [69] employed a sparse RNN with flexible topology for hurricane trajectory prediction. Sheila et al. [3] used LSTM and Grid model to forecast the hurricane trajectory. None of these approaches utilize the multi-model ensemble forecast nor were they designed for an online learning setting unlike the approach proposed in this thesis.

3.2 Problem Formulation

We investigate the problem of predicting the trajectory of a moving object based on a set of input features that can be derived either from historical observations or through other means (e.g., forecasts generated from a multi-model ensemble in the case of hurricane

trajectory prediction). For brevity, the rest of the discussion in this chapter is presented in the context of hurricane prediction, although the problem formulation and methodology can be applied to other domains with similar characteristics. Consider a set of N hurricanes, $h_1 \leq h_2 \leq \cdots \leq h_N$, ordered by their start times. For the i-th hurricane, let $\mathbf{y}^{i,t} \in \mathbb{R}^2$ denote its location (latitude and longitude) at time t, where $t \in \{t_{i,1}, \cdots, t_{i,\Gamma_i}\}$ and Γ_i denotes the observed trajectory length for hurricane h_i. Furthermore, at each time t, our goal is to forecast the hurricane's location at a future time step $t+\tau$, where $\tau \in \{1, \cdots, T\}$ is the lead time and T is the forecast horizon.

Let m_i be the number of ensemble member forecasts available for hurricane h_i. The set of ensemble member forecasts available to predict the location of h_i at time $t+\tau$ is represented by the matrix $\mathbf{X}^{i,t,\tau} \in \mathbb{R}^{2\times m_i}$, while its ground truth location is given by $\mathbf{y}^{i,t+\tau} \in \mathbb{R}^2$. The hurricane trajectory data is given by a set of 2-tuples, $\{(\mathbf{X}^{i,t,\tau}, \mathbf{y}^{i,t+\tau})\}$, where the superscript i denotes the hurricane, t is the forecast generation time, and τ is the forecast lead time.

Varying Feature Length: One of the key characteristics of the multi-model ensemble hurricane trajectory data is that its feature length, i.e., number of ensemble member forecasts (m_i) associated with each hurricane, may vary from one hurricane to another . Specifically, although there are numerous ensemble member forecasts generated over the years, each hurricane has forecasts obtained from an average of only 19 ensemble members in our dataset. The unavailable ensemble members would create non-random missing patterns in the data. Imputing the missing values is not a viable solution due to the high missing rate. Instead, we propose an approach that can automatically handle the varying feature length by renormalizing the weights of the ensemble members.

Temporal Inconsistencies: Outputs from the dynamical models have varying degrees of temporal inconsistencies. First, the dynamical models can have different forecast time intervals. Some models generate their forecasts every 6 hours, while others every 12 hours. To address this problem, we perform interpolation to impute the missing values of the 12-hourly forecast intervals to obtain 6-hour forecasts for all ensemble members. Second, the

forecast duration often varies among the ensemble members. For example, some members generate their forecasts for only a few days, while others may extend longer than a week. In addition, their forecast horizon are also different. For example, some models produce forecasts with a maximum lead time of 24 hours, while others may generate forecasts for a lead time up to 120 hours. A novel weight renormalization approach is proposed in this chapter to address such temporal discrepancies.

Partially Observed Labeled Data: Another challenge is that ground truth values for the multi-lead time forecasts are only partially observed. This problem is illustrated in Figure 3.2. Let $\mathbf{X}^{i,t-1,1}$ be the set of ensemble member forecasts generated for hurricane h_i at time $t-1$ for the lead time $\tau = 1$. If the current time is t, then the ground truth value $\mathbf{y}^{i,t}$ will be available to verify the accuracy of the forecasts in $\mathbf{X}^{i,t-1,1}$. However, for the longer-range forecasts, $\mathbf{X}^{i,t-1,2}$, $\mathbf{X}^{i,t-1,3}$, $\cdots$, $\mathbf{X}^{i,t-1,T}$, the ground truth values have not been observed. In fact, the ground truth values are only available for any previous forecast $\mathbf{X}^{i,t-k,\tau}$ for which $\tau - k \leq 0$. This corresponds to the red rectangles shown in Figure 3.2. As time progresses to $t+1$, the true value for $\mathbf{y}^{i,t+1}$ will be known. Conventional online learning algorithms use the new observation $\mathbf{y}^{i,t+1}$ to update their latest models only. This is insufficient for multi-lead time forecasting as the new observation data may trigger a cascading effect since some of the earlier models from which the current models are obtained are also outdated. The models for various lead times generated at time t must be rolled-back all the way to time $t-T+1$ and updated again with the new observation data to alleviate the error propagation problem. This strategy is known as online learning with restart [84].

3.3 Methodology

We consider an online model of the form $f(\mathbf{X}^{i,t,\tau}) = \mathbf{X}^{i,t,\tau}\mathbf{w}^{i,t,\tau}$ for predicting the location of hurricane h_i at time $t+\tau$, where $\mathbf{w}^{i,t,\tau} \in \mathbb{R}^{m_i}$ is the estimated weight vector associated with the m_i ensemble member forecasts. Conventional online learning algorithms [21, 84] typically assume that the feature matrix $\mathbf{X}^{i,t,\tau}$ is complete, i.e., has no missing values, or

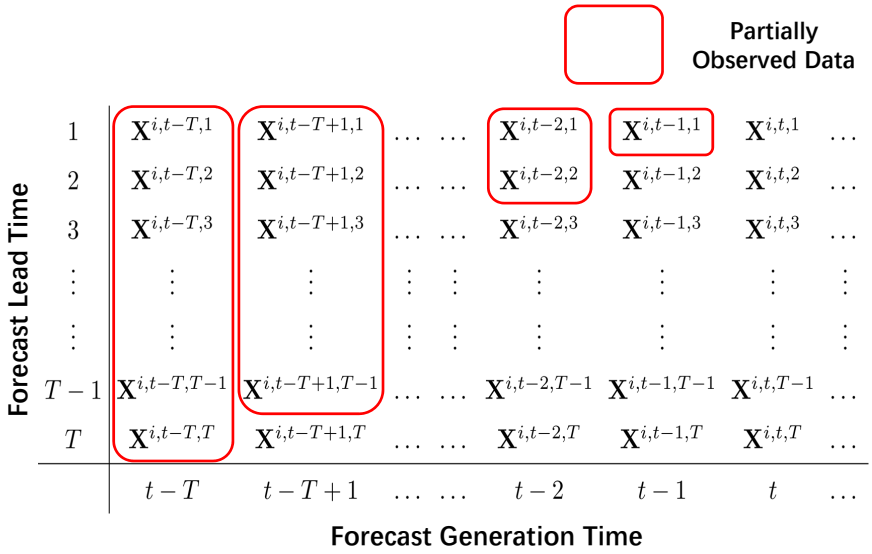


Figure 3.2: An illustration of the partially observed labeled data problem. The red rectangles denote the set of model forecasts for which ground truth are available at time t.

contains missing values that have been imputed. However, due to the varying feature length problem described in the previous section, some ensemble member forecasts may not be available for a given hurricane h_i. Below, we describe our proposed approach to address this problem.

3.3.1 Weight Renormalization

This section presents the weight renormalization approach employed by our online learning framework to overcome the varying feature length problem. Let $\boldsymbol{\mu} = \{\mu_1, \mu_2, \cdots, \mu_m\}$ be the set of all ensemble members and $\mathbf{M}_i \subseteq \boldsymbol{\mu}$ be the subset of members whose forecasts are available for hurricane h_i. Since $|\mathbf{M}_i| \ll m$, imputing the missing ensemble member forecasts is not an effective approach given the large amount of missing values present in the data. Instead, we present an online learning approach that uses only the ensemble member forecasts available for the given hurricane ($\mathbf{M}_i$) and update their weights accordingly when new observation data becomes available at each time step. Specifically, we assume the forecasts from each ensemble member follow a Gaussian distribution centered at the true location. To illustrate this, Figure 3.3 shows a normalized histogram of the trajectory forecast

errors for 5 dynamical models when applied to more than 200 hurricanes in our dataset. Observe that the forecast error distribution indeed resembles that of a Gaussian distribution. We also assume that the weights of the ensemble members form an m-dimensional simplex, i.e.:

$$\Delta_m = \{w_1^i, w_2^i, \cdots, w_m^i \mid \forall i : \sum_j w_j^i = 1, w_j^i \geq 0\}.$$

Given a hurricane h_i, our framework performs the following steps to incrementally update the weights:

1. We extract the subvector $\mathbf{w}_0^i \in \mathbb{R}^{m_i}$ from the full vector $\mathbf{w} \in \mathbb{R}^m$, whose elements contain only the weights of the m_i ensemble members in $\mathbf{M}_i$.

2. We normalize $\mathbf{w}_0^i$ to have unit sum as follows:

$$\mathbf{w}_0^i \leftarrow \mathbf{w}_0^i / c, \text{ where } c = |\mathbf{w}_0^i|_1 = \sum_j \mathbf{w}_{0,j}^i$$

3. At each time step $t = \{t_{i,1}, t_{i,2}, \cdots, t_{i,\Gamma_i}\}$:

   a) We use the normalized weights to predict the location of the hurricane at lead time $t+\tau$, i.e., $f(\mathbf{X}^{i,t,\tau}) = \mathbf{X}^{i,t,\tau}\mathbf{w}^{i,t,\tau}$, where $\mathbf{w}^{i,t,\tau}$ is computed from $\mathbf{w}_0^i$ according to Eqn. (3.6).

   b) After observing the ground truth location $\mathbf{y}^{i,t}$, we update $\mathbf{w}^{i,t,\tau}$ using the method described in Section 3.3.3.

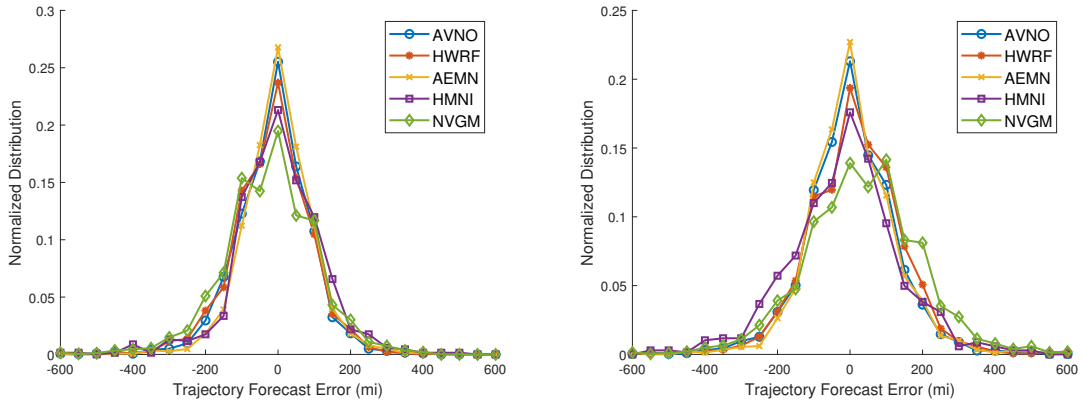
4. After the last update at time $t = \Gamma_i$, the updated weights are renormalized to their original sum:

$$\mathbf{w}_0^i \leftarrow c\mathbf{w}_0^i \tag{3.1}$$

   before being replaced into the full vector $\mathbf{w}$. This ensures that $\mathbf{w}$ remains a simplex after the weight update.

The preceding approach enables our framework to update only the weights of the ensemble members whose forecasts are available for hurricane h_i. The weights need to be renormalized

when replacing them back into the full vector $\mathbf{w}$. Below, we provide a formal justification for the weight renormalization approach. For brevity, we assume the hurricane location is a scalar variable, even though the theorem below can be extended to 2-dimensional location vectors.



(a) Forecast errors along the latitude direction.

(b) Forecast errors along the longitude direction.

Figure 3.3: Normalized distribution of trajectory forecast errors (in 50 miles bin) for 5 different dynamical models along their latitude and longitude directions.

Example 1. Consider a set of forecasts generated from an ensemble of 5 members. Assume their weights is given by the full vector $\mathbf{w} = [0.15, 0.1, 0.4, 0.05, 0.3]$. Suppose we want to update $\mathbf{w}$ based on the observed trajectory of hurricane h_i. Assume the forecast data of h_i are only available for the second and third ensemble members. In the first step, we extract the subvector $\mathbf{w}^i = [0.1, 0.4]$. After normalizing the vector to have unit sum, the weights become $\mathbf{w}^i = [0.2, 0.8]$. Using the forecast data from the two ensemble members and their ground truth values, suppose the weights are updated to $\mathbf{w}^i = [0.4, 0.6]$. We renormalize their values to maintain the original sum of 0.5 before replacing them into the full weight vector. The full vector after the weight update is $\mathbf{w} = [0.15, 0.2, 0.3, 0.05, 0.3]$.

Next, we provide formal proofs to justify the rationale for our proposed weight renormalization approach. For brevity, we assume the hurricane location is a scalar variable instead

of a 2-dimensional vector of latitude and longitude, though the proof below can be extended to d-dimensional location vectors.

Theorem 1. Let y denotes the true hurricane location and $\{X_1, X_2, \cdots, X_M\}$ be M i.i.d. variables, representing the M ensemble member forecasts. Assume each X_j is a random perturbation around y, i.e.:

$$X_j = y + \epsilon(0, \sigma_j^2),$$

where $\epsilon(0, \sigma_j^2)$ is a Gaussian distribution with mean 0 and variance σ_j^2. Then, the best unbiased linear estimator (BLUE) for y is $z = \sum_j w_j X_j$, where $w_j = \frac{1}{\sigma_j^2} \sum_{j=1}^{M} \frac{1}{\sigma_j^2}$.

Proof. We consider linear estimators of the form $z = \sum_j w_j X_j$. Since $\{X_1, X_2, \ldots, X_M\}$ are i.i.d. variables:

$$E(z) = \sum_{j=1}^{M} w_j E(X_j) = \sum_{j=1}^{M} w_j y \tag{3.2}$$

Since z is unbiased, its expected value is equal to $E[y]$. Thus,

$$E(z) = \sum_{j=1}^{M} w_j y = y \Rightarrow \sum_{j=1}^{M} w_j = 1 \tag{3.3}$$

The variance of the linear estimator is

$$Var(z) = \sum_{j=1}^{M} w_j^2 Var(X_j) = \sum_{j=1}^{M} w_j^2 \sigma_j^2 \tag{3.4}$$

To find the $\mathbf{w}$ that minimizes the variance, subject to the constraint in Eq. (3.3), we consider the following Lagrangian function:

$$L = \sum_{j=1}^{M} w_j^2 \sigma_j^2 - \lambda(\sum_{j=1}^{M} w_j - 1)$$

Taking its partial derivative w.r.t w_k and setting it to 0 yields

$$\frac{\partial L}{\partial w_k} = 2 w_k \sigma_j^2 - \lambda = 0 \Rightarrow w_k = \frac{\lambda}{2\sigma_k^2}$$

Following the constraint $\sum_{j=1}^{M} w_j = 1$, we can solve for λ and obtain:

$$w_j = \frac{1}{\sigma_j^2} / \sum_{k=1}^{M} \frac{1}{\sigma_k^2} \tag{3.5}$$

which completes the proof. □

The preceding theorem considers an estimator z computed from M i.i.d. variables. Let $\tilde{z}$ be another estimator computed using K of the i.i.d. variables in $\{X_j\}$. Without loss of generality, we assume the K variables are $X_1, X_2, \cdots, X_K$.

Corollary 1. Let y be the true location of the hurricane and $\tilde{z} = \sum_{j=1}^{K} \tilde{w}_j X_j$ be a linear estimator of y, where each $X_j = y + \epsilon(0, \sigma_j)^2$. Then the best linear unbiased estimator (BLUE) for y using the K i.i.d. variables is $\tilde{z} = \sum_{j=1}^{K} \tilde{w}_j X_j$, where $\tilde{w}_j = c w_j$ and $c = \frac{1}{\sum_{j=1}^{K} w_j}$.

Proof. The BLUE for $\tilde{w}_j$ is similar to Eq. (3.5) except the sum in the denominator goes from 1 to K. Taking the ratio of $\tilde{w}_j$ to w_j:

$$\frac{\tilde{w}_j}{w_j} = \frac{1}{\sum_{j=1}^{K} \frac{1}{\sigma_j^2} / \sum_{l=1}^{M} \frac{1}{\sigma_l^2}} = \frac{1}{\sum_{j=1}^{K} w_j} \equiv c$$

Thus, $\tilde{w}_j = c w_j$, which completes the proof. □

The preceding corollary shows the normalization factor needed to re-scale the weights of a subset of the ensemble members.

3.3.2 Geographic Distance Loss Function

Instead of using a squared ℓ_2 (Euclidean) loss function, our framework considers the squared geographic distance to compute the error in location estimation. Let $\mathbf{z}^{i,t,\tau} = [z_1^{i,t,\tau}, z_2^{i,t,\tau}]$ be the predicted latitude and longitude position of hurricane h_i at time t for the lead time τ and $\mathbf{y}^{i,t+\tau} = [y_1^{i,t+\tau}, y_2^{i,t+\tau}]$ be the corresponding true location. The squared geographic distance between the predicted and true locations, $d[\mathbf{z}^{i,t,\tau}, \mathbf{y}^{i,t+\tau}]^2$, can be estimated as follows:

$$R_e^2 \Big[(z_1^{i,t,\tau} - y_1^{i,t+\tau})^2 + (z_2^{i,t,\tau} - y_2^{i,t+\tau})^2 \cos^2 y_1^{i,t+\tau} \Big],$$

where R_e is the radius of the earth. As R_e is a constant that can be absorbed into the regularizer term of an objective function, we can set $R_e = 1$ to simplify the notation. Furthermore,

by transforming the coordinates of the location to

$$\tilde{\mathbf{y}}^{i,t+\tau} = [y_1^{i,t+\tau}, y_2^{i,t+\tau} \cos y_1^{i,t+\tau}]$$
$$\tilde{\mathbf{z}}^{i,t,\tau} = [z_1^{i,t,\tau}, z_2^{i,t,\tau} \cos y_1^{i,t+\tau}]$$

the geographic distance can be further simplified as follows:

$$d(\mathbf{z}^{i,t,\tau}, \mathbf{y}^{i,t+\tau})^2 = \|\tilde{\mathbf{z}}^{i,t,\tau} - \tilde{\mathbf{y}}^{i,t+\tau}\|^2$$

which is an ℓ_2 loss on the transformed coordinates.

3.3.3 `OMuLeT` Framework

Our proposed framework, named `OMuLeT`, learns the optimal weights for the ensemble members in an online fashion. Let M be the total number of ensemble members and m_i be the number of ensemble members whose forecasts are available for hurricane h_i. Assume we have extracted the subvector $\mathbf{w}^i$ whose elements consist of the weights of the m_i selected members and renormalize the weights to sum up to 1. To predict the location of hurricane h_i at time $t+\tau$, we use the following linear estimator, $\mathbf{z}^{i,t,\tau} = \mathbf{X}^{i,t,\tau}\mathbf{w}^{i,t,\tau}$, where $\mathbf{X}^{i,t,\tau} \in \mathbb{R}^{2\times m_i}$, $\mathbf{w}^{i,t,\tau} \in \mathbb{R}^{m_i}$, and $\mathbf{1}_{m_i}^T \mathbf{w}^{i,t,\tau} = 1$. In addition, all weights should be non-negative according Theorem 1, where the weight is assumed to be $\frac{1}{\sigma^2}$. To simplify the problem, we relax it and remove the weight non-negative constraint. Instead, we project the weight to a non-negative weight space. The details is discussed in the following section.

OMuLeT assumes the weight vector $\mathbf{w}^{i,t,\tau}$ can be decomposed into the sum of the following three factors:

$$\mathbf{w}^{i,t,\tau} = \mathbf{o}^i + \mathbf{u}^{i,t} + \mathbf{v}^{i,t,\tau} \tag{3.6}$$
$$\text{where} \quad \mathbf{1}_{m_i}^T \mathbf{o}^i = 1, \mathbf{1}_{m_i}^T \mathbf{u}^{i,t} = 0, \mathbf{1}_{m_i}^T \mathbf{v}^{i,t,\tau} = 0$$

The first term, $\mathbf{o}^i$, is a *global weight* factor that retains the weight information from the analysis of past hurricanes. The second term, $\mathbf{u}^{i,t}$, is a *hurricane-specific* factor that modifies

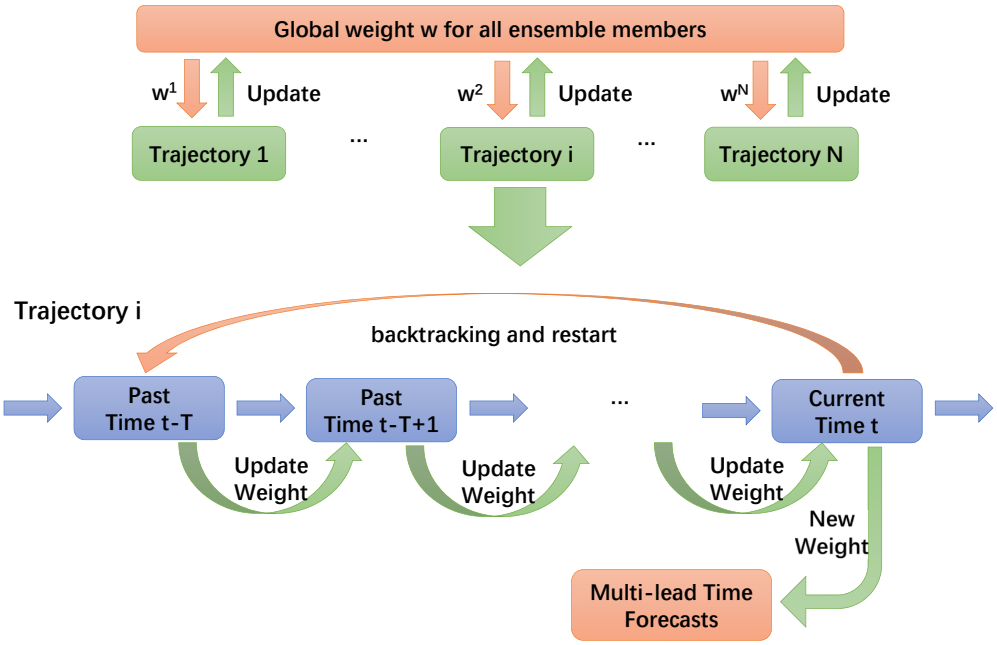


Figure 3.4: Proposed online multi-lead time trajectory forecasting framework.

the global weight to better fit the prediction for the current hurricane. The third term, $\mathbf{v}^{i,t,\tau}$, is a *lead time adjustment* factor to improve the model prediction at lead time τ.

The overall framework is shown in Figure 3.4. Given a hurricane h_i, we extract the subvector $\mathbf{w}^i$ from $\mathbf{w}$ and normalize it to have unit sum before assigning it as the initial value for $\mathbf{o}^i$. Both $\mathbf{u}^{i,0}$ and $\mathbf{v}^{i,0,\tau}$ are initialized to a vector of all zeros, 0_{m_i}. As new observation data become available, we will update $\mathbf{u}^{i,t}$ and $\mathbf{v}^{i,t,\tau}$ accordingly using the online approach described in Section 3.3.3.1. Note that the global weight $\mathbf{o}^i$ is not updated throughout the online update. After processing all the observation data for hurricane h_i, $\mathbf{o}^i$ is used to derive the new weight $\mathbf{w}^i$ as follows:

$$\mathbf{w}^i = \mathbf{o}^i + \rho \mathbf{u}^{i,\Gamma_i}. \tag{3.7}$$

where Γ_i denotes the trajectory length for h_i. The hyperparameter ρ controls the tradeoff between biasing $\mathbf{w}^i$ with the learned weights from current and past hurricanes. Finally, the weights in $\mathbf{w}^i$ are renormalized based on the formula given in Corollary 1 before replacing their original values in $\mathbf{w}$.

3.3.3.1 Objective Function

The online learning process for trajectory i is illustrated in the bottom diagram of Figure 3.4. At time t, when the true location $\mathbf{y}^{i,t}$ is known, it can be used to verify the following set of earlier forecasts, $\mathcal{X}^{i,t} = \{\mathbf{X}^{i,t-1,1}, \mathbf{X}^{i,t-2,2}, \cdots, \mathbf{X}^{i,t-T,T}\}$. Let $\mathbf{W}^{i,t-1} = [\mathbf{w}^{i,t-1,1}, \mathbf{w}^{i,t-1,2}, \cdots, \mathbf{w}^{i,t-1,T}]$ denote the weight matrix for all lead times at time $t-1$. Using $\mathbf{y}^{i,t}$ to update the weight matrix $\mathbf{W}^{i,t-1}$ alone is insufficient as some of the weight vectors in $\mathbf{W}^{i,t-1}$ are also outdated since they ignore the true values for $\mathcal{X}^{i,t}$. Instead, our proposed framework employs a backtracking and restart strategy in its online learning process. Specifically, since $\mathbf{W}^{i,t-1}$ was updated from $\mathbf{W}^{i,t-2}$, which in turn, was updated from $\mathbf{W}^{i,t-3}$, and so on, we restart the weight update procedure from time $t-T$ and update $\mathbf{w}^{i,t-T,T}$ to account for the new ground truth value available for $\mathbf{X}^{i,t-T,T}$. The updated weight matrix $\mathbf{W}^{i,t-T}$ is then used to update $\mathbf{W}^{i,t-T+1}$, taking into account the new ground truth value available for $\mathbf{X}^{i,t-T+1,T-1}$. This procedure is repeated until the new weight matrix $\mathbf{W}^{i,t}$ is obtained.

The weights are updated based on the objective function below:

$$
\begin{aligned}
\min_{\mathbf{u}^{i,t},\{\mathbf{v}^{i,t,\tau}\}} \quad & \frac{1}{2}\sum_{\tau=1}^{T} \delta^{i,t,\tau}\gamma^{\tau} d\big[\mathbf{z}^{i,t,\tau}, \mathbf{y}^{i,t+\tau}\big]^2 \\
+ \quad & \frac{\omega}{2}\sum_{\tau=1}^{T-1} \left\|\mathbf{w}^{i,t,\tau+1} - \mathbf{w}^{i,t,\tau}\right\|^2 + \frac{\mu}{2}\left\|\mathbf{u}^{i,t} - \mathbf{u}^{i,t-1}\right\|^2 \\
+ \quad & \frac{\nu}{2}\sum_{\tau=1}^{T} \left\|\mathbf{v}^{i,t,\tau} - \mathbf{v}^{i,t-1,\tau}\right\|^2 + \frac{\eta}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{i,t,\tau}\right\|^2 \\
\text{s.t.} \quad & \forall\, t,\tau:\ \mathbf{1}_{m_i}^T\mathbf{u}^{i,t} = 0, \mathbf{1}_{m_i}^T\mathbf{v}^{i,t,\tau} = 0,
\end{aligned} \tag{3.8}
$$

where $d[\cdot]$ is the geographic distance function described in Section 3.3.2 while $\delta^{i,t,\tau}$ is delta function whose value is 1 if $\mathbf{y}^{i,t+\tau}$ is known at time t and 0 otherwise. The first term in the objective function represents the forecast error. The hyperparameter γ determines the relative importance of making accurate forecasts at different lead times τ. The second term ensures smoothness in the model parameters for different lead times whereas the third and fourth terms are designed to ensure the hurricane-specific factor $\mathbf{u}^{i,t}$ and lead time adjustment

factor $\mathbf{v}^{i,t}$ do not change rapidly from their previous values at time $t-1$. The last term in the objective function imposes a sparsity constraint on the lead time adjustment factor.

The Lagrange formulation for the optimization problem is

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\sum_{\tau=1}^{T}\delta^{i,t,\tau}\gamma^{\tau}\left\|\tilde{\mathbf{X}}^{i,t,\tau}\mathbf{w}^{i,t,\tau}-\tilde{\mathbf{y}}^{i,t+\tau}\right\|_2^2 \\
&+ \frac{1}{2}Tr\left[\mathbf{V}^{i,t^T}(\omega\mathbf{L}+\eta\mathbf{I})\mathbf{V}^{i,t}\right] \\
&+ \frac{\mu}{2}\left\|\mathbf{u}^{i,t}-\mathbf{u}^{i,t-1}\right\|^2+\frac{\nu}{2}\left\|\mathbf{V}^{i,t}-\mathbf{V}^{i,t-1}\right\|_F^2 \\
&- \lambda\mathbf{1}_{m_i}^T\mathbf{u}^{i,t}-\sum_{\tau=1}^{T}\theta_{\tau}\mathbf{1}_{m_i}^T\mathbf{v}^{i,t,\tau}
\end{aligned} \tag{3.9}$$

where $\mathbf{V}^{i,t}=[\mathbf{v}^{i,t,1},\mathbf{v}^{i,t,2},\ldots,\mathbf{v}^{i,t,T}]$ and $\mathbf{L}$ is a matrix defined as follows

$$\mathbf{L_{i,j}}=\begin{cases}1, & \text{if } i=j=1 \text{ or } i=j=T\\ 2, & \text{if } i=j\neq 1 \text{ and } i=j\neq T\\ -1, & \text{if } i=j+1 \text{or } i=j-1\\ 0, & \text{otherwise}\end{cases}$$

To solve the function, taking the partial derivative of $\mathcal{L}$ and setting it to zero. Then, we can built an equation as follows

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\mathbf{u}^{i,t}} =& \sum_{\tau=1}^{T}\delta^{i,t,\tau}\gamma^{\tau}\tilde{\mathbf{X}}^{i,t,\tau^T}\left(\tilde{\mathbf{X}}^{i,t,\tau}\mathbf{w}^{i,t,\tau}-\tilde{\mathbf{y}}^{i,t+\tau}\right)+\mu\left(\mathbf{u}^{i,t}-\mathbf{u}^{i,t-1}\right)-\lambda\mathbf{1}_{d_i} \\
=& \sum_{\tau=1}^{T}\delta^{i,t,\tau}\gamma^{\tau}\tilde{\mathbf{X}}^{i,t,\tau^T}\left(\tilde{\mathbf{X}}^{i,t,\tau}(\mathbf{w}^{i,t-1,\tau}+\Delta\mathbf{u}^{i,t}+\Delta\mathbf{v}^{i,t,\tau})-\tilde{\mathbf{y}}^{i,t+\tau}\right)+\mu\Delta\mathbf{u}^{i,t}-\lambda\mathbf{1}_{d_i} \\
=& \left(\tilde{\mathbf{M}}^{i,t}+\mu\mathbf{I}_{d_i}\right)\Delta\mathbf{u}^{i,t}+\sum_{\tau=1}^{T}\delta^{i,t,\tau}\mathbf{M}^{i,t,\tau}\Delta\mathbf{v}^{i,t,\tau}+\tilde{\mathbf{C}}^{i,t}-\lambda\mathbf{1}_{d_i} \\
=& \mathbf{0}_{d_i}
\end{aligned} \tag{3.10}$$

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\mathbf{v}^{i,t,\hat{\tau}}} =& \delta^{i,t,\hat{\tau}}\gamma^{\hat{\tau}}\tilde{\mathbf{X}}^{i,t,\hat{\tau}^T}\left(\tilde{\mathbf{X}}^{i,t,\tau}\mathbf{w}^{i,t,\hat{\tau}}-\tilde{\mathbf{y}}^{i,t+\hat{\tau}}\right)+\left[\mathbf{V}^{i,t}\mathbf{Q}\right]_{\tau}+\nu\left(\mathbf{v}^{i,t,\hat{\tau}}-\mathbf{v}^{i,t-1,\hat{\tau}}\right)-\theta_{\hat{\tau}}\mathbf{1}_{d_i} \\
=& \delta^{i,t,\hat{\tau}}\gamma^{\hat{\tau}}\tilde{\mathbf{X}}^{i,t,\hat{\tau}^T}\left(\tilde{\mathbf{X}}^{i,t,\hat{\tau}}(\mathbf{w}^{i,t-1,\hat{\tau}}+\Delta\mathbf{u}^{i,t}+\Delta\mathbf{v}^{i}_{t,\hat{\tau}})-\tilde{\mathbf{y}}^{i,t+\hat{\tau}}\right)+\left[\mathbf{V}^{i,t}\mathbf{Q}\right]_{\tau}+\nu\Delta\mathbf{v}^{i,t,\hat{\tau}}-\theta_{\hat{\tau}}\mathbf{1}_{d_i} \\
=& \delta^{i,t,\hat{\tau}}\mathbf{M}^{i}_{t,\hat{\tau}}\Delta\mathbf{u}^{i,t}+\left(\delta^{i,t,\hat{\tau}}\mathbf{M}^{i}_{t,\hat{\tau}}+\nu\mathbf{I}_{d_i}\right)\Delta\mathbf{v}^{i}_{t,\hat{\tau}}+\left[\mathbf{V}^{i,t}\mathbf{Q}\right]_{\tau}+\delta^{i,t,\hat{\tau}}\mathbf{C}^{i}_{t,\hat{\tau}}-\theta_{\hat{\tau}}\mathbf{1}_{d_i} \\
=& \mathbf{0}_{d_i}
\end{aligned} \tag{3.11}$$

where $[\mathbf{V}^{i,t}\mathbf{Q}]_\tau$ is the τ_{th} column of $[\mathbf{V}^{i,t}\mathbf{Q}]$. From the definition, we have

$$\left[\mathbf{V}^{i,t}\mathbf{Q}\right]_\tau = \begin{cases} (\omega+\eta)\mathbf{v}^{i,t,1} - \omega\mathbf{v}^{i,t,2} & \text{if } \tau = 1 \\ (2\omega+\eta)\mathbf{v}^{i,t,\tau} - \omega\mathbf{v}^{i,t,\tau-1} - \omega\mathbf{v}^{i,t,\tau+1} & \text{if } 1 < \tau < T \\ (\omega+\eta)\mathbf{v}^{i,t,T} - \omega\mathbf{v}^{i,t,T-1} & \text{if } \tau = T \end{cases} \tag{3.12}$$

Furthermore, I can express it in the term of $\Delta\mathbf{v}^{i,t,\tau}$

$$\left[\mathbf{V}^{i,t}\mathbf{Q}\right]_\tau = \begin{cases} (\omega+\eta)\Delta\mathbf{v}^{i,t,1} - \omega\Delta\mathbf{v}^{i,t,2} + (\omega+\eta)\mathbf{v}^{i,t-1,1} - \omega\mathbf{v}^{i,t-1,2} & \text{if } \tau = 1 \\ (2\omega+\eta)\Delta\mathbf{v}^{i,t,\tau} - \omega\Delta\mathbf{v}^{i,t,\tau-1} - \omega\Delta\mathbf{v}^{i,t,\tau+1} + (2\omega+\eta)\mathbf{v}^{i,t-1,\tau} & \\ \qquad -\omega\mathbf{v}^{i,t-1,\tau-1} - \omega\mathbf{v}^{i,t-1,\tau+1} & \text{if } 1 < \tau < T \\ (\omega+\eta)\Delta\mathbf{v}^{i,t,T} - \omega\Delta\mathbf{v}^{i,t,T-1} + (\omega+\eta)\mathbf{v}^{i,t-1,T} - \omega\mathbf{v}^{i,t-1,T-1} & \text{if } \tau = T \end{cases} \tag{3.13}$$

Using the Lagrange multiplier method with the notation given in Table 3.2, a closed-form solution can be found by solving the following system of linear equations: $\mathbf{A}^{i,t}\varphi^{i,t} = \mathbf{b}^{i,t}$, where

$$\mathbf{A}^{i,t} = \begin{bmatrix} \mathbf{A}^{i,t}_{1,1} & \mathbf{A}^{i,t}_{1,2} \\ \mathbf{A}^{i,t}_{2,1} & \mathbf{0} \end{bmatrix} \qquad \varphi^{i,t} = \begin{bmatrix} \boldsymbol{\Delta}u^{i,t} \\ \boldsymbol{\Delta}v^{i,t,1} \\ \vdots \\ \boldsymbol{\Delta}v^{i,t,T} \\ \lambda \\ \theta_1 \\ \vdots \\ \theta_T \end{bmatrix}$$

$$[\mathbf{b}^{i,t}]_j = \begin{cases} -\tilde{\mathbf{C}}^{i,t} & \text{if } j = 1 \\ -\delta^{i,t,j-1}\mathbf{C}^{i,t,j-1} - (\omega+\eta)\mathbf{v}^{i,t-1,j-1} + \omega\mathbf{v}^{i,t-1,j} & \text{if } j = 2 \\ -\delta^{i,t,j-1}\mathbf{C}^{i,t,j-1} - (2\omega+\eta)\mathbf{v}^{i,t-1,j-1} + \omega\mathbf{v}^{i,t-1,j-2} & \\ \quad +\omega\mathbf{v}^{i,t-1,j} & \text{if } 2 < j \leq T \\ -\delta^{i,t,j-1}\mathbf{C}^{i,t,j-1} - (\omega+\eta)\mathbf{v}^{i,t-1,j-1} + \omega\mathbf{v}^{i,t-1,j-2} & \text{if } j = T+1 \\ 0 & \text{otherwise} \end{cases}$$

$$[\mathbf{A}^{i,t}_{1,1}]_{j,k} = \begin{cases} \tilde{\mathbf{M}}^{i,t} + \mu\mathbf{I}_{m_i} & \text{if } j = 1, k = 1 \\ \delta^i_{t,j}\mathbf{M}^i_{t,j} & \text{if } j > 1, k = 1 \\ \delta^i_{t,k}\mathbf{M}^i_{t,k} & \text{if } j = 1, k > 1 \\ -\omega\mathbf{I}_{m_i} & \text{if } 2 \leq j \leq T, k = j+1 \\ -\omega\mathbf{I}_{m_i} & \text{if } 2 \leq k \leq T, j = k+1 \\ \delta^{i,t,1}\mathbf{M}^{i,t,1} + (\omega+\eta+\nu)\mathbf{I}_{m_i} & \text{if } j = 2, k = 2 \\ \delta^{i,t,T}\mathbf{M}^{i,t,T} + (\omega+\eta+\nu)\mathbf{I}_{m_i} & \text{if } j = T+1, k = T+1 \\ \delta^{i,t,T}\mathbf{M}^{i,t,T} + (2\omega+\eta+\nu)\mathbf{I}_{m_i} & \text{if } j = k, 2 < j < T+1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{A}^{i,t}_{1,2} = -\mathbf{I}_{T+1} \otimes \mathbf{1}_{m_i}$$

$$\mathbf{A}^{i,t}_{2,1} = \mathbf{I}_{T+1} \otimes \mathbf{1}^T_{m_i}$$

where $[\mathbf{b}^{i,t}]_j$ denote the j-th block of $\mathbf{b}^{i,t}$ and $[\mathbf{A}^{i,t}_{1,1}]_{j,k}$ denote the jk-th block of $\mathbf{A}^{i,t}_{1,1}$. After obtaining $\varphi^{i,t}$, the weights $\mathbf{u}^{i,t}$ and $\mathbf{v}^{i,t,\tau}$ are updated as follows:

$$\begin{aligned} \mathbf{u}^{i,t} &= \mathbf{u}^{i,t-1} + \Delta\mathbf{u}^{i,t} \\ \mathbf{v}^{i,t,\tau} &= \mathbf{v}^{i,t-1,\tau} + \Delta\mathbf{v}^{i,t,\tau} \end{aligned} \tag{3.14}$$

The pseudocode of our framework is shown in Algorithm 1.

Notation	Definition
$\mathbf{0}_d$	a d-dimensional column vector of 0s
$\mathbf{1}_d$	a d-dimensional column vector of 1s
$\delta^{i,t,\tau}$	$\begin{cases} 1, & \text{if } \mathbf{X}^{i,t,\tau} \text{ exist and } \mathbf{y}^{i,t+\tau} \text{ is observed} \\ 0, & \text{otherwise} \end{cases}$
$\Delta\mathbf{u}^{i,t}$	$\mathbf{u}^{i,t} - \mathbf{u}^{i,t-1}$
$\Delta\mathbf{v}^{i,t,\tau}$	$\mathbf{v}^{i,t,\tau} - \mathbf{v}^{i,t-1,\tau}$
$\mathbf{M}^{i,t,\tau}$	$\gamma^\tau \tilde{\mathbf{X}}^{i,t,\tau^T} \tilde{\mathbf{X}}^{i,t,\tau}$
$\tilde{\mathbf{M}}^{i,t}$	$\sum_{\tau=1}^{T} \delta^{i,t,\tau} \mathbf{M}^{i,t,\tau}$
$\mathbf{C}^{i,t,\tau}$	$\gamma^\tau \tilde{\mathbf{X}}^{i,t,\tau^T} \left(\tilde{\mathbf{X}}^{i,t,\tau} \mathbf{w}^{i,t-1,\tau} - \mathbf{y}^{i,t+\tau} \right)$
$\tilde{\mathbf{C}}^{i,t}$	$\sum_{\tau=1}^{T} \delta^{i,t,\tau} \mathbf{C}^{i,t,\tau}$

Table 3.2: Summary of notations.

Input: $\rho, \gamma, \omega, \mu, \nu, \eta$
Output: Ensemble forecasts $\mathbf{z}$
Initialize: $\mathbf{o} = \mathbf{1}^N / M$;
for $i = 1, 2, \ldots, N$ **do**
  Extract $\mathbf{o}^i$ from $\mathbf{w}$;
  Initialize: $\forall t, \tau : \mathbf{u}^{i,t} = \mathbf{0}_{m_i}, \mathbf{v}^{i,t,\tau} = \mathbf{0}_{m_i}$;
  for $t = 1, 2, \ldots, \Gamma_i$ **do**
    Observe $\mathbf{y}^{i,t}$;
    for $t' = t - T, t - T + 1, \ldots, t$ **do**
      Update $\mathbf{u}^{i,t'}, \{\mathbf{v}^{i,t',\tau}\}$ using Eq. (3.14);
    end
    Make new forecasts using linear estimator;
  end
  Update $\mathbf{w}^i$ based on Eq. (3.7);
end

Algorithm 1: Proposed `OMuLeT` framework

3.3.3.2 Computational Complexity

To process each hurricane trajectory, the weight update formula involves calculating Eq. (3.14), which requires computing the matrix $\mathbf{A}^{i,t} \in \mathbb{R}^{(T+1)(m_i+1)\times(T+1)(m_i+1)}$. Calculating $\mathbf{A}^{i,t}$ requires $O(T^2 m_i^2)$ floating point operations (flops), while solving the system of linear equations for $\varphi^{i,t}$ takes $O(T^3 m_i^3)$ flops. Therefore, the overall complexity for processing all T lead times is $O(T^4 \Gamma_i m_i^3)$. Each hurricane in our dataset has an average trajectory length $\Gamma_{avg} \approx 24$ time steps, a forecast lead time of at most $T = 20$ time steps, and $m_i \leq 30$

ensemble members. The overall computation needed for each hurricane is at most hundreds of billions of floating point operations, which can be easily computed in less than a minute on modern day computers with CPUs that can process tens of billions of operations per second (GigaFlops).

3.4 Experimental Evaluation

The hurricane best track (ground truth) data and NHC official forecasts are available from the NHC website[1], while the ensemble member forecasts were downloaded from the Hurricane Forecast Model Output website at University of Wisconsin-Milwaukee[2]. According to NHC, 46 models were used in the preparation of their official forecasts. However, only 28 of them were available at the UWM website, which we use for our experiments.

While the physical models provide the trajectory forecasts, partial models also generate intensity forecasts. According to NHC, 30 models were used in the preparation of their intensity forecasts. 21 of these models were available at the UWM website. In our experiments, we also applying our `OMuLeT` framework on the intensity forecasts.

Although the best track data dates back to 1851 for Atlantic and 1949 for Pacific oceans, both the NHC official forecasts as well as the ensemble of dynamical model forecasts have a much shorter history. After fusing the data, our final dataset contains 212 tropical cyclones spanning 2012 to 2018. We performed linear interpolation to impute the missing values for ensemble members with 12-hourly forecasts to ensure they also generate 6-hourly forecasts. We set the maximum forecast lead time to 48 hours. We use the hurricane data from 2012 to 2014 (84 tropical cyclones) for training and validation while those from 2015 to 2018 (128 tropical cyclones) for testing. Each trajectory has an average length of 24 time steps at 6 hourly intervals. In total, there are 2086 observations in the training period to fit the online model and 2946 observations in the test period. The hurricanes are divided into two groups, those originating from the Atlantic ocean and those from the Eastern Pacific ocean. We

[1] https://www.nhc.noaa.gov
[2] http://derecho.math.uwm.edu/models

report the forecasting results on the hurricanes from both groups as well as those from each group separately. Following the approach used by NHC, we evaluate the forecast error of a method in terms of the average geographic distance between the predicted and best track locations (see Section 3.3.2).

We compared `OMuLeT` against the following baseline methods:

1. **LSTM:** LSTM is a widely used deep learning approach for time series forecasting. Following the approach used in [3], the LSTM model was trained on historical data only. We consider training LSTM using the best track data from 2012 to 2014 as well as using the entire history (since 1851 for Atlantic and 1949 for Pacific oceans). Though the results were not significantly different, we report the test results (2015-2018) for LSTM trained on the longer history.

2. **Ensemble mean (EM):** This corresponds to taking the average of all the ensemble member forecasts.

3. **NHC:** This corresponds to the official forecasts generated by NHC, which is the gold standard for hurricane prediction.

4. **Passive-Aggresive (PA)**[21]: A well-known online algorithm that updates the weights of its linear model based on the following equation:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + sign(\mathbf{y}^t - \mathbf{z}^t)\tau^t\mathbf{x}^t \tag{3.15}$$

5. **ORION**[84]: A recently developed online learning algorithm for multi-lead time prediction.

For a fair comparison, the baseline methods such as PA and ORION also use the weight renormalization strategy to deal with the varying feature length problem.

3.4.1 Performance Comparison

Table 3.3 summarizes the forecast errors of the different methods, in terms of their average geographic distance (in miles) between the true and predicted locations. There are several interesting conclusions can be drawn from the results shown in the table. First, the LSTM results were significantly worse than the results of other methods despite using a longer history of hurricane trajectory data to train the model. This is not surprising as the historical tracks do not capture the varying atmospheric conditions and ocean temperatures that affect the path of the hurricanes. Furthermore, the parameters of the LSTM model are fixed after training unlike online learning models such as PA, ORION, and `OMuLeT` that can continuously update its model with new observation data.

Second, the performance of ensemble mean is comparable to the NHC official forecasts, which validates the skills of the ensemble members. This result is again not surprising as the skills of the dynamical models have continuously improved over the years and the ensemble members considered in this study are a subset of those used in the NHC official forecasts.

Third, existing online learning algorithms such as PA and ORION do not significantly improve the prediction error of ensemble mean even though their weights are updated continuously. PA updates only its latest model whenever new observation data becomes available unlike ORION and `OMuLeT`, which both employ the online update with restart strategy to revise some of the older models as well. This shows the importance of addressing the partially labeled data problem in online learning for multi-lead time forecasting. ORION was designed for multi-lead time forecasting at a single location. Extending the approach to modeling different hurricanes is not effective as it fails to retain the weight information from past hurricanes, unlike the weight decomposition approach used in `OMuLeT` (see Eqs. (3.6)) and (3.7)).

Fourth, the results suggest that `OMuLeT` consistently outperforms all the baseline methods in all regions irrespective of the forecast lead time. In particular, it significantly outperforms the NHC official forecasts, especially at longer lead times. Figure 3.6 illustrates the forecast

improvement of OMuLeT over NHC and other baselines for different lead times. First, observe that the forecast improvement of OMuLeT over the baseline methods continue to grow with increasing lead times. More importantly, OMuLeT outperforms the gold standard, i.e., NHC, by more than 10% for the 48-hour lead time forecast. The significant improvement over other online algorithms also show the effectiveness of the strategies used in OMuLeT to overcome the varying feature length, temporal inconsistencies, and partially labeled data problems encountered in hurricane trajectory forecasting.

Location	Atlantic and Pacific ocean				Atlantic ocean only				Pacific ocean only			
Lead Time	12	24	36	48	12	24	36	48	12	24	36	48
LSTM	142.89	211.42	305.91	537.29	175.62	257.16	361.63	594.01	113.21	161.93	259.31	385.17
Ensemble Mean	26.09	40.20	53.77	68.49	27.74	42.74	57.85	75.39	24.67	38.06	50.33	62.74
PA	26.09	40.07	53.60	68.01	27.68	42.57	57.77	74.87	24.66	38.03	50.29	62.67
ORION	25.50	39.04	51.97	66.27	26.77	41.08	55.55	72.26	24.51	37.46	49.27	61.48
NHC	26.58	40.97	54.41	68.47	28.83	44.03	58.66	75.94	24.65	38.39	50.83	62.25
OMuLeT	**24.94**	**37.08**	**48.78**	**59.08**	**26.53**	**38.77**	**51.85**	**63.61**	**23.73**	**36.03**	**45.85**	**55.80**

Table 3.3: Comparison of mean geographic distance error (in miles) for various hurricane trajectory forecasting methods.

Location	Atlantic and Pacific ocean			
Lead Time	12	24	36	48
LSTM	142.89	211.42	305.91	537.29
Ensemble Mean	26.09	40.2	53.77	68.49
NHC	26.58	40.97	54.41	68.47
PA	26.09	40.07	53.6	68.01
OMuLeT (28 models)	24.94	37.08	48.78	59.08
OMuLeT (28 models + NHC)	24.51	36.93	47.96	58.77
OMuLeT (50 models)	25.34	38.79	50.38	62.16
OMuLeT (100 models)	25.27	38.03	49.12	62.35
OMuLeT (155 models)	25.22	37.88	48.93	60.54

Table 3.4: Comparison of mean geographic distance error (in miles) for various hurricane trajectory forecasting methods.

We also compared the performance of OMuLeT when using different set of models. We begin with the NHC used models as the default models for our OMuLeT framework and then keep increasing more models based on their frequency of use. Table 3.4 demonstrates the forecast errors as more physical model forecasts are incorporated into the OMuLeT framework. If we add NHC official forecasts as one of the input models, the performance of OMuLeT

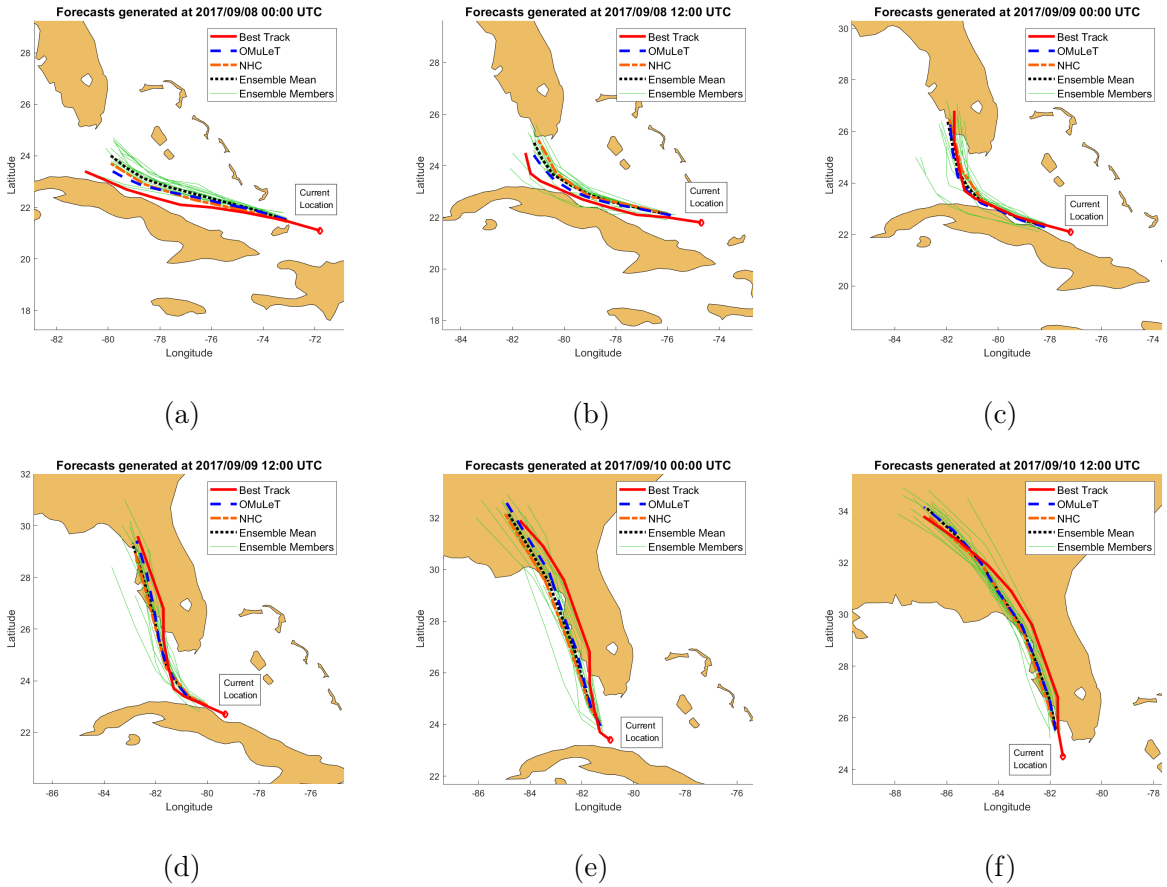


Figure 3.5: Comparison of 48-hour forecasts for Hurricane Irma from 2017/09/08 to 2017/09/10 by different methods.

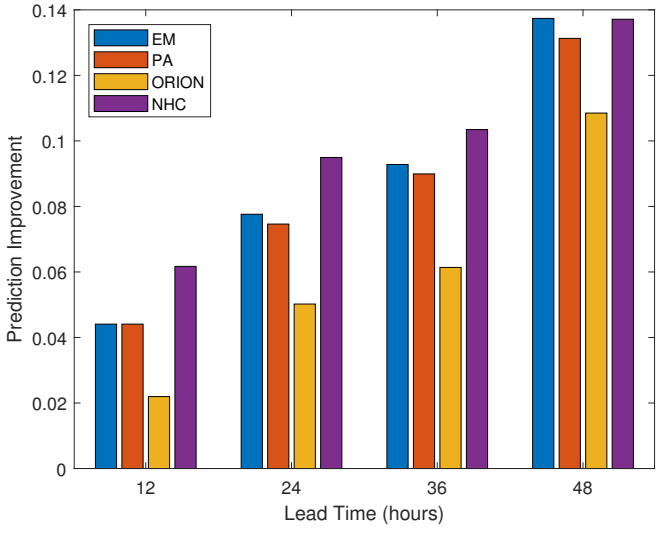


Figure 3.6: Percentage of forecast improvement of `OMuLeT` compared to the baseline methods.

slightly improved. However, we obtain mixed results when a larger number of ensemble members are included into the learning framework. OMuLeT's performance decreases when the number of ensemble members increases from 28 to 50 models. However, the performance improves as the number of ensemble members increases from 50 to 100 and 155 models. Nevertheless, the best result is still obtained by using the 28 models employed by NHC in their official forecasts.

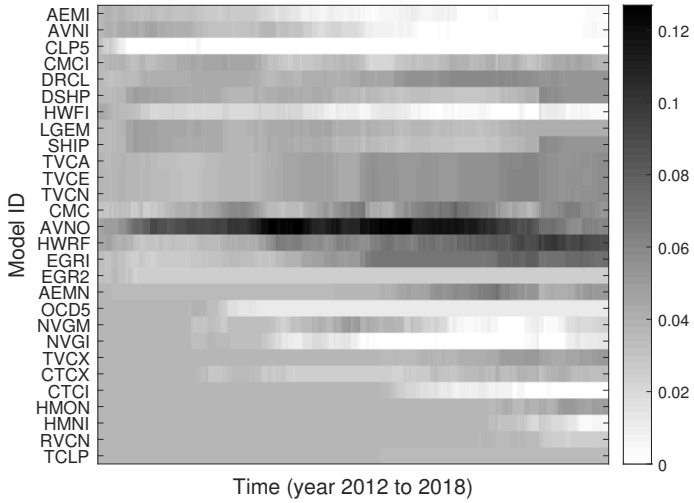


Figure 3.7: Global weight changes over time

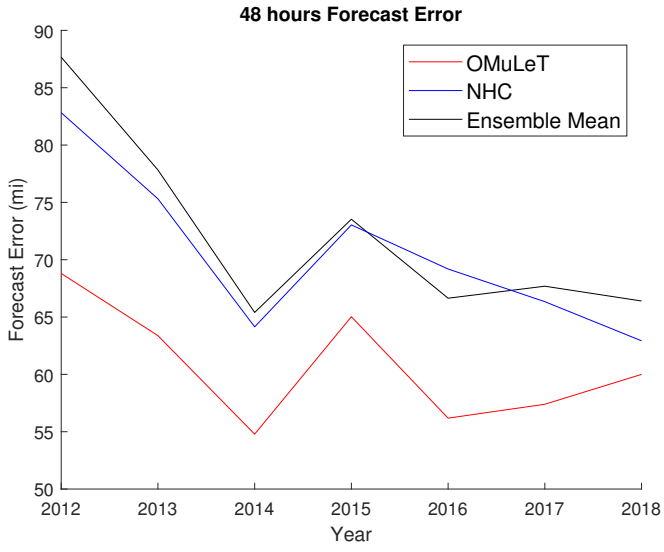


Figure 3.8: The 48 hour forecast error over year 2012 to 2018.

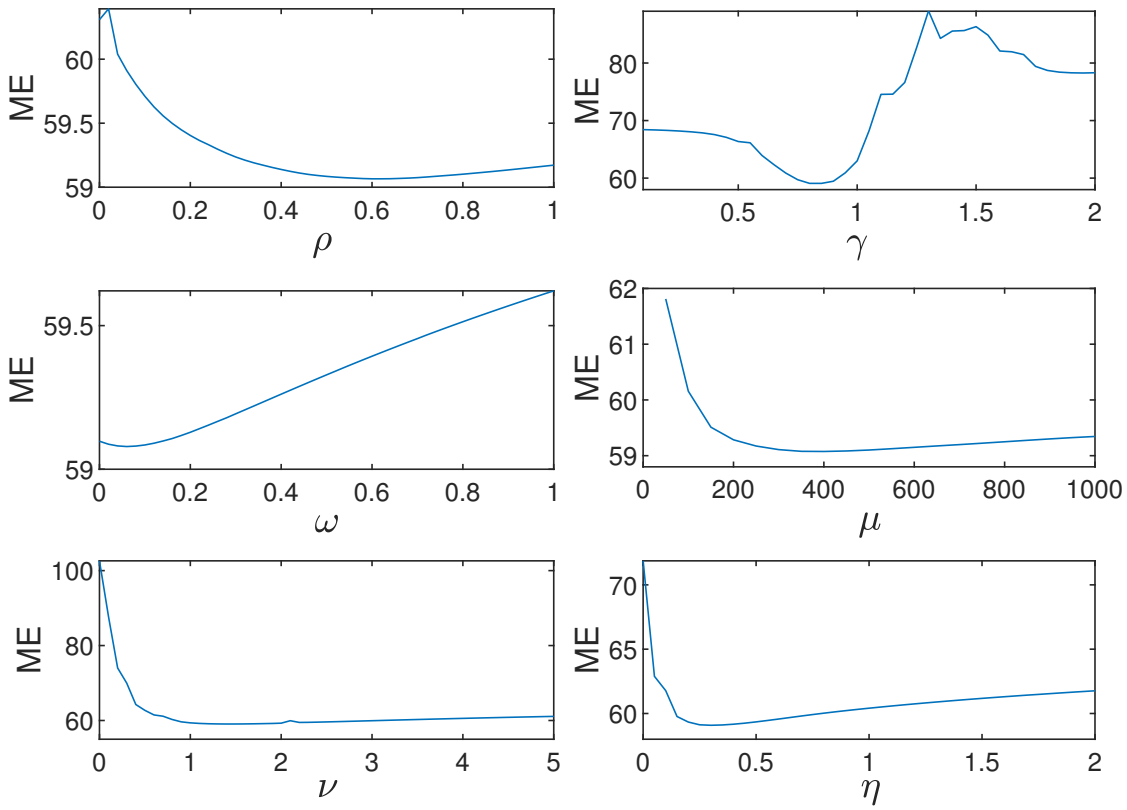


Figure 3.9: Effect of varying the hyperparameters $\rho, \gamma, \omega, \mu, \nu, \eta$ on mean forecast error (ME).

Figure 3.5 shows an example of the trajectory forecasts for Hurricane Irma from 2017/09/08 to 2017/09/10. Observe that `OMuLeT`'s 48-hour forecasts are closest to the best track compared to the baseline methods, especially in Figure 3.5(a) and 3.5(b). Despite the large variability among the ensemble member forecasts, `OMuLeT` was able to assign the appropriate set of weights to the ensemble members, which led to more accurate forecasts.

Figure 3.7 shows the dynamic weights of the ensemble members learned using `OMuLeT`. The plot suggests that the Global Forecast System (AVNO) generally has higher weights than others. However, there are other models such as the Hurricane Weather Research and Forecast system (HWRF) and U.K. Met Office Global Model (EGRI) that have become increasingly skillful in recent years. This result shows the advantage of using an online learning approach that can continuously adapt the weights of the ensemble members instead of using batch learning approaches that assign equal or static weights to the ensemble members.

Figure 3.8 shows the average forecast errors of the models for 48-hour lead time from 2012 to 2018. The plot generally shows a decreasing trend in the forecast error. The official forecasts of NHC are more accurate than the ensemble mean from 2012 to 2014, but have become more similar in recent year. Our proposed `OMuLeT` framework outperforms both ensemble mean and the NHC official forecasts for all the years in the given time period.

3.4.2 Sensitivity Analysis

Figure 3.9 shows the results of `OMuLeT` when varying the hyperparameters of the framework. The hyperparameter ρ controls the tradeoff between updating the weight using the past or recent hurricane data. The result shows that one choosing a ρ around 0.5 or larger can lead to lower forecast error. The results also suggest a low value of γ is more effective, which allows the model for different lead times to vary quite significantly. The bottom three figures show that the regularization parameters should be sufficiently large to ensure the model is sufficiently sparse and does not change too rapidly from one online update round to another. In our experiments, these hyperparameter values are determined based on their performance on the training period. Empirically, there was no noticeable difference between choosing the best hyperparameter based on the forecast error in training or validation period.

3.5 Conclusions

This chapter focuses on addressing research question **RQ1** by developing a novel online learning framework called `OMuLeT` for multi-lead time hurricane trajectory forecasting. Unlike existing methods, `OMuLeT` uses multi-model ensemble member outputs to train its model. `OMuLeT` also employs weight renormalization and backtracking with restart strategies to address the missing value and error propagation problems. Experimental results showed that `OMuLeT` significantly outperforms various baseline methods, including NHC official forecasts, especially for long-range trajectory forecasting.

CHAPTER 4

ONLINE MULTI-LEAD TIME TRAJECTORY STATE PREDICTION WITH ORDINAL DATA

Trajectory prediction aims to infer the future positions of a moving object. In addition to the time-varying location information, a trajectory dataset often contains other (non-spatial) information describing the state of the moving object. For example, the state of a moving vehicle could be the driver's driving proficiency and aggressiveness; the state of a hurricane could be the current wind speed, air pressure and temperature; the state of a pedestrian could be walking speed and emotional state. In many applications, predicting the future state of the moving object is just as important as predicting its trajectory. For example, besides the trajectory path, accurate prediction of the hurricane intensity is essential to determine the severity of its potential damages.

Due to increasing human activities, our climate system is now changing more rapidly than in the past, which affects the frequency and severity of natural disasters such as hurricanes. The destructive ability of hurricanes can be measured by their maximum sustained wind speeds, also known as their intensities. Table 4.1 shows the categories of hurricane intensities according to the Saffir-Simpson hurricane wind scale. Based on data from the NHC, high-intensity hurricanes of categories 3 or higher usually cause huge damages. For example, hurricane Harvey, a category 4 hurricane, caused an estimated $125 billion of property damages and 107 confirmed deaths in 2017 [18, 8]. In 2018, a category 4 hurricane named Florence caused $24.2 billion in damages and 54 deaths [1, 78].

Category	1	2	3	4	5
Wind speeds (mph)	74-95	96-110	111-129	130-156	$\geq$ 157

Table 4.1: Saffir-Simpson hurricane wind scale (SSHWS), for 1-minute maximum sustained winds.

There have been numerous efforts devoted to developing machine learning techniques for

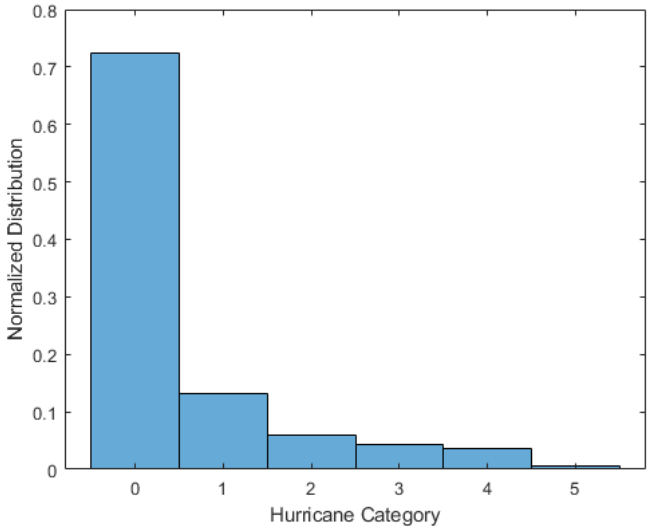


Figure 4.1: Hurricane category distribution from year 2012 to 2020.

the hurricane prediction problem [59, 69, 3]. While these techniques have found success in hurricane trajectory forecasting, challenges remain in applying them to improve the accuracy of hurricane intensity prediction. First, to communicate the severity of an impending hurricane to the public, accurate prediction of its category is often more important than the wind speed itself. Indeed, a prediction error of 60 mph may seem trifle for a category 0 tropical storm but is significant if a category 5 hurricane at 160mph is incorrectly predicted as a category 2 storm at 100mph. Second, high category hurricanes tend to occur less frequently than lower category ones, which leads to a class imbalance problem. Figure 4.1 shows that the hurricane category distribution from year 2012 to 2020. The results suggest that there are much less high category hurricanes. Third, accurate forecasts at longer lead times are needed to ensure there is ample time for emergency preparation.

To address these challenges, we developed two online ordinal regression frameworks called `OOR` (**O**nline **O**rdinal **R**egression) and `OOR-`ϵ (**O**nline **O**rdinal **R**egression with $\underline{\epsilon}$ insensitivity loss) for hurricane category and intensity prediction, respectively. These are extensions of the `OMuLeT` framework described in the Chapter 3 and are designed to address the research question **RQ2**. `OOR` aims to predict the hurricane categories by utilizing an ensemble of

hurricane category forecasts produced from a set of prediction models. In contrast, OOR-ϵ uses the real-valued hurricane intensities and employs an ϵ-insensitive loss function [28] with constraints to preserve the accuracy of its ordinal category prediction. At first glance, the ordinal category of a hurricane may not appear to add any new information since it is derived from the maximum sustained wind speed value (see Table 4.1). However, as will be shown in this work, the constraints help to ensure that the predicted category will not deviate significantly from its true value even though the error in maximum sustained wind speed is large. For example, consider a category 5 hurricane with maximum sustained wind speed of 160 mph. A model that predicts its intensity to be 125 mph will have a lower error than one that predicts its intensity to be 220 mph; yet, the former has a larger category error (since 125 mph is a category 3 hurricane) compared to the latter, which predicts the category correctly. Similarly, a category 2 hurricane at 110 mph will have a lower intensity error if predicted as 130 mph instead of 80 mph. However, in terms of its ordinal category, the former has a larger error since the category 2 hurricane is incorrectly predicted as a major category 4 storm rather than category 1, which is closer. These examples clearly illustrate the importance of leveraging both ordinal category and real-valued intensity information into the learning formulation.

Handling the skewed ordinal category distribution of the data is another challenge as high category hurricanes occur less frequently than lower category ones. As the higher category hurricanes often cost more property destruction and loss of human lives, its accurate detection is critical to minimize its severe impact. In this chapter, we show how our proposed framework can be extended to leverage a quantile loss function to bias its prediction towards forecasting higher category hurricanes more accurately. The extended frameworks are known as OOQR (**O**nline **O**rdinal with **Q**uantile **R**egression) and OOQR-ϵ (**O**nline **O**rdinal with **Q**uantile **R**egression and ϵ insensitivity loss) for hurricane category prediction and intensity prediction, respectively. The developed algorithms with quantile losses answered my research question **RQ3**. Experimental results using real-world data demonstrates the effi-

cacy of our frameworks, outperforming the baseline methods with results comparable to the gold standard, which is the official forecasts of the U.S. National Hurricane Center.

4.1 Related Work

Due to the severe damages could be made by the hurricane, hurricane trajectory and intensity predictions become very critical to give civilians enough time to take appropriate actions. Numerous methods have been developed for hurricane prediction using dynamical models [81, 22], statistical models [24, 76], or a hybrid of statistical-dynamical models [80]. More recently, machine learning approaches have become increasingly popular to improve the accuracy of hurricane prediction tasks [59, 69, 3, 54]. These approaches have been mostly applied to historical trajectory data, ensemble model outputs, or satellite imagery data. Unlike previous works, this research work focuses on designing online learning algorithms for postprocessing the ensemble model outputs to improve hurricane intensity and its ordinal category prediction. Ordinal regression [65] has been widely used for diverse applications, such as credit ratings [25, 53, 32], medical research [27, 15, 73], and wind speed forecasting [46, 36]. Despite recent works on online ordinal regression [20, 43], current methods are not designed to generate both real- and ordinal-valued predictions. They also do not capture the inherent temporal autocorrelation in multi-lead time forecasting problems such as hurricane prediction. The frameworks developed in this chapter were designed to overcome these limitations.

4.2 Problem Statement

The problem of predicting the state of a moving object is similar to predicting the object trajectory. It can be studied based on a set of features derived from historical observations or other methods, such as forecasts generated from a multi-model ensemble for trajectory state prediction. In order to better describe our approach, the following discussion in this chapter is conducted in the context of hurricane prediction, although the problem formulation

and method can be applied to other fields with similar characteristics. Consider a set of hurricanes, $\{h_1, h_2, \ldots, h_C\}$, ordered by their start times. Assuming there are n_i data points (time steps) associated with hurricane h_i, let $N = \sum_{i=1}^{C} n_i$ be the total number of time steps and $t \in \{1, 2, \cdots, N\}$ be the time steps available in the hurricane dataset. At each time step t, assume there are m_t numerical (dynamical) model outputs available to generate our weighted ensemble forecast. If T is the forecast horizon, i.e., maximum lead time forecast, then let $\mathbf{x}^{t,\tau} \in \mathbb{R}^{m_t}$ be the set of forecasts generated by the dynamical models at time t for lead time $\tau \in \{1, 2, \cdots, T\}$. Furthermore, suppose $y^{t,\tau} \in \mathbb{R}$ is the actual hurricane intensity value at $t + \tau$ and $\hat{y}^{t,\tau} \in \{0, 1, \cdots, 5\}$ is its corresponding ordinal category according to the Saffir-Simpson scale. The hurricane category consists of 6 ordinal classes with boundaries $b_j, j \in \{1, ..., 5\}$. Let $b_0 = -\infty$ and $b_6 = \infty$. Then, for any intensity prediction $z^{t,\tau}$ belongs to class $\hat{y}^{t,\tau}$, we have $b_{\hat{y}^{t,\tau}-1} < z^{t,\tau} \leq b_{\hat{y}}^{t,\tau}$.

At each time step t, we use the set of forecasts from the ensemble members $\mathbf{x}^{t,\tau}$ to generate the intensity prediction for lead time τ. First, we consider the real-valued intensity prediction $z^{t,\tau} \in \mathbb{R}$ computed from the following linear predictor

$$z^{t,\tau} = f^{t,\tau}(\mathbf{x}^{t,\tau}) = (\mathbf{w}^{t,\tau})^T \mathbf{x}^{t,\tau} \tag{4.1}$$

where $\mathbf{w}^{t,\tau} \in \mathbb{R}^{m_t}$ is the learned weight vector associated with m_t ensemble member forecasts. The weight vector is updated in an online fashion whenever new observation data becomes available. Unlike standard online learning algorithms, which typically assume that there are no or few missing values in the data, a significant proportion of the ensemble members may not generate any forecasts at a given time t, as shown in Figure 4.2. We termed this as the varying feature length problem (i.e., m_i varies from one hurricane to another). In this chapter, we apply the weight re-normalization technique developed in Chapter 3 to address this problem.

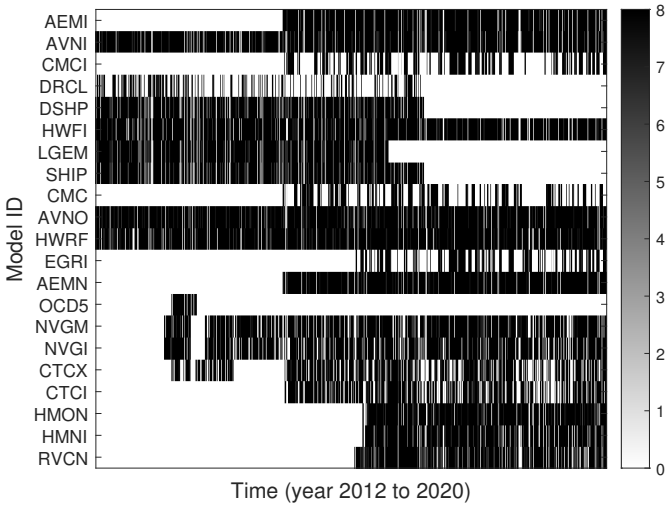


Figure 4.2: Availability of ensemble model forecast data. The dark cells denote the time steps in which the model forecasts are available while the white cells denote otherwise.

4.3 Methodology

This section presents the proposed frameworks for multi-lead time hurricane category prediction and intensity prediction.

4.3.1 `OOR` Framework

`OOR` is a novel online learning algorithm for multi-lead time prediction of ordinal target variables. The framework is an extension of our proposed `OMuLeT` framework in Chapter 3, which was designed for real-valued trajectory prediction. In this framework, the weight vector $\mathbf{w}^{t,\tau}$ is decomposed into the following factors:

$$\mathbf{w}^{t,\tau} = \mathbf{u}^t + \mathbf{v}^{t,\tau} \tag{4.2}$$
$$\text{s.t.} \qquad \mathbf{1}_{m_t}^T \mathbf{u}^t = 0, \mathbf{1}_{m_t}^T \mathbf{v}^{t,\tau} = 0$$

where $\mathbf{u}^t$ is a the shared weight vector for all lead times and $\mathbf{v}^{t,\tau}$ is the lead time adjustment weight vector to improve prediction accuracy at lead time τ.

Let $\mathbf{W}^t = [\mathbf{w}^{t,1}, \mathbf{w}^{t,2}, \cdots, \mathbf{w}^{t,T}]$ be the weight matrix for all lead times at time t. At each

time step t, the weight matrix $\mathbf{W}^t$ is trained to minimize the following objection function:

$$\begin{aligned}
\mathcal{L} = \quad & \frac{1}{2}\sum_{\tau=1}^{T}\delta^{t,\tau}\gamma^{\tau}\left(\zeta^{t,\tau}+\zeta^{*t,\tau}\right)+\frac{\omega}{2}\sum_{\tau=1}^{T-1}\left\|\mathbf{w}^{t,\tau+1}-\mathbf{w}^{t,\tau}\right\|^2 \\
+ \quad & \frac{\mu}{2}\left\|\mathbf{u}^{t}-\mathbf{u}^{t-1}\right\|^2+\frac{\nu}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau}-\mathbf{v}^{t-1,\tau}\right\|^2+\frac{\eta}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau}\right\|^2 \\
\text{s.t.} \quad & \forall\, t,\tau:\ \mathbf{1}_{m_t}^{T}\mathbf{u}^{t}=1,\ \mathbf{1}_{m_t}^{T}\mathbf{v}^{t,\tau}=0, \\
& z^{t,\tau}-b_{\hat{y}^{t,\tau}}\leq -1+\zeta^{t,\tau} \\
& z^{t,\tau}-b_{\hat{y}^{t,\tau}-1}\geq 1-\zeta^{*t,\tau} \\
& \zeta^{t,\tau}\geq 0, \zeta^{*t,\tau}\geq 0
\end{aligned} \tag{4.3}$$

where $\delta^{t,\tau}$ is an indicator function whose value is equal to 1 if $\mathbf{x}^{t,\tau}$ and $\mathbf{y}^{t,\tau}$ values are both available; otherwise its value is 0. γ^{τ} represents the relative importance of the prediction error at lead time τ. The first term in the objective function along with the inequality constraints are analogous to the loss function defined for support vector ordinal regression [17]. The second term ensures that the estimated model parameters would vary smoothly at different lead times, thus preserving the temporal autocorrelation of the predicted intensities. The third and fourth terms ensure that the shared weight vector and lead time adjustment weights are close to their values at previous time step. Finally, the last term penalizes large values in the lead time adjustment weight vectors. The objective function is solved using standard quadratic programming solvers.

For multi-lead time prediction, it is insufficient to update $\mathbf{W}^t$ from $\mathbf{W}^{t-1}$ alone to prevent the error in $\mathbf{W}^{t-1}$ being propagated to its future prediction. To address this problem, we apply the online learning with backtracking and restart strategy described in Chapter 3. Specifically, at each time step t, we use the newly available ground truth values $\{\mathbf{y}^{t-T,T}, \mathbf{y}^{t-T+1,T-1}, \ldots, \mathbf{y}^{t-1,1}\}$ to verify the long-term forecasts in $\mathbf{W}^{t-1}, \mathbf{W}^{t-2}, \cdots, \mathbf{W}^{t-T}$. Starting from $\mathbf{W}^{t-T}$, we solve the objective function to re-compute $\mathbf{W}^{t-T+1}$, which is then used to estimate $\mathbf{W}^{t-T+2}$ and so on until $\mathbf{W}^{t}$ is obtained. This strategy is illustrated in Figure 4.3.

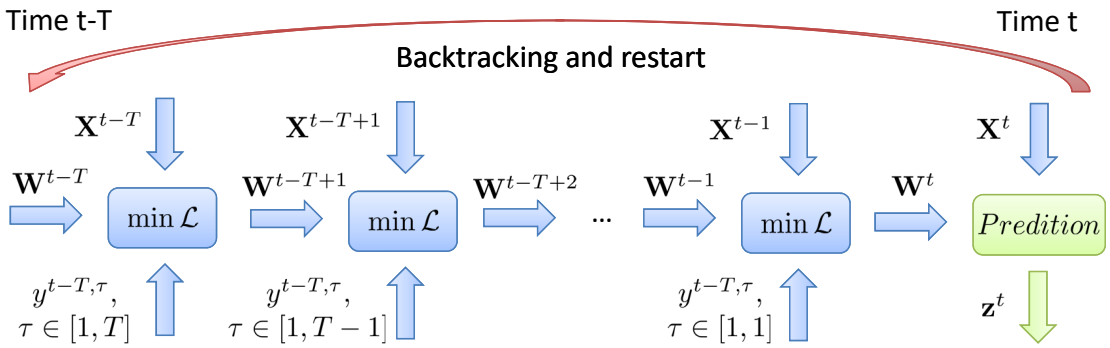


Figure 4.3: OOR/OOR-ϵ framework with backtracking and restart.

Input: $\rho, \gamma, \omega, \mu, \nu, \eta$
Output: Model parameters $\mathbf{W}$ and forecasts $\mathbf{Z}$
Initialize: $\mathbf{W} = \mathbf{I}_M/M, \mathbf{u} = \mathbf{1}_M/M$;
for $t = 1, 2, \ldots, N$ **do**
  Extract $\mathbf{W}^t$ from $\mathbf{W}$,$\mathbf{u}^t$ from $\mathbf{u}$
  Normalize: $\mathbf{w}^{t,\tau} \leftarrow \mathbf{w}^{t,\tau}/c^\tau$,$\mathbf{u}^t \leftarrow \mathbf{u}^t/c$
  `// Backtracking and restart step`
  for $t' = t-T, t-T+1, \ldots, t-1$ **do**
    Solve objective function Eqn. (4.3) for OOR
    Solve objective function Eqn. (4.5) for OOQR
    Solve objective function Eqn. (4.10) for OOR-ϵ
    Solve objective function Eqn. (4.11) for OOQR-ϵ
    Update the weights $\mathbf{W}^{t'+1}$ and $\mathbf{u}^{t'+1}$
  end
  `// Prediction step`
  for $\tau = 1, 2, \cdots, T$ **do**
    $\mathbf{z}^{t,\tau} = \mathbf{X}^{t,\tau}\mathbf{w}^{t,\tau}$
  end
  Renormalize: $\mathbf{w}^{t,\tau} \leftarrow c^\tau \mathbf{w}^{t,\tau}$,$\mathbf{u}^t \leftarrow c\mathbf{u}^t$
  Substitute $\mathbf{W}^t$ back into the full vector $\mathbf{W}$
  Substitute $\mathbf{u}^t$ back into the full vector $\mathbf{u}$
  if $t = \tilde{n}_i$ **then**
    `// End of hurricane predictions`
    Update weights using Eqn. (4.4)
  end
end

Algorithm 2: Proposed OOR/OOQR/OOR-ϵ/OOQR-ϵ framework

The pseudocode for OOR framework is shown in Algorithm 2. Since the model is updated incrementally, we need to consider how to transition the model weights from one hurricane to another. Let $\tilde{n}_i$ be the accumulated number of data points from hurricane h_1 to h_i, i.e.

$\tilde{n}_i = \sum_{j=1}^{i} n_j$. When $t = \tilde{n}_i$, which is the last time step observed for hurricane h_i, we need to control the percentage how the current weight affects the prediction for future hurricanes. This requires a tradeoff between using the weights from the current hurricane against past hurricanes, which is governed by a hyperparameter ρ, defined as follows:

$$\begin{aligned} \mathbf{W}^{\tilde{n}_i} &\leftarrow \rho \mathbf{W}^{\tilde{n}_i} + (1-\rho)\mathbf{W}^{\tilde{n}_{i-1}+1} \\ \mathbf{u}^{\tilde{n}_i} &\leftarrow \rho \mathbf{u}^{\tilde{n}_i} + (1-\rho)\mathbf{u}^{\tilde{n}_{i-1}+1} \end{aligned} \tag{4.4}$$

4.3.2 OOQR Framework

As hurricanes of categories 3 or higher can cause more severe damages, it is essential to predict such hurricanes more accurately compared to lower category hurricanes. In this case, a quantile loss is useful to bias the model to achieve more accurate predictions for high-category hurricanes. OOR can be extended to accommodate the following quantile loss function (the first term in Eqn. (4.5)) in order to penalize models that incorrectly predict high category hurricanes.

$$\begin{aligned} \mathcal{L} = \quad & \sum_{\tau=1}^{T} \delta^{t,\tau}\gamma^{\tau}\left((1-\xi)\zeta^{t,\tau} + \xi\zeta^{*t,\tau}\right) + \frac{\omega}{2}\sum_{\tau=1}^{T-1}\left\|\mathbf{w}^{t,\tau+1} - \mathbf{w}^{t,\tau}\right\|^2 \\ + \quad & \frac{\mu}{2}\left\|\mathbf{u}^{t} - \mathbf{u}^{t-1}\right\|^2 + \frac{\nu}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau} - \mathbf{v}^{t-1,\tau}\right\|^2 + \frac{\eta}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau}\right\|^2 \\ \text{s.t.} \quad & \forall\, t,\tau:\ \mathbf{1}_{m_t}^{T}\mathbf{u}^{t} = 1,\ \mathbf{1}_{m_t}^{T}\mathbf{v}^{t,\tau} = 0, \\ & z^{t,\tau} - b_{\hat{y}^{t,\tau}} \le -1 + \zeta^{t,\tau} \\ & z^{t,\tau} - b_{\hat{y}^{t,\tau}-1} \ge 1 - \zeta^{*t,\tau} \\ & \zeta^{t,\tau} \ge 0, \zeta^{*t,\tau} \ge 0 \end{aligned} \tag{4.5}$$

where ξ is the quantile hyperparameter. Note that $\zeta^{t,\tau}$ represents the loss term if the predicted category is higher than the true category, whereas $\zeta^{*t,\tau}$ represents the loss term if the predicted category is less than the true category. As the prediction can either be larger or smaller than the ground truth, only one of the two losses is incurred. The quantile hyperparameter ξ represents the importance of the two losses. When $\xi \approx 1$, predictions with higher

categories are encouraged as the loss from these predictions are much smaller. Furthermore, the OOQR framework is equivalent to OOR when $\xi = 0.5$.

4.3.3 OOR-ϵ Framework

The OOR and OOQR frameworks described in the previous sections were designed to generate ordinal-valued predictions only. In this section, we extend OOR to OOR-ϵ to utilized both the hurricane intensity and category information. This is accomplished by incorporating an ϵ-insensitivity loss function [84] into the formulation. We begin by describing our approach for generating real-valued predictions. A standard approach is to learn a regression function that minimizes the mean square error (MSE) loss function. Unlike mean square error, the ϵ-insensitive loss is more robust as it provides a margin of tolerance ϵ [28, 5] when learning the regression function. The loss function can be defined as follows:

$$\begin{aligned} \mathcal{L} = \quad & \zeta^{t,\tau} + \zeta^{*t,\tau} \\ \text{s.t.} \quad & z^{t,\tau} - y^{t,\tau} \leq \epsilon + \zeta^{t,\tau} \\ & z^{t,\tau} - y^{t,\tau} \geq -\epsilon - \zeta^{*t,\tau} \\ & \zeta^{t,\tau} \geq 0, \zeta^{*t,\tau} \geq 0 \end{aligned} \tag{4.6}$$

where $\zeta^{t,\tau}$ represents the loss if the prediction is at least ϵ larger than the true value whereas $\zeta^{*t,\tau}$ represents the loss if the prediction is at least ϵ lower than the true value. Notice that when $\epsilon = 0$, it reduces to the standard ℓ_1 loss function.

In addition to predicting the real-valued intensity, our goal is to accurately predict the ordinal category of the hurricane. In practice, predicting the category is more useful to communicate the destructive power of a hurricane to policymakers and the public. As noted in the introduction, intensity prediction alone is insufficient since it ignores how significant the prediction error affects its predicted category. By adding an ordinal loss, we hope the model will focus more on data points located near the boundary between two ordinal

categories. To achieve this, we introduce the following ordinal loss constraints:

$$\begin{aligned} \mathcal{L} = \quad & \zeta^{t,\tau} + \zeta^{*t,\tau} \\ \text{s.t.} \quad & z^{t,\tau} - b_{\hat{y}^{t,\tau}} \leq -1 + \zeta^{t,\tau} \\ & z^{t,\tau} - b_{\hat{y}^{t,\tau}-1} \geq 1 - \zeta^{*t,\tau} \\ & \zeta^{t,\tau} \geq 0, \zeta^{*t,\tau} \geq 0 \end{aligned} \tag{4.7}$$

where $[b_{\hat{y}^{t,\tau}-1}, b_{\hat{y}^{t,\tau}}]$ denote the range of the ordinal category $\hat{y}^{t,\tau}$. Here, $\zeta^{t,\tau}$ represents the loss if the prediction lies at the larger category whereas $\zeta^{*t,\tau}$ represents the loss if the prediction is in the lower category.

Our joint real-valued intensity and ordinal category prediction is obtained by combining Equations (4.6) and (4.7):

$$\begin{aligned} \mathcal{L} = \quad & \zeta^{t,\tau} + \zeta^{*t,\tau} \\ \text{s.t.} \quad & z^{t,\tau} - b_{\hat{y}^{t,\tau}} \leq -1 + \zeta^{t,\tau} \\ & z^{t,\tau} - b_{\hat{y}^{t,\tau}-1} \geq 1 - \zeta^{*t,\tau} \\ & z^{t,\tau} - y^{t,\tau} \leq \epsilon + \zeta^{t,\tau} \\ & z^{t,\tau} - y^{t,\tau} \geq -\epsilon - \zeta^{*t,\tau} \\ & \zeta^{t,\tau} \geq 0, \zeta^{*t,\tau} \geq 0 \end{aligned} \tag{4.8}$$

Since their predictions are correlated, we expect the model to produce better results when they are jointly learned.

Next, we extend the preceding formulation to an online, multi-lead time forecasting setting. To do this, we decompose the weight vector $\mathbf{w}^{t,\tau}$ as follows:

$$\begin{aligned} & \mathbf{w}^{t,\tau} = \mathbf{u}^t + \mathbf{v}^{t,\tau} \\ \text{s.t.} \quad & \mathbf{1}_{m_t}^T \mathbf{u}^t = 0, \mathbf{1}_{m_t}^T \mathbf{v}^{t,\tau} = 0 \end{aligned} \tag{4.9}$$

where $\mathbf{u}^t$ is a the shared weight vector for all lead times and $\mathbf{v}^{t,\tau}$ is the lead time adjustment weight vector to improve prediction accuracy at lead time τ. Let $\mathbf{W}^t = [\mathbf{w}^{t,1}, \mathbf{w}^{t,2}, \cdots, \mathbf{w}^{t,T}]$ be the weight matrix for all lead times at time t. At each time step t, the weight matrix $\mathbf{W}^t$

is updated by minimizing the following objection function:

$$
\begin{aligned}
\mathcal{L} = \quad & \frac{1}{2}\sum_{\tau=1}^{T}\delta^{t,\tau}\gamma^{\tau}\left(\zeta^{t,\tau}+\zeta^{*t,\tau}\right)+\frac{\omega}{2}\sum_{\tau=1}^{T-1}\left\|\mathbf{w}^{t,\tau+1}-\mathbf{w}^{t,\tau}\right\|^2 \\
+ \quad & \frac{\mu}{2}\left\|\mathbf{u}^{t}-\mathbf{u}^{t-1}\right\|^2+\frac{\nu}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau}-\mathbf{v}^{t-1,\tau}\right\|^2+\frac{\eta}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau}\right\|^2 \\
\text{s.t.} \quad & \forall\ t,\tau:\ \mathbf{1}_{m_t}^{T}\mathbf{u}^{t}=1,\ \mathbf{1}_{m_t}^{T}\mathbf{v}^{t,\tau}=0, \\
& z^{t,\tau}-b_{\hat{y}^{t,\tau}}\leq -1+\zeta^{t,\tau} && \text{(4.10a)} \\
& z^{t,\tau}-b_{\hat{y}^{t,\tau}-1}\geq 1-\zeta^{*t,\tau} && \text{(4.10b)} \\
& z^{t,\tau}-y^{t,\tau}\leq \epsilon+\zeta^{t,\tau} && \text{(4.10c)} \\
& z^{t,\tau}-y^{t,\tau}\geq -\epsilon-\zeta^{*t,\tau} && \text{(4.10d)} \\
& \zeta^{t,\tau}\geq 0,\zeta^{*t,\tau}\geq 0 \\
& && \text{(4.10)}
\end{aligned}
$$

where $\delta^{t,\tau}$ is an indicator function whose value is equal to 1 if $\mathbf{x}^{t,\tau}$ and $\mathbf{y}^{t,\tau}$ values are both available; otherwise its value is 0. γ^{τ} represents the relative importance of the prediction error at lead time τ. The first term in the objective function along with the inequality constraints are analogous to the loss function defined for support vector ordinal regression [17]. The second term ensures that the estimated model parameters would vary smoothly at different lead times, thus preserving the temporal autocorrelation of the predicted intensities. The third and fourth terms ensure that the shared weight vector and lead time adjustment weights are close to their values at previous time step. Finally, the last term penalizes large values in the lead time adjustment weight vectors. The objective function is solved using standard quadratic programming solvers. Similar to `OMuLeT`, we also employed the backtracking and restart strategy when updating the model weights. This strategy is illustrated in Figure 4.3, with the pseudocode shown in Algorithm 2.

4.3.4 OOQR-ϵ Framework

Another challenge to be addressed is the skewed distribution of the hurricane intensity values as high category hurricanes tend to occur less frequently than lower category ones. Similar to OOQR framework, OOR-ϵ framework can be extended to incorporate a quantile loss function. Unlike conventional regression methods that minimize the square error loss to estimate the conditional mean of a target variable, a quantile loss estimates the conditional quantiles of the target variable.

To extend the framework, recall from Equation (4.8) that the loss for a single prediction is $\mathcal{L} = \zeta^{t,\tau} + \zeta^{*t,\tau}$, where $\zeta^{t,\tau}$ represents the loss when the prediction is larger than a threshold value while $\zeta^{*t,\tau}$ is the corresponding loss if the prediction is smaller than the threshold. By adding a quantile hyperparameter ξ, the prediction loss can be modified as $\mathcal{L} = (1-\xi)\zeta^{t,\tau} + \xi\zeta^{*t,\tau}$. The quantile hyperparameter ξ represents the importance of the two losses. When $\xi \approx 1$, predictions for higher category are encouraged as the losses from these predictions will be significantly reduced. Therefore, QR can help bias the framework towards increasing the number of high intensity predictions.

Thus, our OOQR-ϵ framework is designed to the minimize the following loss function.

$$
\begin{aligned}
\mathcal{L} = \quad & \sum_{\tau=1}^{T} \delta^{t,\tau}\gamma^{\tau}\left((1-\xi)\zeta^{t,\tau} + \xi\zeta^{*t,\tau}\right) + \frac{\omega}{2}\sum_{\tau=1}^{T-1}\left\|\mathbf{w}^{t,\tau+1} - \mathbf{w}^{t,\tau}\right\|^2 \\
+ \quad & \frac{\mu}{2}\left\|\mathbf{u}^{t} - \mathbf{u}^{t-1}\right\|^2 + \frac{\nu}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau} - \mathbf{v}^{t-1,\tau}\right\|^2 + \frac{\eta}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau}\right\|^2 \\
\text{s.t.} \quad & \forall\, t,\tau:\ \mathbf{1}_{m_t}^{T}\mathbf{u}^{t} = 1,\ \mathbf{1}_{m_t}^{T}\mathbf{v}^{t,\tau} = 0, \\
& z^{t,\tau} - b_{\hat{y}^{t,\tau}} \le -1 + \zeta^{t,\tau} \\
& z^{t,\tau} - b_{\hat{y}^{t,\tau}-1} \ge 1 - \zeta^{*t,\tau} \\
& z^{t,\tau} - y^{t,\tau} \le \epsilon + \zeta^{t,\tau} \\
& z^{t,\tau} - y^{t,\tau} \ge -\epsilon - \zeta^{*t,\tau} \\
& \zeta^{t,\tau} \ge 0, \zeta^{*t,\tau} \ge 0
\end{aligned}
\tag{4.11}
$$

where ξ is the quantile hyperparameter. Note that OOQR is equivalent to OOR when $\xi = 0.5$.

4.4 Experiments

We have performed extensive experiments to evaluate the performance of our proposed frameworks. The ground truth hurricane intensity data along with the official forecasts by the U.S. National Hurricane Center (NHC) are obtained from their website[1]. Our hurricane prediction models are trained using outputs generated by an ensemble of 21 statistical and dynamical models obtained from the Hurricane Forecast Model Output website[2]. Note that the ensemble members used as predictors in our framework are a subset of the statistical and dynamical models used by NHC to generate their official forecasts. We collected 6-hourly hurricane intensity data spanning the years 2012 to 2020, which contains 336 tropical cyclones. Each tropical cyclone has an average length of 21.9 time steps (data points), which gives a total of 7,364 data points. The data from 2012 to 2017 (208 tropical cyclones) are used for training and validation, while those from 2018 to 2020 (128 tropical cyclones) are used for testing.

4.4.1 Baseline and Evaluation Metrics

We compared our proposed frameworks against the following baseline methods:

1. **Ensemble mean**: This method simply computes the average value of the ensemble member outputs at each lead time as its predictions.

2. **Persistence**: This method assumes the intensity at each time step is equal to intensity at its previous time step.

3. **Passive-Aggressive(PA)** [21]: This is a popular online learning algorithm that updates the weights based on newly observed data points.

4. **ORION** [84]: This is an online multi-task learning algorithm for multi-lead time forecasting.

[1] https://www.nhc.noaa.gov
[2] http://derecho.math.uwm.edu/models

5. **OMuLeT**: This is the online learning algorithm described in the previous chapter for trajectory prediction. It can be applied to the intensity prediction as well using the ensemble member forecasts as predictors.

6. **NHC**: This corresponds to the official forecasts generated by NHC.

For a fair comparison, all the online methods apply the backtracking and restart strategy to update their weights. Hyperparameters for the methods were tuned by minimizing the following macro-averaged mean absolute error (MAE) on the validation set.

$$\text{macro-MAE} = \frac{1}{6}\sum_{i=0}^{5}\left(\frac{1}{N_i}\sum_{\hat{y}^{t,\tau}=i}\left|\hat{z}^{t,\tau}-\hat{y}^{t,\tau}\right|\right) \tag{4.12}$$

where N_i is the number of data points of category i in the validation set. We use MAE on the test data to evaluate the error in both real- and ordinal-valued predictions. We also use F1-score to evaluate accuracy of the ordinal category predictions.

4.4.2 Experimental Results for `OOR`/`OOQR`

4.4.2.1 Performance Comparison

Table 4.2 shows the MAE values for lead times 12 hours to 48 hours. The results suggest that `OOR` clearly outperformed the other baseline methods for all lead times. Table 4.3 summarizes the F1-scores of the different approaches over all lead times. Several interesting conclusions can be drawn from the result. First, the F1-score for high category is generally lower than the F1-score for low category, which means predicting high category is more challenging. As a consequence, micro-F1 is higher than macro-F1 for all methods since there are more data points with lower categories than higher ones. Second, ensemble mean has very low macro-F1 score due to its poor predictive performance on high category hurricanes. Third, `OMuLeT`, `ORION`, and `PA` also performed poorly since they are not designed for ordinal data, unlike `OOR`. Fourth, `OOR` is consistently among the best models for predicting high category hurricanes. Finally, the NHC official forecasts outperform all the competing methods,

which is not surprising as the NHC official forecasts were generated from a larger suite of dynamical and statistical models as well as other data sources and human experts beyond the resources available to us. Despite this, our `OOR` framework clearly outperformed all the baseline methods and achieve macro-F1 and MAE values that are comparable to those for NHC.

	MAE			
Lead Time (hrs)	12	24	36	48
Ensemble Mean	0.217	0.303	0.374	0.438
Persistence	0.239	0.451	0.625	0.787
PA	0.226	0.394	0.533	0.643
ORION	0.626	0.682	0.712	0.751
OMuLeT	0.508	0.582	0.647	0.699
OOR	**0.158**	**0.265**	**0.342**	**0.387**
NHC	0.144	0.249	0.310	0.380

Table 4.2: Comparison of MAE for various hurricane category forecasting methods at different lead times.

	F1-score for hurricane categories						
Category	macro-F1	0	1	2	3	4	5
Ensemble Mean	0.390	0.917	0.449	0.309	0.305	0.168	0.190
Persistence	0.396	0.859	0.327	0.247	0.234	0.330	0.378
PA	0.395	0.879	0.411	0.293	0.243	0.300	0.241
ORION	0.382	0.627	0.243	0.301	0.356	0.350	0.412
OMuLeT	0.408	0.594	0.349	0.359	0.336	**0.404**	0.403
OOR	**0.492**	**0.920**	**0.495**	**0.364**	**0.373**	0.374	**0.425**
NHC	0.540	0.924	0.554	0.411	0.390	0.462	0.502

Table 4.3: Comparison of F1-score for various hurricane category forecasting in different categories.

To obtain a more detailed performance comparison, we examined the confusion matrices generated by the various baseline algorithms in Figure 4.4. Based on these figures, it is obvious that the hurricane category data is skewed since there are much more lower category hurricanes than higher category ones. Yet, it is more important to make accurate predictions on the higher category hurricanes, since these hurricanes are more powerful and can cause more severe damages. The results suggest that the persistence approach performs the worst

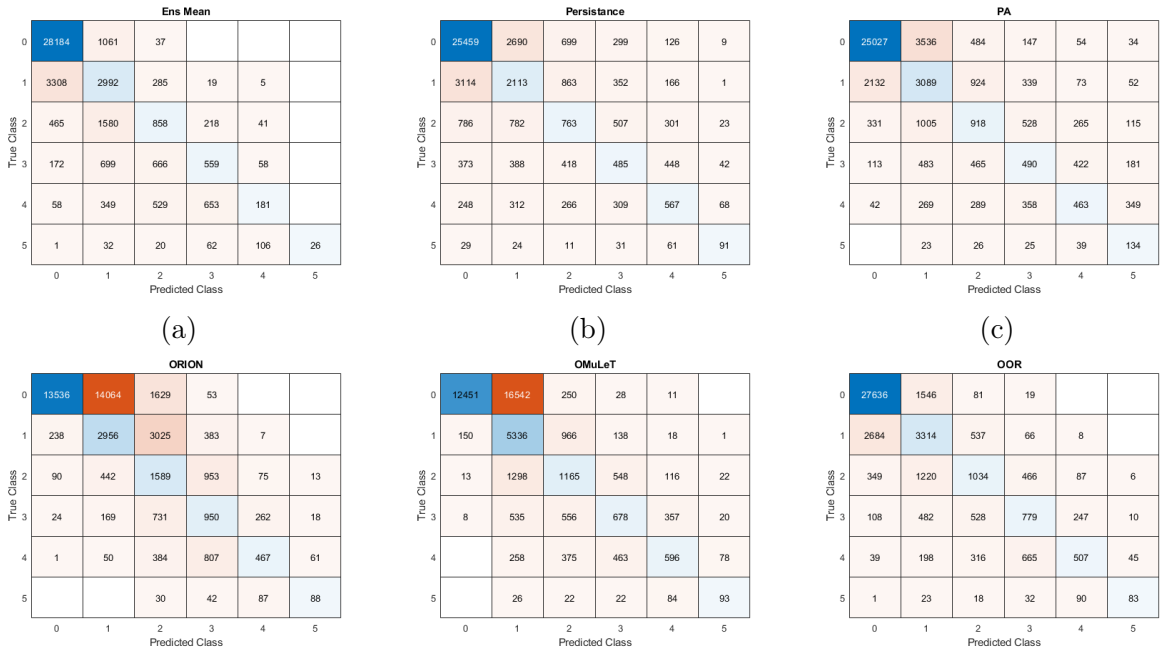


Figure 4.4: Comparison of the confusion matrices for hurricane category predictions from multiple baseline algorithms.

among all the baseline methods. This is not surprising since it simply assumes that the hurricane category does not change over time. The PA algorithm did a much better job in terms of accurately predicting the high category hurricanes, but it also produces a large number of false alarms. The ensemble mean approach can only identify a small number of high category hurricanes. This is because the approach assumes each model is equally skilful, without ruling out the effect of bad models. While ORION and `OMuLeT` algorithms did a much better job compared to other baselines, they still produce a large number of incorrect predictions. Overall, `OOR` outperformed other baselines by balancing the trade-off between true positives and false alarms. As a result, `OOR` has higher F1 scores for most of the hurricane categories compared to the other baselines.

Figure 4.5 shows how the shared weight vector $\mathbf{u}^t$ of the ensemble members changes over time using `OOR`. The plot suggests that some models, such as SHIP and DSHP, generally have higher weights than others. Some models also become increasingly skillful in recent years, such as CTCX. The plot provides evidence to support the need for applying online

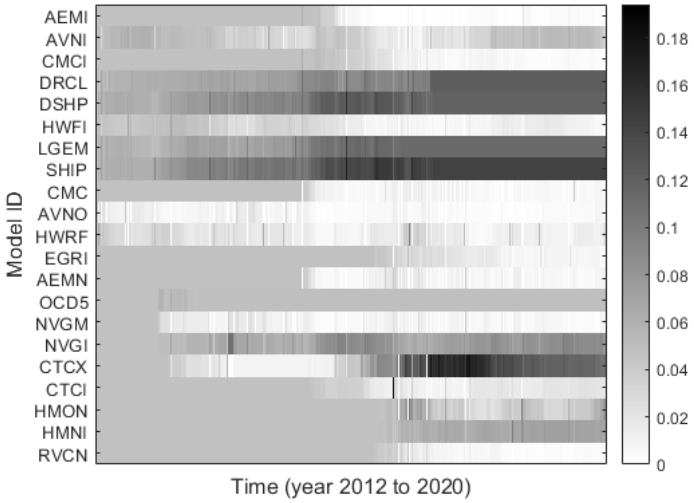


Figure 4.5: Weight for the model learned from OOR over hurricanes from year 2012 to 2020.

learning to hurricane intensity prediction in order to continuously adapt the model to the varying skills of the ensemble members.

To investigate the effect of using quantile loss, we calculated the precision and recall of OOQR predictions for different values of ξ. The results are presented in Table 4.4. Observe that increasing the hyperparameter ξ generally leads to smaller precision but larger recall for higher category hurricanes. This validates our rationale for using quantile loss to help forecast high category hurricanes more accurately. To demonstrate how the hyperparameter ξ affects the predictions, Figure 4.6 illustrates the distribution of predicted categories for the OOQR framework using different hyperparameter values, ξ. For brevity, we only show the results for 48-hour lead time forecasts generated by OOQR from the year 2012 to 2020. It is clear that the predictions generated by the OOQR framework tend to be higher categories with larger hyperparameter ξ. More details can be found in the confusion matrices for OOQR with different hyperparameter ξ shown in Figure 4.7. As we increase the hyperparameter ξ from 0.5 to 0.9, the number of accurately predicted hurricanes increases for both category 3 and category 4 hurricanes. For category 3 hurricanes, the number increases from 779 to 790. For category 4 hurricanes, the number increases from 507 to 544. For category 5 hurricanes,

the number decreases slightly from 83 to 82. Nevertheless, the results suggest that the overall performance of our proposed framework improves for high category predictions when utilizing the quantile loss function.

	Precision						
Category	macro-Precision	0	1	2	3	4	5
OOQR(ξ=0.5)	**0.550**	0.897	**0.489**	0.411	**0.384**	**0.540**	**0.576**
OOQR(ξ=0.6)	0.548	0.897	0.488	0.409	0.383	0.536	**0.576**
OOQR(ξ=0.7)	0.544	0.898	**0.489**	0.409	0.380	0.532	0.557
OOQR(ξ=0.8)	0.544	0.898	0.488	0.411	0.379	0.532	0.554
OOQR(ξ=0.9)	0.541	**0.899**	0.488	**0.412**	0.378	0.527	0.539
	Recall						
Category	macro-Recall	0	1	2	3	4	5
OOQR(ξ=0.5)	0.459	**0.944**	0.501	**0.327**	0.362	0.286	**0.336**
OOQR(ξ=0.6)	0.460	0.943	0.501	0.324	0.364	0.290	**0.336**
OOQR(ξ=0.7)	0.460	0.942	0.503	0.325	0.361	0.295	**0.336**
OOQR(ξ=0.8)	0.461	0.941	0.504	0.325	0.364	0.301	0.332
OOQR(ξ=0.9)	**0.463**	0.939	**0.505**	**0.327**	**0.367**	**0.307**	0.332
	F1-score						
Category	macro-F1	0	1	2	3	4	5
OOQR(ξ=0.5)	**0.492**	**0.920**	0.495	0.364	0.373	0.374	**0.425**
OOQR(ξ=0.6)	**0.492**	0.919	0.494	0.362	**0.374**	0.377	**0.425**
OOQR(ξ=0.7)	0.491	0.919	**0.496**	0.362	0.370	0.380	0.419
OOQR(ξ=0.8)	**0.492**	0.919	**0.496**	0.363	0.371	0.385	0.415
OOQR(ξ=0.9)	**0.492**	0.919	**0.496**	**0.365**	0.372	**0.388**	0.411

Table 4.4: Comparison of precision, recall, and F1-score for OOQR with different hyperparameter ξ.

4.4.2.2 Case Study

Figure 4.8 shows an example of the category forecasts for Hurricane Dorian from 2019/08/28 to 2019/08/29. During this period, the hurricane Dorian is rapidly intensified to category 5. The error bars represent the range of category forecasts from ensemble members. The results showed that the ensemble member outputs have a large variance in their forecasts, which implies that the forecast models are not equally skilful. Figure 4.8 demonstrates that OOR/OOQR's 48-hour forecasts are closer to the NHC official forecasts compared to the baseline methods. OOQR with hyperparameter $\xi = 0.9$ would be preferred for forecasts with

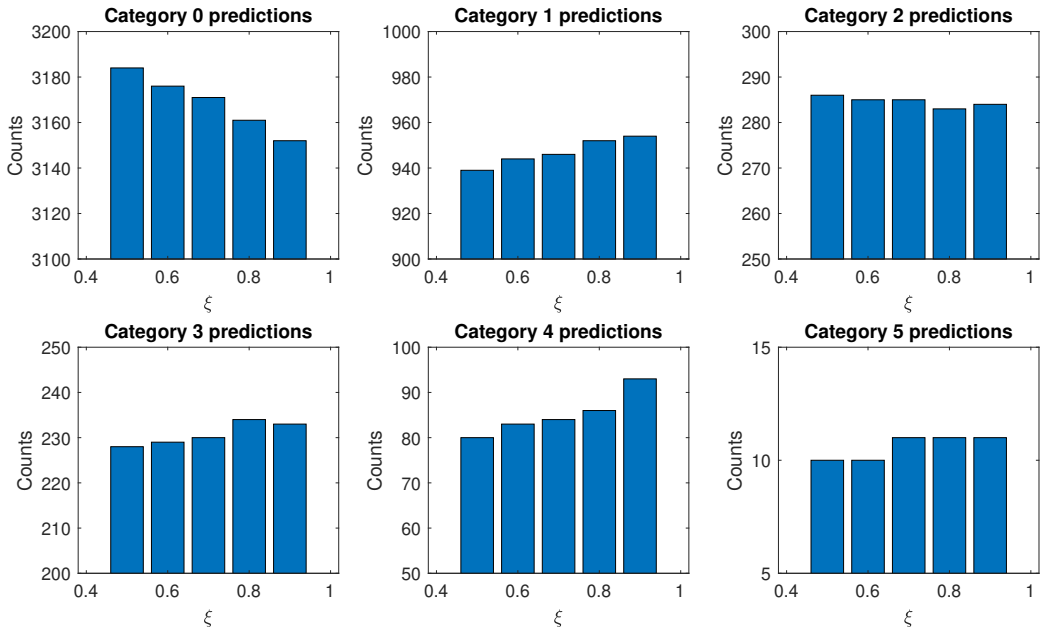


Figure 4.6: The figure shows the prediction difference for OOQR framework with different hyperparameter ξ. The results considered the 48-hour forecasts generated by OOQR framework for the hurricanes from year 2012 to year 2020.

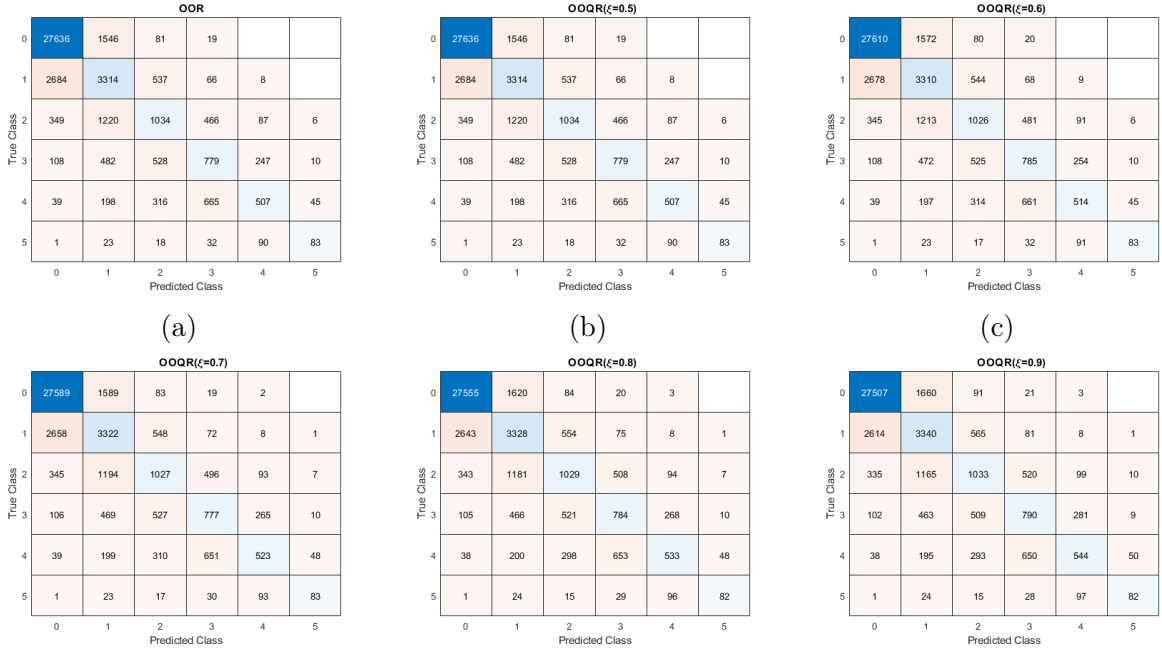


Figure 4.7: Comparison of the confusion matrices for hurricane category predictions from multiple baseline algorithms.

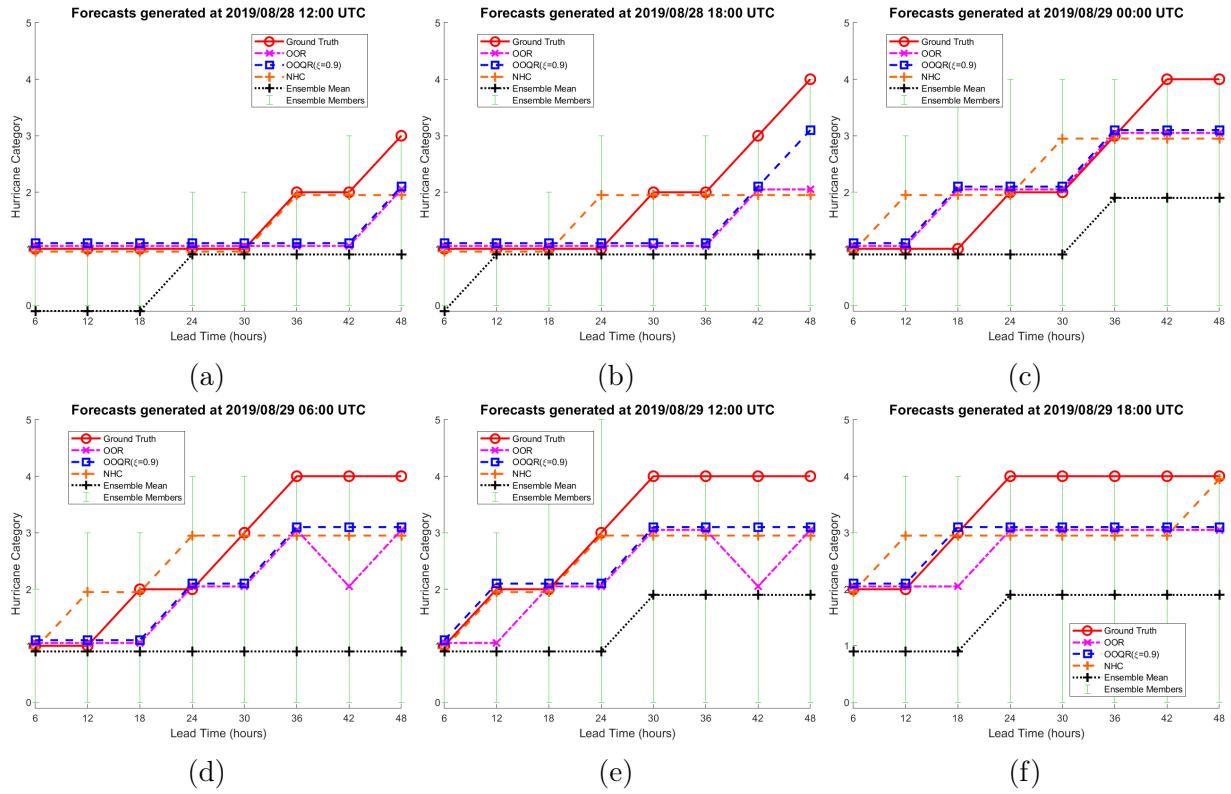


Figure 4.8: Comparison of 48-hour forecasts for Hurricane Dorian from 2019/08/28 to 2019/08/29 by different methods. The error bars represent the range of category forecasts from ensemble members.

higher categories. Figure 4.8(b)(d)(e)(f) clearly show that `OOQR` generate more accurate high category predictions than `OOR`. Although the predictions of the ensemble members vary greatly, `OOR`/ `OOQR` were able to assign the appropriate set of weights to the ensemble members to make its prediction more accurate.

4.4.3 Experimental Results for `OOR-`ϵ/`OOQR-`ϵ

4.4.3.1 Performance Comparison

Table 4.5 summarizes the intensity MAE and category MAE for various hurricane intensity forecasting methods at different lead times. There are several interesting observations from the results shown in the table. First, the persistence method performs poorly compared to other methods, which is not surprising as it simply assumes that the intensity remains

unchanged. Thus, its MAE becomes worse with increasing lead time. Second, PA is better than the ensemble mean at shorter lead times but is worse at longer lead times, which makes sense since PA does not consider correlations between the multi-lead time predictions. Third, ORION and `OMuLeT` outperforms PA, ensemble mean, and persistence since they consider the correlation between the multi-lead time predictions. Nevertheless, our proposed framework `OOR-`ϵ outperforms all these baseline methods especially at longer lead times. Its intensity prediction is comparable to NHC official results (better in terms of intensity MAE but slightly worse in terms of category MAE). This result is achieved despite the fact that `OOR-`ϵ uses only a small subset of the models used by NHC to generate their official forecasts.

	MAE (intensity in kt)				MAE (category)			
Lead Time (hrs)	12	24	36	48	12	24	36	48
Ensemble Mean	6.742	8.692	9.899	11.036	0.217	0.303	0.374	0.438
Persistence	7.717	13.797	18.246	22.102	0.239	0.451	0.625	0.787
PA	6.042	8.868	10.245	11.917	0.181	0.298	0.372	0.452
ORION	5.792	7.941	9.388	10.551	0.171	0.254	0.338	0.411
OMuLeT	**5.632**	7.923	9.285	10.513	**0.160**	0.262	0.338	0.403
OOR-ϵ	5.700	**7.654**	**8.880**	**9.840**	**0.160**	**0.251**	**0.329**	**0.375**
NHC	5.019	7.730	8.959	10.140	0.144	0.249	0.310	0.380

Table 4.5: Comparison of the intensity and category MAE for various hurricane intensity forecasting methods at different lead times.

We also compute the precision, recall, and F1-score of the different methods to assess how well they perform in terms of predicting the different categories. The results are shown in Table 4.6. Observe that ensemble mean performs poorly especially for high category predictions. This suggests the need for an online learning approach that considers the forecast skills of the individual ensemble members. PA is better than ensemble mean in terms of its macro-F1 but is still much worse than other online learning approaches. While ORION seems to have relatively high F1-scores, this is due to their higher recall values at the expense of their low precision values. In other words, ORION may potentially generate a large number of false alarms, which makes it ineffective as an early warning system. Finally, our `OOR-`ϵ framework outperforms all the baselines in terms of its overall precision and F1-score, especially for

	F1-score						
Category	macro-F1	0	1	2	3	4	5
Ensemble Mean	0.390	0.917	0.449	0.309	0.305	0.168	0.190
Persistence	0.396	0.859	0.327	0.247	0.234	0.330	0.378
PA	0.453	0.912	0.487	0.323	0.321	0.362	0.313
ORION	0.491	0.922	**0.530**	**0.377**	0.349	**0.418**	0.350
OMuLeT	0.494	**0.923**	0.502	0.367	0.386	0.380	0.404
OOR-ϵ	**0.498**	**0.923**	0.499	0.361	**0.390**	0.372	**0.444**
NHC	0.540	0.924	0.554	0.411	0.390	0.462	0.502
	Precision						
Category	macro-Precision	0	1	2	3	4	5
Ensemble Mean	0.585	0.876	0.446	0.358	0.370	0.463	**1.000**
Persistence	0.402	0.848	0.335	0.253	0.245	0.340	0.389
PA	0.449	**0.924**	0.456	0.328	0.305	0.449	0.230
ORION	0.497	0.917	**0.516**	0.410	0.340	0.488	0.313
OMuLeT	0.552	0.896	0.502	**0.422**	0.398	0.555	0.537
OOR-ϵ	**0.590**	0.896	0.492	0.416	**0.407**	**0.578**	0.750
NHC	0.579	0.924	0.517	0.448	0.386	0.556	0.642
	Recall						
Category	macro-Recall	0	1	2	3	4	5
Ensemble Mean	0.359	**0.963**	0.453	0.271	0.260	0.102	0.105
Persistence	0.391	0.869	0.320	0.241	0.225	0.320	0.368
PA	0.479	0.899	0.522	0.318	0.339	0.303	**0.490**
ORION	**0.490**	0.928	**0.544**	**0.349**	0.359	**0.366**	0.397
OMuLeT	0.461	0.951	0.502	0.324	**0.374**	0.289	0.324
OOR-ϵ	0.457	0.951	0.507	0.319	**0.374**	0.275	0.316
NHC	0.517	0.925	0.596	0.379	0.393	0.395	0.413

Table 4.6: Comparison of F1-score, precision and recall for various hurricane intensity forecasting methods at different categories.

large categories. Although its precision for large categories is high, its recall is relatively low. The OOQR-ϵ framework is designed to overcome this limitation.

To investigate the effect of using quantile loss, Table 4.7 shows the precision, recall, and F1-score for OOQR-ϵ at different values of ξ for the different categories. The results validate our assumption that increasing ξ will bias the framework towards predicting more high category hurricanes, as evidenced by the increasing recall but decreasing the precision values. Note that the result at $\xi = 0.5$ is equivalent to that for OOR-ϵ. Figure 4.9 plots the 48-hour forecast distributions of OOQR-ϵ for different values of ξ, which shows the increasing number of high category predictions when ξ is larger. The F1-score of OOQR-ϵ at $\xi = 0.9$

is also slightly higher than that for OOR-ϵ, which suggests the tradeoff between precision and recall. Figure 4.10 shows the QQ-plots for 48-hour lead time forecasts generated by the different methods. There are several interesting observations can be made from these plots. Once again, persistence gives the worst result with many of its predicted values deviate from the ground truth. This is to be expected since the persistence method does not perform any learning from the data. Second, the predicted intensity values generated by the ensemble mean are generally lower than the ground truth for high intensity hurricanes. Third, the predictions generated by the PA algorithm are often much larger than the ground truth. The predicted distributions for ORION, OOR-ϵ and OOQR-ϵ looked quite similar. A detailed comparison between OOR-ϵ and OOQR-ϵ to the ground truth is shown in Figure 4.11. As expected, when the hyperparameter ξ increases, the model is more biased towards predicting higher intensity values, as evidenced by the upward shift in the distribution of the blue points (for OOQR-ϵ) compared to the red points (for OOR-ϵ) in the diagram.

	F1-score						
Category	macro-F1	0	1	2	3	4	5
OOQR-ϵ(ξ=0.5)	0.498	**0.923**	0.499	0.361	**0.390**	0.372	0.444
OOQR-ϵ(ξ=0.6)	0.500	**0.923**	0.502	**0.362**	**0.390**	0.378	0.445
OOQR-ϵ(ξ=0.7)	0.500	0.922	0.501	0.356	0.388	0.382	**0.448**
OOQR-ϵ(ξ=0.8)	0.500	0.922	**0.504**	0.357	0.385	0.385	**0.448**
OOQR-ϵ(ξ=0.9)	**0.501**	0.921	0.503	0.358	0.383	**0.393**	0.446
	Precision						
Category	macro-Precision	0	1	2	3	4	5
OOQR-ϵ(ξ=0.5)	**0.590**	0.896	0.492	**0.416**	**0.407**	**0.578**	**0.750**
OOQR-ϵ(ξ=0.6)	0.579	0.898	**0.494**	0.415	0.402	0.571	0.692
OOQR-ϵ(ξ=0.7)	0.569	0.900	0.492	0.407	0.396	0.563	0.656
OOQR-ϵ(ξ=0.8)	0.562	0.901	**0.494**	0.405	0.390	0.555	0.628
OOQR-ϵ(ξ=0.9)	0.559	**0.902**	0.492	0.403	0.388	0.549	0.619
	Recall						
Category	macro-Recall	0	1	2	3	4	5
OOQR-ϵ(ξ=0.5)	0.457	**0.951**	0.507	0.319	0.374	0.275	0.316
OOQR-ϵ(ξ=0.6)	0.461	0.949	0.510	0.321	0.378	0.282	0.328
OOQR-ϵ(ξ=0.7)	0.464	0.946	0.511	0.316	**0.381**	0.289	0.340
OOQR-ϵ(ξ=0.8)	0.467	0.944	0.514	0.320	0.379	0.294	**0.348**
OOQR-ϵ(ξ=0.9)	**0.469**	0.941	**0.516**	**0.322**	0.378	**0.306**	**0.348**

Table 4.7: Comparison of F1-score, precision, and recall for OOQR-ϵ with different values of ξ.

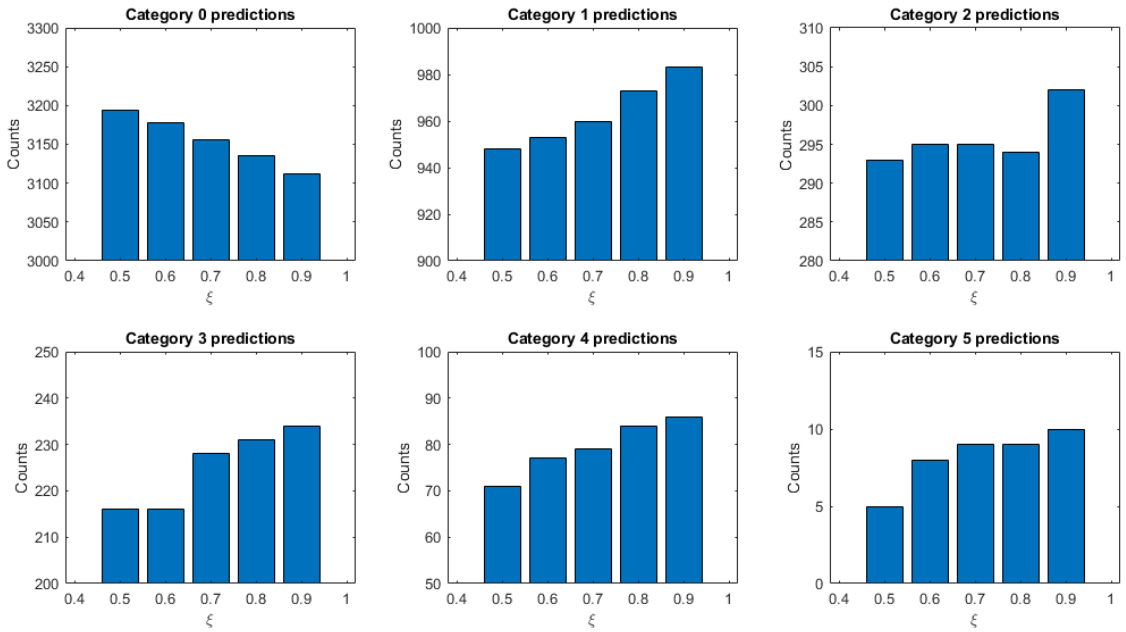
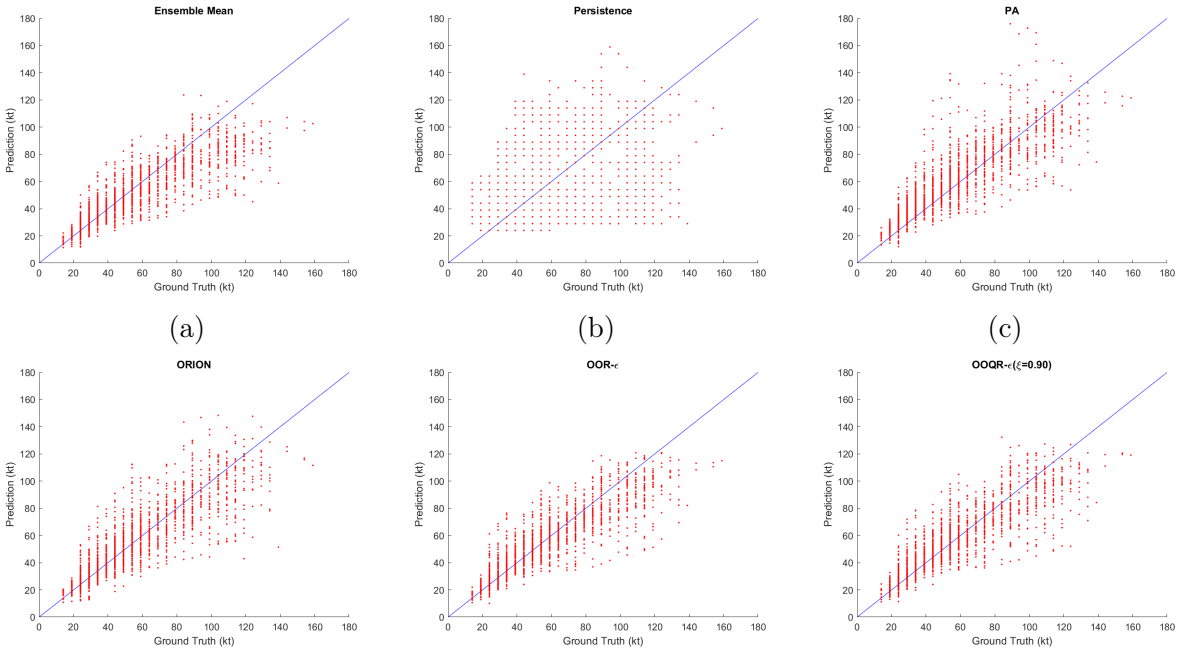


Figure 4.9: 48-hour forecast distribution of OOQR-ϵ for different ξ.



(a) (b) (c)

(d) (e) (f)

Figure 4.10: Comparison of QQ plot for 48-hour forecasts with different methods.

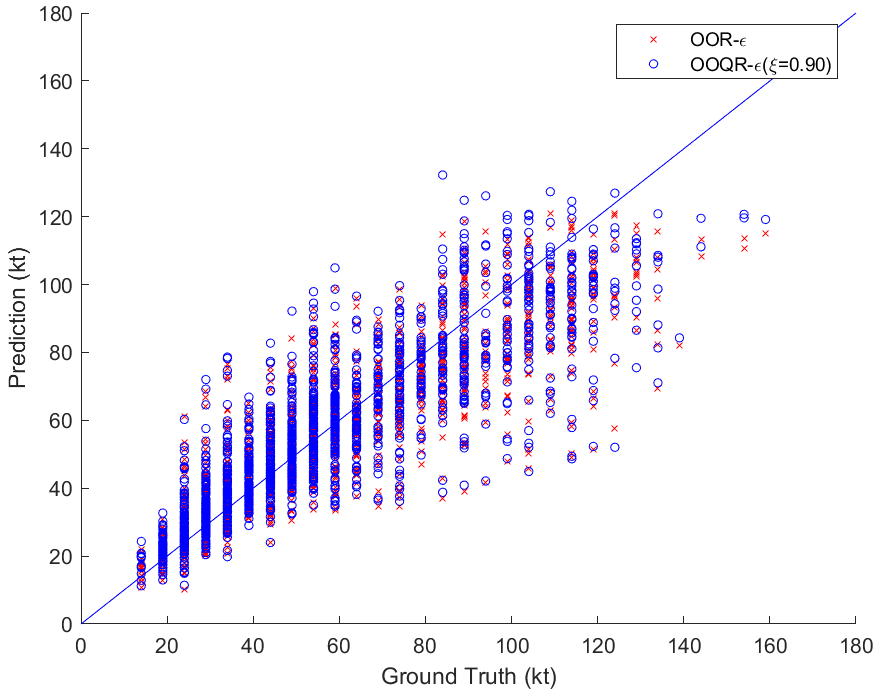


Figure 4.11: Comparison of QQ plot for OOR-ϵ and OOQR-ϵ.

4.4.4 Ablation Study

Our OOR-ϵ framework uses an ϵ-insensitive loss function with ordinal constraints to generate its predictions. To investigate the advantages of combining intensity with category information, we consider two variations of our OOR-ϵ framework. OOR-I removes the ϵ-insensitive constraints given by the Inequalities in (4.10c) and (4.10d) when optimizing Eqn. (4.10) whereas OOR-II ignores the ordinal constraints given by the inequalities in (4.10a) and (4.10b).

Table 4.8 summarizes the intensity and category MAE for the two variations of OOR-ϵ. OOR-I has the worst performance in terms of both its intensity and category predictions. This is not surprising because OOR-I utilizes only the coarser category information while ignoring the intensity values. OOR-ϵ generally outperforms both variations especially at longer lead times, which suggests the benefits of using both real-valued intensity and its ordinal category information.

	MAE (intensity in kt)				MAE (category)			
Lead Time (hrs)	12	24	36	48	12	24	36	48
OOR-ϵ	5.700	**7.654**	**8.880**	**9.840**	0.160	**0.251**	**0.329**	**0.375**
OOR-I	5.767	8.030	9.411	10.520	**0.158**	0.265	0.342	0.387
OOR-II	**5.669**	7.819	9.072	10.119	0.163	0.259	0.333	0.379

Table 4.8: Comparison of intensity and category MAE for OOR-ϵ variations at different lead times.

	F1-score						
Category	macro-F1	0	1	2	3	4	5
OOR-ϵ	**0.498**	**0.923**	0.499	0.361	**0.390**	0.372	**0.444**
OOR-I	0.492	0.920	0.495	0.364	0.373	**0.374**	0.425
OOR-II	0.495	**0.923**	**0.500**	**0.377**	**0.390**	0.363	0.415
	Precision						
Category	macro-Precision	0	1	2	3	4	5
OOR-ϵ	**0.590**	0.896	0.492	0.416	0.407	**0.578**	**0.750**
OOR-I	0.550	**0.897**	0.489	0.411	0.384	0.540	0.576
OOR-II	0.585	0.895	**0.498**	**0.427**	**0.410**	0.561	0.720
	Recall						
Category	macro-Recall	0	1	2	3	4	5
OOR-ϵ	0.457	0.951	**0.507**	0.319	**0.374**	0.275	0.316
OOR-I	**0.459**	0.944	0.501	0.327	0.362	**0.286**	**0.336**
OOR-II	0.454	**0.952**	0.503	**0.337**	0.372	0.268	0.291

Table 4.9: Comparison of F1-score, precision and recall for OOR-ϵ variations at different categories.

Table 4.9 compares the precision, recall, and F1-score for the different methods. Unlike MAE, the F1-scores for both OOR-I and OOR-II are close, which suggests that the category constraints (without ϵ-insensitive loss) still contain enough useful information for hurricane category predictions. The results of OOR-I is better for higher categories, suggesting that the ordinal constraints is beneficial for high category predictions. OOR-ϵ also demonstrates good performance in high categories, with slightly better overall performance.

4.4.4.1 Case Study

Figure 4.12 shows an example of the forecasts for Hurricane Dorian from August 28 to 29, 2019. During this period, the hurricane rapidly intensified to category 5. The green error

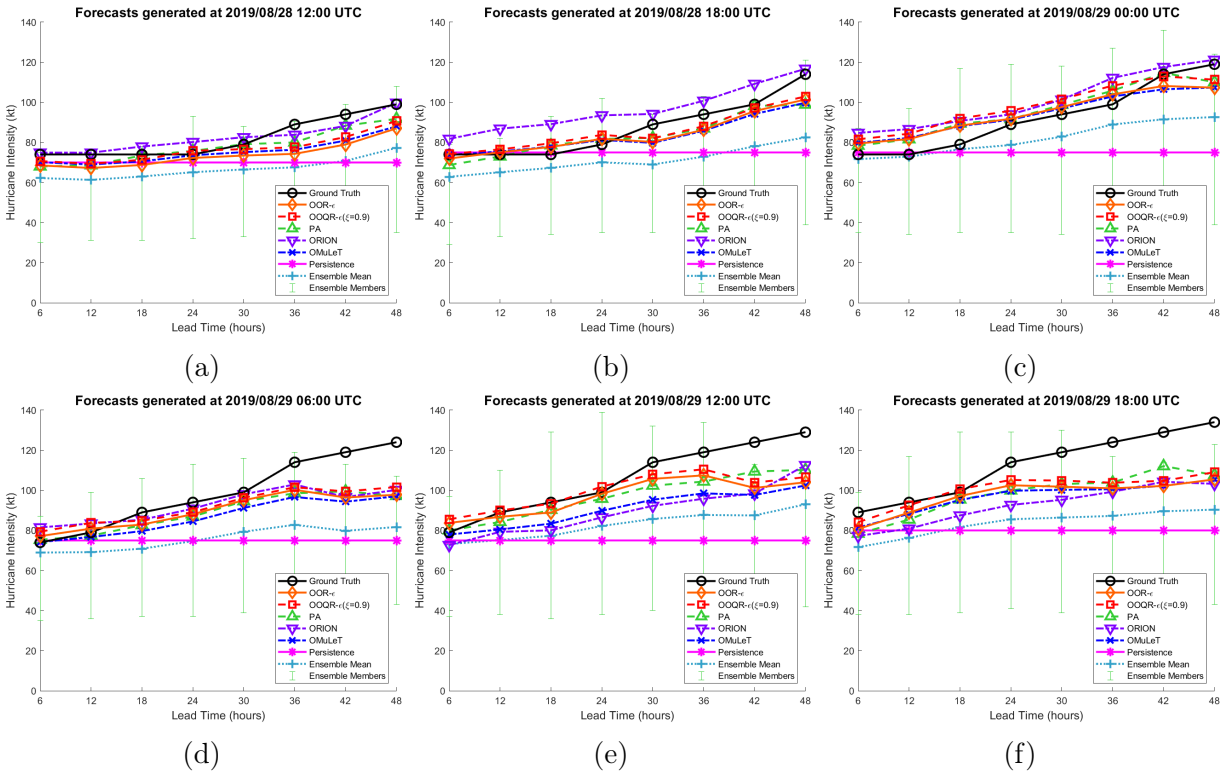


Figure 4.12: 6 to 48 hour forecasts of Hurricane Dorian from August 28 to 29, 2019. The error bars represent the range of intensity predictions from the ensemble members.

bars represent the range of forecasts generated by the ensemble members. Despite the large variance in the ensemble member outputs, OOR-ϵ and OOQR-ϵ's multi-lead time forecasts were closer to the ground truth compared to other the baseline methods. Nevertheless, Figure 4.12(d)(e)(f) show that none of the ensemble members could correctly infer the rapid intensification 48-hours ahead of time, which is not surprising since predicting rapid intensification at longer lead time is a hard problem, This suggests a potential limitation of using the ensemble member forecasts; better input features are therefore needed for predicting rapid intensification at longer lead times. Finally, PA and ORION were able to generate high category predictions since they do not impose the constraints that the sum of weights must be equal to one. However, this leads to larger prediction errors at other times.

4.5 Conclusions

In this chapter, I present two novel frameworks called OOR and OOR-ϵ for online multi-lead time hurricane intensity predictions. These frameworks are designed to address research question **RQ2**, which is to handle hurricane prediction with an ordinal-valued target variable. While OOR utilized only the hurricane categories information to update the model, OOR-ϵ considered the real-valued hurricane intensity information to further improve its predictive performance. Experimental results showed that OOR/OOR-ϵ outperforms various state-of-the-art online learning methods and can generate predictions close to the NHC official forecasts. In addition, to address the research question **RQ3**, both frameworks were extended to accommodate a quantile loss function to further improve its prediction accuracy for high category hurricanes. Experimental results showed that OOQR/OOQR-ϵ can both improve the model performance in high category hurricane predictions.

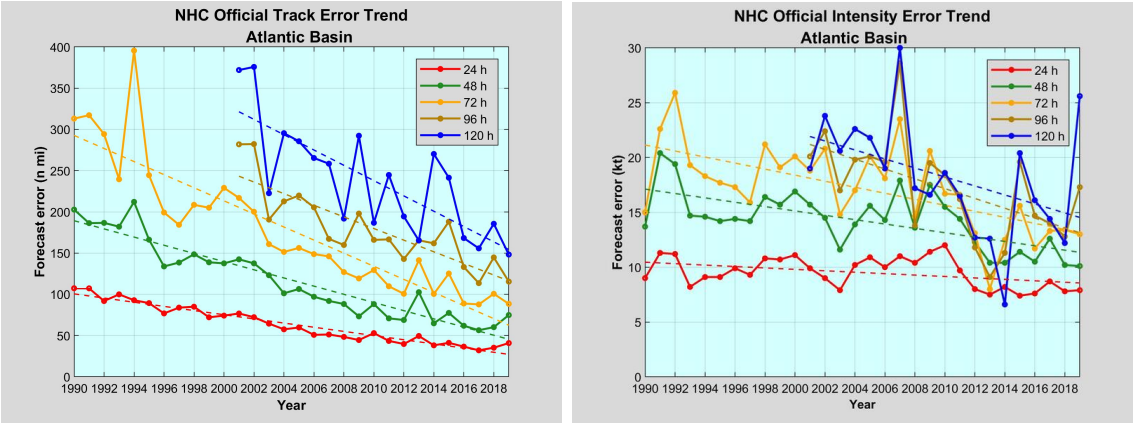
CHAPTER 5

ONLINE JOINT PREDICTION OF TRAJECTORY LOCATION AND STATE

In the previous two chapters, I have developed online learning algorithms for predicting the trajectory path and state of a moving object separately. Since the tasks are often quite related, this begs the question whether it is possible to leverage information from the trajectory path to improve state prediction, and vice-versa. For example, the state of a moving vehicle could be the driver's driving proficiency and aggressiveness. The future trajectory of the moving vehicle is greatly affected by the state of the driver. Therefore, it is important to determine if the performance of two tasks can be further improved when learning them jointly. The goal of this chapter is to develop a joint prediction framework that can generate the trajectory location and state predictions simultaneously, to address the research question **RQ4** described in Chapter 1. To develop the joint learning algorithm, I will combine the proposed `OMuLeT` algorithm for trajectory location prediction with the `OOR`/`OOR-`ϵ framework for state predictions. As proof of concept, the developed joint learning framework will be applied to the hurricane prediction task similar to previous chapters.

Hurricane prediction is a notoriously hard problem due to the complex physical mechanisms governing the dynamics of a tropical cyclone, which include factors such as sea surface temperature and vertical wind shear. To address this issue, numerous physics-based models [41] have been developed over the years to improve hurricane trajectory and intensity forecasting. Figure 5.1 shows the track and intensity errors of NHC official forecasts in the past 30 years. Despite the advances in trajectory prediction, little improvements have been achieved for intensity prediction.

In recent years, there have been growing interests in applying machine learning techniques to improve the performance of hurricane prediction tasks [59, 69, 3, 82]. However, many of the existing works were developed for forecasting hurricane trajectories only, with very few



(a) (b)

Figure 5.1: Errors of NHC official forecasts at Atlantic basin basin from year 1990 to 2019 [12]. Figure (a) shows the track error. Figure (b) shows the intensity error.

of them designed to predict intensities or both. Furthermore, accurate forecasting of its ordinal category is often more important than the wind speed itself when communicating the severity of an impending hurricane to the public. Indeed, a prediction error of 60 mph may seem trifle for a category 0 tropical storm but is significant if a category 5 hurricane at 160mph was incorrectly predicted as a category 2 storm at 100mph. Furthermore, as shown in Table 5.1, current methods were mostly limited to short-range predictions (24 hours or less) using historical observations as predictors. These methods are also mostly trained in a batch learning mode, and thus, are incapable of modeling the non-stationary nature of hurricane trajectories and their intensities.

Reference	Method	Input Features	Prediction Task	Lead Time (Forecast horizon)	Learning Mode
DeMaria et al. 2005	Linear regression	Historical data	Intensity	Multi-step (72 hrs)	Batch
Moradi Kordmahalleh et al. 2016	RNN	Historical data	Trajectory	Multi-step (12 hrs)	Batch
Cox et al. 2018	Association rule	Historical data	Trajectory	Multi-step	Batch
Mudigonda et al. 2017	ConvLSTM	Atmospheric data	Trajectory	Multi-step	Batch
Gao et al. 2018	LSTM	Historical data	Trajectory	Multi-step (72 hrs)	Batch
Alemany et al. 2019	RNN	Historical data	Trajectory	Multi-step (120 hrs)	Batch
Rüttgers et al. 2019	GAN	Atmospheric image	Trajectory	Single step (6 hrs)	Batch
Kim et al. 2019	ConvLSTM	Climate data	Trajectory	Multi-step (15 hrs)	Batch
Eslami et al. 2019	CNN	Physical model outputs	Trajectory & intensity	Multi-step	Batch
Wang et al. 2020	Online linear	Physical model outputs	Trajectory	Multi-step (48 hrs)	Online
Giffard-Roisin et al. 2020	Neural network	Historical data and atmospheric image	Trajectory	Multi-step (24 hrs)	Batch

Table 5.1: Literature review of recent works on tropical cyclone prediction.

To overcome these limitations, this chapter presents a novel online learning framework called `JOHAN` (**J**oint **O**nline **H**urricane Tr**A**jectory and I**N**tensity Prediction) for long-term

forecasting (up to 48 hours) of hurricane trajectory and intensity. By using an online learning approach, our model can be efficiently updated to fit new observations while adapting to concept drifts present in the non-stationary data. `JOHAN` employs outputs from an ensemble of dynamical (physical) models such as U.S. Navy Global Environmental Model (NAVGEM) [48] and Hurricane Weather Research and Forecasting system (HWRF) [41] to generate its forecasts. These dynamical models are designed to simulate the current and future atmospheric conditions by solving a set of physical equations. However, the skills of these ensemble members (i.e., dynamical models) may vary from one hurricane to another. By training the model in an online fashion, our framework will be able to take into account the varying skills of the ensemble members over time.

There are several reasons for developing an algorithm that can predict the hurricane trajectory and intensity jointly. First, previous studies have shown the importance of using trajectory information for intensity prediction [23, 24]. As an illustration, Figure 5.2 shows the relationship between hurricane intensity and its distance to the nearest U.S. coastline using 6-hourly hurricane data between 1851 to 2020 from NHC. The plot suggests that hurricanes with higher intensities are more likely to be distributed at shorter distances to the coastline. This phenomenon has been observed in other recent studies [83]. For example, Wang and Toumi have noted that the distance at which the tropical cyclone hits its peak intensity has grown closer to the coastline, decreasing at a rate of 30km per decade. This suggests the utility of using location information from the trajectory to help improve the prediction accuracy for high category hurricanes. Furthermore, the plot also shows that most of the hurricanes lose their intensity after landfall, which is not surprising as their energy dissipates rapidly on land, causing a sharp drop in its intensity.

It is also worth noting that not all predictions are equal in importance. Accurate prediction of high category hurricanes with potential for landfall is more critical than lower category hurricanes whose projected path is heading away from the coastline. This is because hurricanes approaching landfall have potential to cause more damaging impacts to civilian

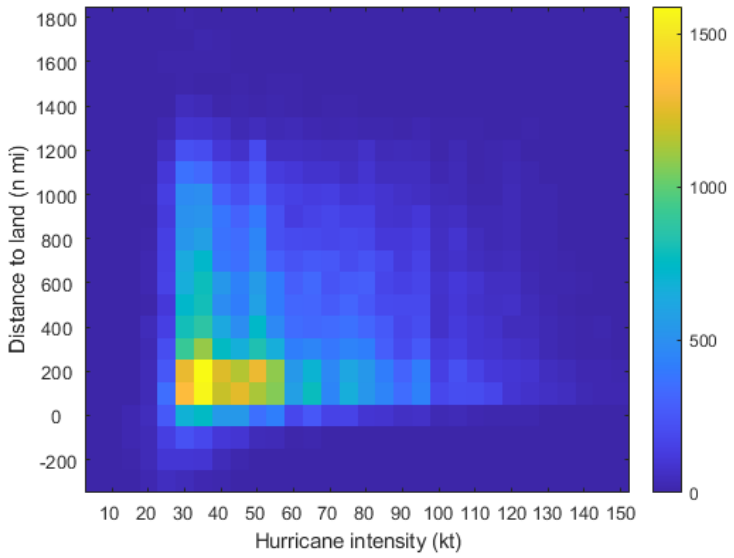


Figure 5.2: Heat map showing the relationship between hurricane intensity and its distance to nearest U.S. coastline. Negative distance indicate that the hurricane has made landfall.

population from storm surges, high winds, inland flooding, etc. Unfortunately, high category hurricanes also tend to occur less frequently than the lower category ones, which leads to a class imbalance problem. To overcome these challenges, `JOHAN` uses an exponentially-weighted quantile loss function to bias its algorithm towards predicting more accurately high intensity hurricanes that are approaching landfall.

5.1 Related Works

Due to the complexity of modeling the dynamics of tropical cyclones, there have been growing interests in developing machine learning and deep learning techniques for the hurricane prediction problem. Table 5.1 reviews some of the existing works, which can be categorized in terms of the input features used, learning approaches, target variable to be predicted, and the forecast horizon (i.e., maximum lead time).

First, existing methods typically use the historical trajectory data, climate/meteorological data, or outputs from physical models as input features for their prediction models. While historical data are more suitable for short-range predictions [69], their performance tend to be poor since they do not capture the current and future environmental conditions

that affect the hurricane's path and intensity. Methods utilizing meteorological data are usually based on deep learning techniques, such as generalized advesarial networks (GAN) [77] and convolutional LSTM (ConvLSTM) [54]. While these works are promising, their prediction errors are still relatively large since the models are typically trained using coarse-scale images (e.g., $0.5^\circ \times 0.5^\circ$). Methods that use physical model outputs tend to generate more reliable long-term forecasts since the dynamical models consider the current environmental conditions when simulating their future forecast scenarios [29, 82].

Second, most of the existing works focused on the trajectory prediction task only even though intensity forecasting is the more challenging problem. Although the regression and deep learning methods can be applied to hurricane intensity forecasting problem, they are not designed for predicting ordinal-valued categories, unlike the approach proposed in this chapter. Third, current methods also employ a batch learning approach to train their models. This may not be feasible nor effective in an operational forecast environment, when a new hurricane is continuously tracked and the model needs to be periodically updated (say every 3 to 6 hours) to reflect the new trajectory and intensity information.

Finally, recent works have focused on using deep learning and online learning approaches for hurricane trajectory prediction problems. For deep learning, [69] used sparse RNN with a flexible topology to generate hurricane trajectory predictions. [35] proposed a Long Short-Term Memory (LSTM) network to predict typhoon tracks using historical observation data from 1949 to 2011 while [3] employed RNN over a grid system to handle the non-linearity of hurricane trajectory forecasting. For online learning, [82] presented a multi-lead time forecasting framework for hurricane trajectory prediction. They showed that ensemble forecasting using outputs from physical models significantly outperform batch methods such as LSTM trained on historical trajectory data.

5.2 Problem Statement

We investigate the problem of jointly predicting the trajectory location and state of a moving object in an online learning fashion. To ground the discussion, we describe the framework in the context of hurricane prediction, though the formulation and methodology can be extended to other application domains with similar characteristics. Specifically. our goal is to design an online learning framework for joint prediction of hurricane trajectory and its intensity (both ordinal category and continuous values). At first glance, knowing the category of a hurricane does not appear to add any new information about the hurricane intensity since the former is derived from latter value (see Table 4.1). Nevertheless, the information is indeed useful as it is possible for the predicted category error to be small even though the error in predicting the maximum sustained wind speed is large. For example, given a category 5 hurricane with maximum sustained wind speed of 140 knots. A model that predicts its intensity to be 100 knots will have a lower error than one that predicts its intensity to be 200 knots; yet, the former has a larger category error (since 100 knots is a category 3 hurricane) compared to the latter, which still predicts the correct category. Furthermore, a category 2 hurricane at 95 knots predicted as 115 knots has a lower intensity prediction error compared to one predicted as 60 knots even though the former has a larger error since the category 2 cyclone is incorrectly predicted as a major category 4 storm rather than category 1, which is closer to it. Thus, leveraging both ordinal category and real-valued intensity information can help improve the prediction framework.

Consider a set of hurricanes, $\{h_1, h_2, \ldots, h_C\}$, ordered by their start times. Assuming there are n_i data points (time steps) associated with hurricane h_i, then $N = \sum_{i=1}^{C} n_i$ is the total number of time steps in the hurricane dataset. Let $\mathbb{X} = \{\mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^N\}$ be the set of trajectory forecasts generated by an ensemble of dynamical models, where each $\mathcal{X}^t$ corresponds to the hurricane trajectory forecasts generated at time step t. Similarly, the intensity forecasts generated by the ensemble members can be denoted as $\tilde{\mathbb{X}} = \{\tilde{\mathbf{X}}^1, \tilde{\mathbf{X}}^2, \ldots, \tilde{\mathbf{X}}^N\}$. Let $\tilde{n}_i$ be the accumulated number of data points from hurricane h_1 to h_i, i.e. $\tilde{n}_i = \sum_{j=1}^{i} n_j$.

Thus, $\{(\mathcal{X}^j, \tilde{\mathbf{X}}^j) \mid \tilde{n}_{i-1} < j \leq \tilde{n}_i\}$ is the set of trajectory and intensity data points associated with hurricane h_i. Assume T is the forecast horizon, i.e., maximum lead-time of the forecasting task. For each time step t, let $\mathcal{X}^t \in \mathbb{R}^{2\times m_t \times T}$ be the hurricane trajectory forecasts (latitude and longitude), where m_t is the number of ensemble member (dynamic model) forecasts available at time step t. The ensemble member trajectory forecasts for lead time τ at time t is denoted as $\mathbf{X}^{t,\tau} \in \mathbb{R}^{2\times m_t}$, with the corresponding ground truth location $\mathbf{y}^{t,\tau} \in \mathbb{R}^2$. Let $\mathbf{Y}^1, \mathbf{Y}^2, \ldots, \mathbf{Y}^N$ be the ground truth locations for all lead times at each time step t, where $\mathbf{Y}^t \in \mathbb{R}^{2\times T}$. Similarly, let $\tilde{\mathbf{X}}^t \in \mathbb{R}^{T\times \tilde{m}_t}$ be the hurricane intensity forecasts at time step t, where $\tilde{m}_t$ is the number of ensemble member forecasts available at time step t. The ensemble member forecasts for lead time τ at time t is denoted as $\tilde{\mathbf{x}}^{t,\tau} \in \mathbb{R}^{\tilde{m}_t}$, with the corresponding ground truth intensity value $\tilde{y}^{t,\tau} \in \mathbb{R}$. Let $\tilde{\mathbf{y}}^1, \tilde{\mathbf{y}}^2, \ldots, \tilde{\mathbf{y}}^N$ be the true intensity values for N time steps, where each $\tilde{\mathbf{y}}^t = [\tilde{y}^{t,1}\ \tilde{y}^{t,2} \cdots \tilde{y}^{t,T}]^T$ is a vector of intensity values for all lead times at time step t. Furthermore, let $\hat{y}^{t,\tau}$ and $\hat{\mathbf{y}}^t$ be the corresponding intensity categories associated with the real-valued intensities in $\tilde{y}^{t,\tau}$ and $\tilde{\mathbf{y}}^t$, respectively.

5.3 Methodology

At each time step t, we use the set of ensemble member forecasts for trajectory $\mathbf{X}^{t,\tau} \in \mathbb{R}^{2\times m_t}$ and intensity $\tilde{\mathbf{x}}^{t,\tau} \in \mathbb{R}^{\tilde{m}_t}$, to generate the trajectory and intensity predictions for lead time τ. The real-valued trajectory prediction $\mathbf{z}^{t,\tau} \in \mathbb{R}^2$ and intensity prediction $\tilde{z}^{t,\tau} \in \mathbb{R}$ are computed by linear predictors as follows:

$$\begin{aligned} \mathbf{z}^{t,\tau} &= f^{t,\tau}(\mathbf{X}^{t,\tau}) = \mathbf{X}^{t,\tau}\mathbf{w}^{t,\tau} \\ \tilde{z}^{t,\tau} &= \tilde{f}^{t,\tau}(\tilde{\mathbf{x}}^{t,\tau}) = \tilde{\mathbf{x}}^{t,\tau}\tilde{\mathbf{w}}^{t,\tau} \end{aligned} \tag{5.1}$$

where $\mathbf{w}^{t,\tau} \in \mathbb{R}^{m_t}, \tilde{\mathbf{w}}^{t,\tau} \in \mathbb{R}^{\tilde{m}_t}$ are the learned weight vectors associated with ensemble member forecasts for the trajectory and intensity models, respectively. The weight vectors are updated simultaneously in an online fashion whenever new observation data becomes available. One major challenge in using the ensemble member forecasts is that a significant proportion of the ensemble members may not generate any forecasts at a given time step t,

which is why the number of ensemble members, m_t, varies from one hurricane to another. This is known as the varying feature length problem [82]. To address this challenge, we applied the weight re-normalization technique proposed in Chapter 3.

5.3.1 JOHAN Framework

The novelty of JOHAN is its ability to jointly predict the hurricane trajectory and intensity. The framework consists of a pair of weight updating components, for hurricane trajectory and intensity prediction. Both components also employ an exponentially-weighted quantile loss to improve their prediction performance for close-to-land hurricanes and high category hurricanes.

To learn the tasks jointly, our framework is trained to minimize the following objective function in an online learning fashion:

$$\begin{aligned} \mathcal{L} = \quad & \mathcal{L}_{tra}(\boldsymbol{\Theta}, \xi) + \mathcal{L}_{int}(\tilde{\boldsymbol{\Theta}}, \tilde{\xi}) \\ \text{s.t.} \quad & \xi^{t,\tau} = \begin{cases} g(\tilde{y}^{t,\tau}), \text{if } \tilde{y}^{t,\tau} \text{ is available} \\ g(\tilde{z}^{t,\tau}), \text{otherwise} \end{cases} \\ & \tilde{\xi}^{t,\tau} = \begin{cases} \tilde{g}(\mathbf{y}^{t,\tau}), \text{if } \mathbf{y}^{t,\tau} \text{ is available} \\ \tilde{g}(\mathbf{z}^{t,\tau}), \text{otherwise} \end{cases} \end{aligned} \tag{5.2}$$

where $\mathcal{L}_{tra}$ corresponds to the loss function for trajectory prediction while $\mathcal{L}_{int}$ is the loss for intensity forecasting. $\boldsymbol{\Theta}$ and $\tilde{\boldsymbol{\Theta}}$ are hyperparameters associated with the hurricane trajectory prediction tasks, respectively. The quantile hyperparameters ξ and $\tilde{\xi}$ are needed to bias the model towards predicting more accurately hurricanes that are close to the coastline or those with high categories. Unlike traditional quantile loss, the quantile hyperparameters here are not constants but are automatically updated in an online fashion. Specifically, the quantile loss terms are updated to reflect the significant threat of a hurricane using the functions $g(\cdot)$ and $\tilde{g}(\cdot)$. Recall that $\mathbf{y}^{t,\tau}$ is the true hurricane location and $\tilde{y}^{t,\tau}$ is the true intensity at time

t for lead time τ. However, since $\mathbf{y}^{t,\tau}$ and $\tilde{y}^{t,\tau}$ may not available during model update, we use the model predictions $\mathbf{z}^{t,\tau}$ and $\tilde{z}^{t,\tau}$ to estimate them and calculate their corresponding quantile hyperparameters. Details of the quantile functions are given in Section 5.3.1.3.

5.3.1.1 $\mathcal{L}_{tra}$ with Distance Quantile Regression

As hurricanes can cause severe damages in civilian populated areas, it is imperative to accurately identify hurricanes that are approaching landfall. Therefore, we would like to bias the model towards learning hurricanes with potential to strike the land. This can be done by encouraging hurricane forecasts that are more likely to make landfall. The possibility of hurricane landfall can be measured by the distance between its current location to the nearest coastline. Specifically, we introduce a distance loss decomposition to evaluate the model performance by taking into account its predicted distance to the coastline. For every ground truth location $\mathbf{y}$, we can find its corresponding projected point $\mathbf{p}$ to the nearest coastline. A unit normal vector to the coastline can be calculated as $\mathbf{n} = \frac{\mathbf{p}-\mathbf{y}}{\|\mathbf{p}-\mathbf{y}\|}$. Given a predicted location $\mathbf{z}$, its distance loss is defined as $\mathbf{d} = \mathbf{z} - \mathbf{y}$. The distance loss vector $\mathbf{d}$ can be decomposed into a parallel, $\mathbf{d}_{\parallel} = \mathbf{d} \cdot \mathbf{n}$, and a perpendicular component, $\mathbf{d}_{\perp} = \mathbf{d} - \mathbf{d}_{\parallel}$, as shown in Figure 5.3. With the definition of distance loss decomposition, the square loss $\frac{1}{2}\|\mathbf{z} - \mathbf{y}\|_2^2$ can be expressed equivalently as follows

$$\begin{aligned} \mathcal{L}_{tra} = \quad & \frac{1}{2}\left(\zeta^2 + \zeta^{*2} + (\mathbf{z}-\mathbf{y})_{\perp}^2\right) \\ \text{s.t.} \quad & (\mathbf{z}-\mathbf{y})_{\parallel} = \zeta - \zeta^*, \\ & \zeta \geq 0, \zeta^* \geq 0 \end{aligned} \tag{5.3}$$

In order to encourage predictions with shorter distances to the coastline, Eqn. (5.3) can be further extended to accommodate the quantile loss in Eqn. (5.4). Note that Eqn. (5.4) is equivalent to Eqn. (5.3) by setting the hyperparameter ξ to 0.5.

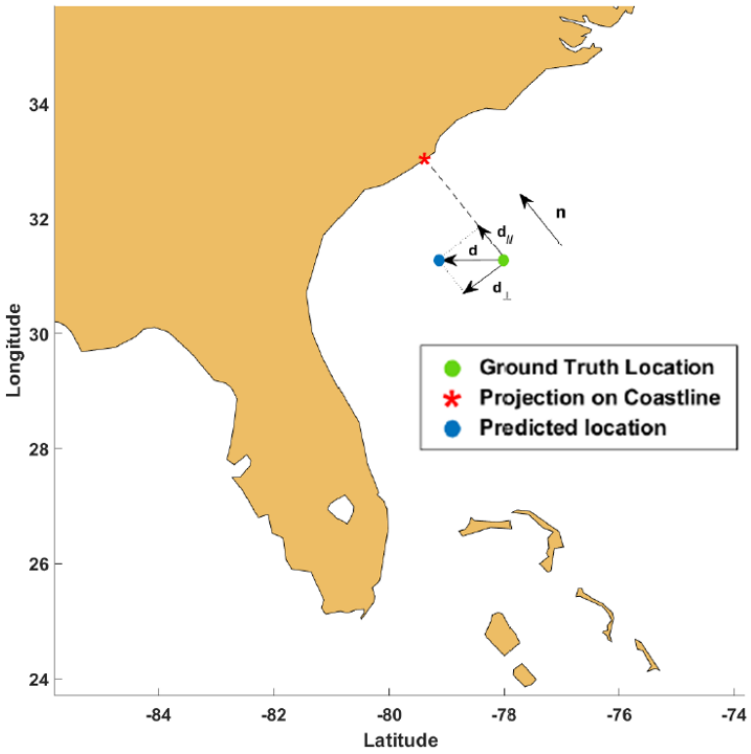


Figure 5.3: Decomposition of the distance loss vector. The green circle is the true location and the red star is its projected nearest coastline. The unit vector $\mathbf{n}$ points in the direction towards the land. The blue circle is the predicted location. The vector directed from the green circle to the blue circle is the distance loss vector $\mathbf{d}$, which can be decomposed into a parallel $\mathbf{d}_{\|}$ and a perpendicular component $\mathbf{d}_{\perp}$.

$$\begin{aligned} \mathcal{L}_{tra} = \quad & (1-\xi)\zeta^2 + \xi\zeta^{*2} + \frac{1}{2}(\mathbf{z}-\mathbf{y})_{\perp}^2 \\ \text{s.t.} \quad & (\mathbf{z}-\mathbf{y})_{\|} = \zeta - \zeta^*, \\ & \zeta \geq 0, \zeta^* \geq 0 \end{aligned} \tag{5.4}$$

We assume that the weight vectors $\mathbf{w}^{t,\tau}$ in Eqn. (5.1) can be decomposed into the following factors:

$$\begin{aligned} & \mathbf{w}^{t,\tau} = \mathbf{u}^t + \mathbf{v}^{t,\tau} \\ \text{s.t.} \quad & \mathbf{1}_{m_t}^T \mathbf{u}^t = 1, \mathbf{1}_{m_t}^T \mathbf{v}^{t,\tau} = 0 \end{aligned} \tag{5.5}$$

where $\mathbf{u}^t$ is the shared weight vectors for all lead times while $\mathbf{v}^{t,\tau}$ is adjustment to the weight vectors associated with the different lead times τ. For brevity, we denote $\mathbf{W}^t = [\mathbf{w}^{t,1}, \mathbf{w}^{t,2}, \cdots, \mathbf{w}^{t,T}]$ as the weight matrix for all T lead times at time step t. To extend the

preceding formulation to an online multi-lead time forecasting setting, the weight matrix $\mathbf{W}^t$ is updated by minimizing the following objective function at each time step t:

$$\begin{aligned}
\mathcal{L}_{tra} = \quad & \sum_{\tau=1}^{T} \delta^{t,\tau}\gamma^{\tau}\left((1-\xi)\zeta^{t,\tau 2} + \xi\zeta^{*t,\tau 2} + \frac{1}{2}(\mathbf{z}^{t,\tau} - \mathbf{y}^{t,\tau})_{\perp}^2\right) \\
+ \quad & \frac{\omega}{2}\sum_{\tau=1}^{T-1}\left\|\mathbf{w}^{t,\tau+1} - \mathbf{w}^{t,\tau}\right\|^2 + \frac{\mu}{2}\left\|\mathbf{u}^t - \mathbf{u}^{t-1}\right\|^2 \\
+ \quad & \frac{\nu}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau} - \mathbf{v}^{t-1,\tau}\right\|^2 + \frac{\eta}{2}\sum_{\tau=1}^{T}\left\|\mathbf{v}^{t,\tau}\right\|^2 \\
\text{s.t.} \quad & \forall\, t,\tau:\ \mathbf{1}_{m_t}^T\mathbf{u}^t = 1,\ \mathbf{1}_{m_t}^T\mathbf{v}^{t,\tau} = 0, \\
& (\mathbf{z}^{t,\tau} - \mathbf{y}^{t,\tau})_{\|} = \zeta^{t,\tau} - \zeta^{*t,\tau}, \\
& \zeta^{t,\tau} \geq 0, \zeta^{*t,\tau} \geq 0
\end{aligned} \tag{5.6}$$

where $\delta^{t,\tau}$ is an indicator function whose value is 1 if $\mathbf{X}^{t,\tau}$ and $\mathbf{y}^{t,\tau}$ values are both available; otherwise its value is 0. In the objective function, the first term represents the forecast errors for all the lead times. The hyperparameter ξ determines the importance of making location predictions with shorter distance to coastlines. The hyperparameter γ decides the relative importance of making accurate predictions at different lead times. The second term ensures that the estimated model parameters would vary smoothly at different lead times, thus preserving the temporal autocorrelation of the predicted intensities. The third and fourth terms guarantee that the shared weight vector $\mathbf{u}^t$ and lead time adjustment weight vectors $\mathbf{v}^{t,\tau}$ are close to their values at previous time step. The last term penalizes large values in the lead time adjustment weight vectors. ω, μ, ν, η are hyperparameters determine the relative importance of each term in the objective function. If we choose a large $\xi \approx 1$, then the distance loss can be ignored if $(\mathbf{z}^{t,\tau} - \mathbf{y}^{t,\tau})_{\|} > 0$. It means that $\mathcal{L}_{tra}$ will give high weights on the models with predictions with shorter distance to the coastline.

5.3.1.2 $\mathcal{L}_{int}$ with Quantile Ordinal Regression

Our goal is to generate accurate long range predictions of hurricane real-valued intensity and category. Here, we use the ϵ-insensitive loss to measure the intensity prediction error.

Compared to mean square loss, the ϵ-insensitive loss is more robust as it provides a margin of tolerance ϵ [28, 5] when learning the regression function.

$$\begin{aligned} \mathcal{L}_{int} = \quad & \zeta + \zeta^* \\ \text{s.t.} \quad & z - y \leq \epsilon + \zeta \\ & z - y \geq -\epsilon - \zeta^* \\ & \zeta \geq 0, \zeta^* \geq 0 \end{aligned} \tag{5.7}$$

Second, in order to communicate the severity of an impending hurricane to the public, accurate prediction of its category is often more important than the wind speed itself. As noted in Section 4.2, intensity prediction alone is insufficient because it ignores whether the prediction error affects its prediction category. Therefore, we introduce an ordinal loss to ensure the model is focused more on data points located near the boundary between two ordinal categories. Analogous to the loss function defined for support vector ordinal regression [17], the ordinal loss function is defined as follows:

$$\begin{aligned} \mathcal{L}_{int} = \quad & \zeta + \zeta^* \\ \text{s.t.} \quad & z - b_y \leq -1 + \zeta, \\ & z - b_{y-1} \geq 1 - \zeta^*, \\ & \zeta \geq 0, \zeta^* \geq 0 \end{aligned} \tag{5.8}$$

Our intensity prediction loss can be measured by combining Eqns. (5.7) and (5.8) as follows. In addition, to penalize models that incorrectly predict high category hurricanes, it can be extended to accommodate the quantile loss as $(1 - \xi)\zeta + \xi\zeta^*$.

$$\begin{aligned} \mathcal{L}_{int} = \quad & (1 - \xi)\zeta + \xi\zeta^* \\ \text{s.t.} \quad & z - b_{\hat{y}} \leq -1 + \zeta \\ & z - b_{\hat{y}-1} \geq 1 - \zeta^* \\ & z - y \leq \epsilon + \zeta \\ & z - y \geq -\epsilon - \zeta^* \\ & \zeta \geq 0, \zeta^* \geq 0 \end{aligned} \tag{5.9}$$

Similar to the trajectory model, we assume that the intensity weight vector $\tilde{\mathbf{w}}^{t,\tau}$ can be decomposed into the following factors:

$$\begin{aligned} \tilde{\mathbf{w}}^{t,\tau} &= \tilde{\mathbf{u}}^t + \tilde{\mathbf{v}}^{t,\tau} \\ \text{s.t.} \quad & \mathbf{1}_{\tilde{m}_t}^T \tilde{\mathbf{u}}^t = 1, \mathbf{1}_{\tilde{m}_t}^T \tilde{\mathbf{v}}^{t,\tau} = 0 \end{aligned} \tag{5.10}$$

where $\tilde{\mathbf{u}}^t$ is the shared weight vectors for all lead times while $\tilde{\mathbf{v}}^{t,\tau}$ is adjustment to the weight vectors associated with the different lead times τ. For brevity, we denote $\tilde{\mathbf{W}}^t = [\tilde{\mathbf{w}}^{t,1}, \tilde{\mathbf{w}}^{t,2}, \cdots, \tilde{\mathbf{w}}^{t,T}]$ as the weight matrix for all T lead times at time step t.

Putting it together, the weight matrix $\tilde{\mathbf{W}}^t$ for intensity prediction is trained to minimize the following objection function:

$$\begin{aligned} \mathcal{L}_{int} = \quad & \frac{1}{2} \sum_{\tau=1}^{T} \tilde{\delta}^{t,\tau} \tilde{\gamma}^{\tau} \left((1-\tilde{\xi}) \zeta^{t,\tau} + \tilde{\xi} \zeta^{*t,\tau} \right) \\ + \quad & \frac{\tilde{\omega}}{2} \sum_{\tau=1}^{T-1} \left\| \tilde{\mathbf{w}}^{t,\tau+1} - \tilde{\mathbf{w}}^{t,\tau} \right\|^2 + \frac{\tilde{\mu}}{2} \left\| \tilde{\mathbf{u}}^t - \tilde{\mathbf{u}}^{t-1} \right\|^2 \\ + \quad & \frac{\tilde{\nu}}{2} \sum_{\tau=1}^{T} \left\| \tilde{\mathbf{v}}^{t,\tau} - \tilde{\mathbf{v}}^{t-1,\tau} \right\|^2 + \frac{\tilde{\eta}}{2} \sum_{\tau=1}^{T} \left\| \tilde{\mathbf{v}}^{t,\tau} \right\|^2 \\ \text{s.t.} \quad & \forall\, t, \tau : \ \mathbf{1}_{\tilde{m}_t}^T \tilde{\mathbf{u}}^t = 1, \ \mathbf{1}_{\tilde{m}_t}^T \tilde{\mathbf{v}}^{t,\tau} = 0, \\ & \tilde{z}^{t,\tau} - b_{\hat{y}^{t,\tau}} \leq -1 + \zeta^{t,\tau} \\ & \tilde{z}^{t,\tau} - b_{\hat{y}^{t,\tau}-1} \geq 1 - \zeta^{*t,\tau} \\ & \tilde{z}^{t,\tau} - \tilde{y}^{t,\tau} \leq \epsilon + \zeta^{t,\tau} \\ & \tilde{z}^{t,\tau} - \tilde{y}^{t,\tau} \geq -\epsilon - \zeta^{*t,\tau} \\ & \zeta^{t,\tau} \geq 0, \zeta^{*t,\tau} \geq 0 \end{aligned} \tag{5.11}$$

where $\tilde{\delta}^{t,\tau}$ is an indicator function whose value is 1 if $\tilde{\mathbf{x}}^{t,\tau}$ and $\tilde{y}^{t,\tau}$ values are both available; otherwise its value is 0. In the objective function, the first term represents the forecast errors for all the lead times. The hyperparameter $\tilde{\xi}$ determines the importance of making accurate predictions for high category hurricanes. The meanings of other terms in the objective function are similar to Equation 5.6.

5.3.1.3 Quantile Hyperparameter Update

As described in the previous section, the quantile hyperparameters are updated in an online fashion. In general, we want the quantile hyperparameter for trajectory prediction to be large if the hurricane category is high; and the quantile hyperparameter for intensity prediction to be large if the hurricane location is close to coastline. In our framework, we use a sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ to generate the quantile hyperparameters.

For $\mathcal{L}_{tra}$, the hyperparameter $\xi^{t,\tau}$ is calculated from the function $g(\mathbf{x})$ defined in Eqn. (5.2) as follows:

$$g(\mathbf{x}) = \begin{cases} 0.5, & \text{for } x < \theta \\ \sigma([x-\theta]/c), & \text{otherwise} \end{cases} \tag{5.12}$$

where $\mathbf{x}$ is the ground truth or predicted intensity with current weight vector, θ is a hyperparameter that decide when quantile loss is not needed, c is the hyperparameter that determines the magnitude of quantile hyperparameters. When the hurricane intensity is low, $g(\mathbf{x}) = 0.5$, and thus, the first loss term in $\mathcal{L}_{tra}$ reduces to the squared loss function. For high intensity hurricanes, $g(\mathbf{x}) \approx 1$. In this case, the framework gives higher weights for models that predict locations with shorter distance to the coastline.

For $\mathcal{L}_{int}$, the hyperparameter $\tilde{\xi}^{t,\tau}$ is calculated from the function $\tilde{g}(\cdot)$ defined in Eqn. (5.2) as follows:

$$\tilde{g}(\mathbf{x}) = \begin{cases} 0.5, & \text{for } d_{coast}(\mathbf{x}) > \tilde{\theta} \\ \sigma([\tilde{\theta} - d_{coast}(\mathbf{x})]/\tilde{c}), & \text{otherwise} \end{cases} \tag{5.13}$$

where $\mathbf{x}$ is either the ground truth or predicted location for the current weight vector, $d_{coast}(\cdot)$ is a function that computes distance to coastline, $\tilde{\theta}$ is a hyperparameter that determines when the quantile loss is not needed, $\tilde{c}$ is the hyperparameter that determines the magnitude of quantile hyperparameters. When the hurricane is far from the coastline, we have $g(\mathbf{x}) = 0.5$. The first loss term of $\mathcal{L}_{int}$ is thus reduced to ℓ_1 loss. When the hurricane is very close the land, $g(\mathbf{x}) \approx 1$. In this case, the framework gives higher weights to models that predict higher intensities.

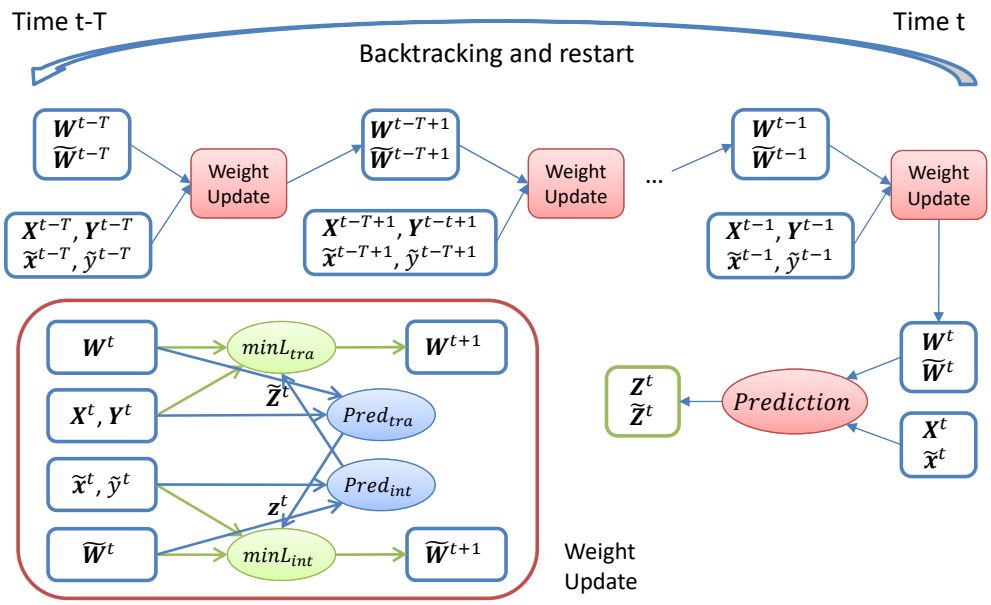


Figure 5.4: Proposed JOHAN framework.

Error propagation is a challenge for multi-lead time forecasting. It is insufficient to update $\mathbf{W}^t$ and $\tilde{\mathbf{W}}^t$ from $\mathbf{W}^{t-1}$ and $\tilde{\mathbf{W}}^{t-1}$ alone as the previous weights are outdated without using the new observation. To address this problem, we apply the backtracking and restart strategy described in Chapter 3. The overall framework with backtracking and restart strategy is illustrated in Figure 5.4. The pseudocode for the proposed framework is described in Algorithm 3. At each time step t, the newly available ground truth value can be determined as $\{\mathbf{y}^{t-T,T}, \mathbf{y}^{t-T+1,T-1}, \ldots, \mathbf{y}^{t-1,1}\}$ for trajectory forecasts and $\{\tilde{y}^{t-T,T}, \tilde{y}^{t-T+1,T-1}, \ldots, \tilde{y}^{t-1,1}\}$ for intensity forecasts. Therefore, to make all the previous learned weight vectors up to date, we updated the weight vectors from time step $t-T$ until $t-1$ with quantile hyperparameters generated by Eqn. (5.12) and (5.13). Then, the multi-lead time predictions for both trajectory and intensity can be produced given the ensemble of model outputs and updated model weight vectors.

5.4 Experiments

We performed experiments using real-world hurricane trajectory and intensity data from various sources. The ground truth hurricane trajectory and intensity data along with the

Input: $\boldsymbol{\Theta}, \tilde{\boldsymbol{\Theta}}, \xi, \xi$
Output: Model parameters $\mathbf{w}, \tilde{\mathbf{w}}$ and forecasts $\mathbf{z}, \tilde{z}$
Initialize: $\mathbf{w} = \mathbf{1}_m/m, \tilde{\mathbf{w}} = \mathbf{1}_m/m$;
for *t = 1,2, . . . , N* **do**
  if *t is the first time step of a trajectory* **then**
    Extract $\mathbf{u}^t$ from $\mathbf{w}$, $\tilde{\mathbf{u}}^t$ from $\tilde{\mathbf{w}}$
    Normalize: $\mathbf{u}^t \leftarrow \mathbf{u}^t/|\mathbf{u}^t|, \mathbf{v}^{t,\tau} \leftarrow \mathbf{0}, \tilde{\mathbf{u}}^t \leftarrow \tilde{\mathbf{u}}^t/|\tilde{\mathbf{u}}^t|, \tilde{\mathbf{v}}^{t,\tau} \leftarrow \mathbf{0}$
  end
  Observe $\mathbf{y}^t, \tilde{\mathbf{y}}^t$
  /* Backtracking and restart step */
  for $t' = t - T, t - T + 1, \ldots, t - 1$ **do**
    Trajectory predictions $\mathbf{z}^{t',\tau} = f^{t',\tau}(\mathbf{X}^{t,\tau}) = \mathbf{X}^{t',\tau}\mathbf{w}^{t',\tau}$
    Intensity predictions $\tilde{z}^{t',\tau} = \tilde{f}^{t,\tau}(\tilde{\mathbf{x}}^{t',\tau}) = \tilde{\mathbf{x}}^{t',\tau}\tilde{\mathbf{w}}^{t',\tau}$
    Calculate hyperparameter $\xi^{t,\tau}$ and $\tilde{\xi}^{t,\tau}$ using Eq. 5.12 and 5.13
    Update $\mathbf{u}^{t'+1}, \mathbf{v}^{t'+1,\tau}$ by minimizing $\mathcal{L}_{tra}$
    Update $\tilde{\mathbf{u}}^{t'+1}, \tilde{\mathbf{v}}^{t'+1,\tau}$ by minimizing $\mathcal{L}_{int}$
  end
  /* Prediction step */
  for $\tau = 1, 2, \cdots, T$ **do**
    Compute $\mathbf{w}^{t,\tau}$ and $\tilde{\mathbf{w}}^{t,\tau}$ for all lead times
    Trajectory predictions $\mathbf{z}^{t,\tau} = f^{t,\tau}(\mathbf{X}^{t,\tau}) = \mathbf{X}^{t,\tau}\mathbf{w}^{t,\tau}$
    Intensity predictions $\tilde{z}^{t,\tau} = \tilde{f}^{t,\tau}(\tilde{\mathbf{x}}^{t,\tau}) = \tilde{\mathbf{x}}^{t,\tau}\tilde{\mathbf{w}}^{t,\tau}$
  end
  if *t is the last time step of a trajectory* **then**
    Substitute $\mathbf{u}^t$ back into the full vector $\mathbf{w}$
    Substitute $\tilde{\mathbf{u}}^t$ back into the full vector $\tilde{\mathbf{w}}$
  end
end

Algorithm 3: Proposed `JOHAN` framework

official forecasts are obtained from the National Hurricane Center (NHC) website[1], while the ensemble member forecasts are obtained from the Hurricane Forecast Model Output website at University of Wisconsin-Milwaukee[2]. We collected 6-hourly hurricane trajectory and intensity data from the year 2012 to 2020, which contains 336 tropical cyclones. Each tropical cyclone has an average length of 21.9 time steps (data points), which gives a total of 7364 data points. There are 27 trajectory forecast models and 21 intensity forecast models used in our experiments, which are a subset of the models used by NHC in the preparation

[1]https://www.nhc.noaa.gov
[2]http://derecho.math.uwm.edu/models

of their official forecasts. The data from 2012 to 2017 (208 tropical cyclones) are used for training and validation, while those from 2018 to 2020 (128 tropical cyclones) are used for testing.

5.4.1 Baseline and Evaluation Metrics

We compared `JOHAN` against the following baseline methods:

1. **Ensemble mean**: This method computes the mean value over all ensemble members for each given lead time.
2. **Persistence**: This method assumes that the intensity and moving speed of the hurricane at the next time step is the same as current time step.
3. **Passive-Aggressive(PA)** [21]: This is a well-known online regression algorithm.
4. **ORION** [84]: This is an online multi-task learning algorithm for ensemble forecasting.
5. **OMuLeT**: This is the online learning algorithm described in the Chapter 3 for trajectory prediction. It can be also applied to the intensity prediction as well using the ensemble member forecasts as predictors.
6. **NHC**: This corresponds to the official forecasts generated by NHC, which is the gold standard.

For a fair comparison, the baseline methods (PA, ORION, OMuLeT, and JOHAN) also apply the backtracking and restart strategy to update their weights. Hyperparameters of the methods were tuned by minimizing the following mean distance error (MDE) for trajectory forecasts and macro-averaged mean absolute error (MAE) for intensity forecasts on the validation set.

$$\text{MDE} = \frac{1}{N} \sum_{t,\tau} \left[dis \left(\mathbf{z}^{t,\tau}, \mathbf{y}^{t,\tau} \right) \right] \tag{5.14}$$

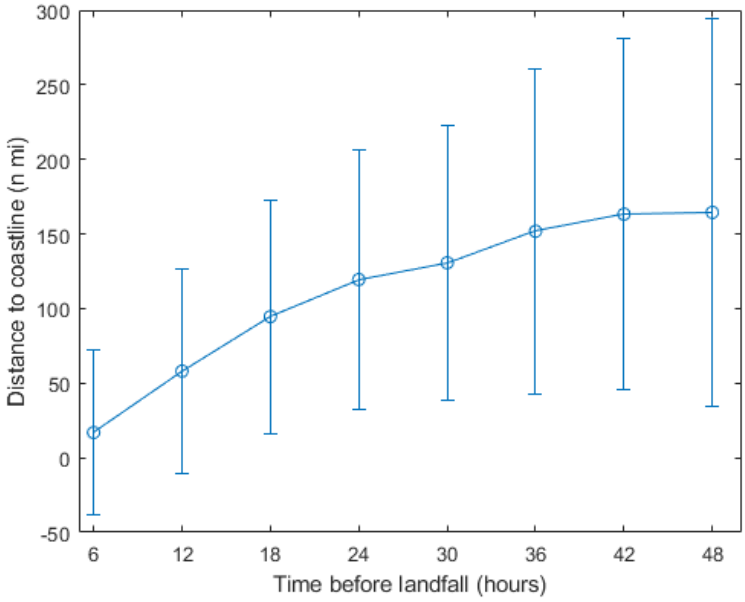


Figure 5.5: Distance to coastline for hurricanes from year 2012 to 2017 at different time before landfall.

$$\text{macro-MAE} = \frac{1}{6}\sum_{i=0}^{5}\left(\frac{1}{N_i}\sum_{\hat{y}^{t,\tau}=i}\left|\hat{z}^{t,\tau}-\hat{y}^{t,\tau}\right|\right) \tag{5.15}$$

where N is total number of trajectory forecasts in the validation set, N_i is the number of data points of category i in the validation set. We use MDE on the trajectory test data to evaluate the location prediction error, and MAE on the intensity test data to evaluate the error in both real- and ordinal-valued predictions. We also use F1-score to evaluate accuracy of the ordinal category predictions.

For `JOHAN` framework, the hyperparameters $\boldsymbol{\Theta}$ and $\tilde{\boldsymbol{\Theta}}$ are tuned with the fixed quantile hyperparameters $\xi = \tilde{\xi} = 0.5$. The threshold θ for the function $g(\cdot)$ is set to 34 knots (kt), which is the lower bound for intensity of a tropical storm. As there are very few time steps with intensity more than 64 kt the hyperparameter c is calculated by solving $g(\mathbf{x}) = \sigma([x-\theta]/c) = 0.9$ with $x = 64$ kt. For the function $\tilde{g}(\cdot)$, Figure 5.5 shows the distance to coastline for landfall hurricanes at different hours before landfall. The threshold $\tilde{\theta}$ is set to 300 nautical miles (n mi) where the hurricanes are unlikely to strike the land in 48 hours. The hyperparameter $\tilde{c}$ is calculated by solving $\tilde{g}(\mathbf{x}) = \sigma([\tilde{\theta} - d_{coast}(\mathbf{x})]/\tilde{c}) = 0.9$

with $x = 200$ n mi which is the distance from the coastline for hurricanes that might reach landfall after 24 hours.

5.4.2 Experimental Results

Table 5.2 summarizes the trajectory and intensity forecast errors of JOHAN and other baseline methods with lead times from 12 hours to 48 hours. For trajectory prediction, there are several interesting conclusions that can be drawn. First, the performance of ensemble mean is comparable to the NHC official forecast, which validates the advantage of using an ensemble of physical model outputs as predictors. Second, persistence performs much worse than other baselines. This is reasonable since persistence assumes that the moving speed of hurricane is unchanged. Third, OMuLeT and JOHAN generate comparable results and both outperform other baselines, including the official forecasts from NHC. This is not surprising as both methods employ similar strategies to update their trajectory prediction models. However, as will be shown below, OMuLeT is inferior to JOHAN in terms of its trajectory prediction error for hurricanes within 200 n mi from the coastline.

For intensity forecasts, the performance of ensemble mean is significantly worse than NHC as shown in Table 5.2. This is not surprising as intensity forecasting is still a very challenging problem for the dynamical and statistical models in the ensemble. Second, both persistence and PA perform poorly, much more so than other methods. Fourth, JOHAN generally has significantly better performance especially at longer lead times with results comparable to the official forecasts of NHC. This is impressive considering the fact that the ensemble members used in JOHAN is only a subset of the models used by NHC to generate their official forecasts.

JOHAN is designed to maximally identify threatening hurricanes with potential to strike landfall. Table 5.3 summarizes the trajectory and intensity forecast errors for hurricanes within 200 n mi to coastline with an intensity of at least 64 kt for different prediction methods at varying lead times (from 12 to 48 hours). The relative performance of all the

	Trajectory error (in n mi)				Intensity error (in kt)			
Method	12	24	36	48	12	24	36	48
Ensemble Mean	23.30	36.34	50.22	65.03	6.742	8.692	9.899	11.036
Persistence	34.84	88.89	155.87	229.63	7.717	13.797	18.246	22.102
PA	23.30	36.34	50.23	64.80	6.547	10.563	13.294	15.268
ORION	23.37	36.36	50.21	65.00	5.792	7.941	9.388	10.551
OMuLeT	**22.20**	**34.94**	**48.07**	62.10	**5.632**	7.923	9.285	10.513
JOHAN	22.25	35.01	48.13	**62.08**	5.732	**7.700**	**8.948**	**10.026**
NHC	24.59	38.49	52.17	65.74	5.019	7.730	8.959	10.140

Table 5.2: Trajectory and intensity forecast errors for different methods at varying lead times from 12 to 48 hours.

	Trajectory error (in n mi)				Intensity error (in kt)			
Method	12	24	36	48	12	24	36	48
Ensemble Mean	15.80	28.47	41.21	52.09	13.274	17.325	20.246	20.382
Persistence	34.28	87.62	159.38	228.91	13.121	22.547	29.194	32.762
PA	15.81	28.48	41.22	51.98	8.765	16.042	24.275	24.656
ORION	16.02	28.61	41.29	52.15	**8.483**	**13.171**	16.806	17.555
OMuLeT	15.33	28.28	39.65	49.67	9.064	14.435	17.562	18.317
JOHAN	**15.28**	**28.15**	**39.28**	**49.06**	8.963	13.367	**16.270**	**16.554**
NHC	16.69	29.47	42.83	54.07	7.962	13.585	16.097	17.587

Table 5.3: Trajectory and intensity forecast errors for hurricanes within 200 n mi to coastline with intensity at least 64 kt for different methods at varying lead times from 12 to 48 hours.

baseline methods is similar to Table 5.2. The trajectory prediction of `JOHAN` outperforms all other baselines for near land hurricanes. This suggests that `JOHAN` is capable of utilizing the relationship between high intensity hurricanes and distance to the coastline to improve its prediction. For intensity predictions, `JOHAN` still maintains the best predictive performance at longer lead times.

In addition, we also evaluated the hurricane category prediction for the different methods by computing their F1-scores. The results are shown in Table 5.4. First, ensemble mean performs poorly for high category predictions, which is not surprising as the different ensemble members are not equally skillful. Second, the overall macro-F1 performance of persistence is similar to ensemble mean and much worse than other baselines. Third, PA is better than ensemble mean but still worse than both ORION and OMuLeT. The macro-F1 score of `JOHAN` is slightly higher than ORION and OMuLeT, though `JOHAN` has better performance

	F1-score						
Category	macro-F1	0	1	2	3	4	5
Ensemble Mean	0.390	0.917	0.449	0.309	0.305	0.168	0.190
Persistence	0.396	0.859	0.327	0.247	0.234	0.330	0.378
PA	0.453	0.912	0.487	0.323	0.321	0.362	0.313
ORION	0.491	0.922	**0.530**	**0.377**	0.349	**0.418**	0.350
OMuLeT	0.494	**0.923**	0.502	0.367	**0.386**	0.380	0.404
JOHAN	**0.499**	0.922	0.499	0.357	0.385	0.380	**0.450**
NHC	0.540	0.924	0.554	0.411	0.390	0.462	0.502

Table 5.4: Comparison of F1-score, precision and recall for various hurricane intensity forecasting methods at different categories.

	F1-score						
Category	macro-F1	0	1	2	3	4	5
Ensemble Mean	0.378	0.919	0.414	0.234	0.318	0.188	0.195
Persistence	0.394	0.854	0.309	0.182	0.236	0.389	0.392
PA	0.454	0.899	0.412	0.281	0.298	0.395	0.436
ORION	**0.505**	**0.927**	**0.509**	**0.330**	0.375	0.431	0.459
OMuLeT	0.493	0.923	0.465	0.290	**0.402**	0.435	0.445
JOHAN	0.496	0.923	0.462	0.274	**0.402**	**0.447**	**0.467**
NHC	0.516	0.920	0.511	0.328	0.362	0.442	0.534

Table 5.5: Comparison of F1-score, precision and recall for hurricanes within 200 n mi to coastline with various hurricane intensity forecasting methods at different categories.

for the higher categories. Table 5.5 summarizes the F1-scores for hurricanes within 200 n mi from the coastline, which clearly demonstrates the superiority of `JOHAN` compared to other baselines for high category hurricanes.

5.4.3 Ablation Study

A key aspect of `JOHAN` is its ability to update the quantile hyperparameter in an online fashion. To investigate the advantages of utilizing a varying quantile hyperparameter, we consider two variations of our framework. `JOHAN-NQ` removes the quantile loss for the first term in Eqn. (5.6) and (5.11) and reduced them to squared loss and ℓ_1 loss. `JOHAN-Q` uses a fixed quantile hyperparameter for all the weight updates.

Table 5.6 summarizes the trajectory and intensity forecast errors for the two variations

of JOHAN framework while Table 5.7 summarizes their corresponding forecast errors for near land high intensity hurricanes. It is clear that increasing the fixed quantile hyperparameters would lead to better performance for near land, high intensity hurricanes (Table 5.6) but at the expense of decreasing overall prediction accuracy (Table 5.7). JOHAN, with its varying quantile hyperparameter, manages to maintain consistently accurate predictions in both scenarios.

	Trajectory (n mi)				Intensity (kt)			
Method	12	24	36	48	12	24	36	48
JOHAN	22.25	35.01	48.13	62.08	5.732	7.700	8.948	10.026
JOHAN-NQ	22.20	34.94	48.07	62.10	5.700	7.654	8.880	9.840
JOHAN-Q(ξ=0.6)	22.21	34.94	48.08	62.07	5.696	7.647	8.877	9.872
JOHAN-Q(ξ=0.7)	22.23	34.94	48.07	62.06	5.704	7.659	8.897	9.939
JOHAN-Q(ξ=0.8)	22.27	34.96	48.08	62.09	5.722	7.704	8.964	10.064
JOHAN-Q(ξ=0.9)	22.33	35.01	48.11	62.14	5.749	7.781	9.073	10.248

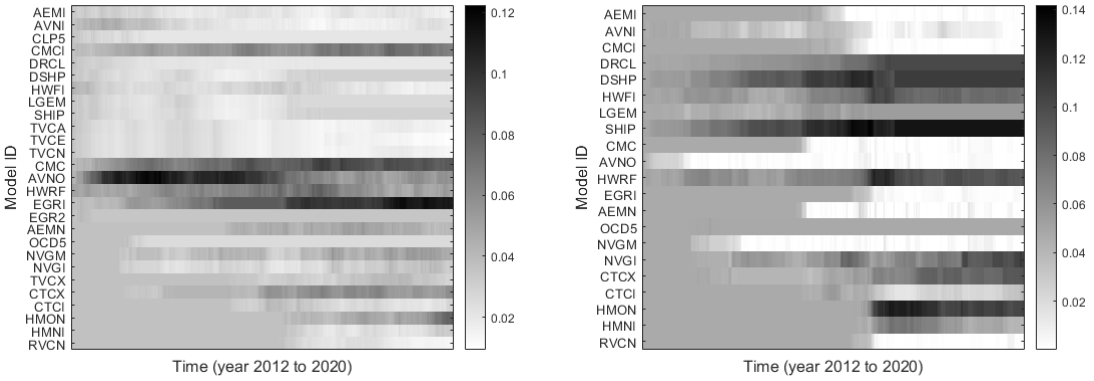
Table 5.6: Trajectory and intensity forecast errors for different variations of JOHAN.

	Trajectory (n mi)				Intensity (kt)			
Method	12	24	36	48	12	24	36	48
JOHAN	15.28	28.15	39.28	49.06	8.963	13.367	16.270	16.554
JOHAN-NQ	15.33	28.28	39.65	49.67	9.176	13.642	16.791	16.854
JOHAN-Q(ξ=0.6)	15.30	28.23	39.53	49.50	9.047	13.488	16.537	16.743
JOHAN-Q(ξ=0.7)	15.28	28.18	39.43	49.33	8.985	13.398	16.346	16.630
JOHAN-Q(ξ=0.8)	15.27	28.15	39.35	49.16	8.950	13.362	16.256	16.606
JOHAN-Q(ξ=0.9)	15.27	28.14	39.29	49.00	8.986	13.412	16.256	16.671

Table 5.7: Trajectory and intensity forecast errors for hurricanes within 200 n mi to coastline and at least 64 kt using different variations of JOHAN.

5.4.4 Comparison of Model Weights

The time-varying model weights generated by JOHAN are shown in Figure 5.6 for trajectory and intensity forecasts, respectively. Despite the shared information between the trajectory and intensity components of the framework, it is clear that the best models for trajectory prediction are not exactly the same as the best models for intensity prediction. For example, AVNO was found to be one of the best models for trajectory prediction but its weight



(a) Trajectory model weights (b) Intensity model weights

Figure 5.6: Trajectory and intensity model weights in `JOHAN` changes over time.

for intensity prediction is close to 0. This result is also reported by [29]. Compared with hurricane track forecasting, intensity forecasting is still a very challenging problem [11].

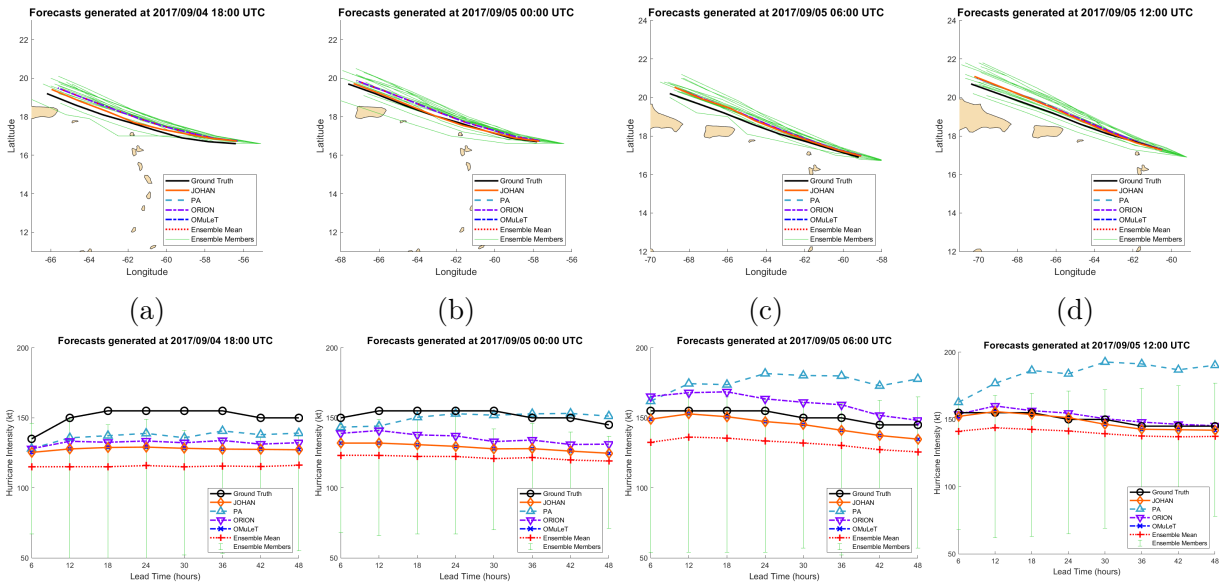


Figure 5.7: 6 to 48 hour forecasts of Hurricane Irma from Sep. 4th to Sep. 5th, 2017. Figure (a) to (d) shows the multi-lead time trajectory forecasts generated from different methods. Figure (e) to (h) demonstrate the multi-lead time intensity predictions from different methods. The error bars represent the range of intensity predictions from the ensemble members.

5.4.5 Case Study

Figure 5.7 shows an example of trajectory and intensity forecasts generated by the different methods for a major hurricane, Irma, from September 4th to 5th, 2017. For trajectory prediction, our JOHAN framework is closest to the best track identified by NHC. The hurricane intensified at the beginning of the period, but neither JOHAN nor other frameworks were able to predict it as none of the ensemble members were able to capture the intensification. Nevertheless, as time progresses, JOHAN was able to adapt its weights and its final prediction gets close to the ground truth intensity. Note that PA and ORION algorithms were able to generate high intensity predictions outside the range of the ensemble members as their model weights are not constrained to sum up to 1. However, they can easily overestimate the intensity at later time steps, as shown in Figure 5.7(g) and (h).

5.5 Conclusions

This chapter presents a novel framework called JOHAN for predicting long-range hurricane trajectory and intensity simultaneously. The framework is designed to address research question **RQ4** described in Chapter 1. JOHAN employs a novel exponentially-weighted quantile loss function to improve its accuracy in predicting high category hurricanes with the potential for near landfall. Experimental results validated the efficacy of JOHAN, especially in terms of accurately predicting near landfall, high category hurricanes.

CHAPTER 6

LSTM FOR TRAJECTORY LOCATION PREDICTION

Long-range trajectory prediction has attracted widespread attention from researchers. There are many applications for trajectory prediction such as animal migration prediction, vehicle trajectory forecasting and hurricane trajectory forecasting. These tasks usually require forecasting multiple lead times into the future, in which the forecasts between lead times are often highly correlated. Researchers had developed many machine learning algorithms to model the correlation between multi-lead times [84, 82]. In Chapter 3, I proposed an online learning approach called `OMuLeT`, which can produce accurate multi-lead times location predictions. However, one potential limitation of `OMuLeT` is that it is a linear model. Due to the complexity and nonlinearity of atmospheric system, the dependencies between the features as well as the multi-lead time forecasts could be non-linear, in which case the `OMuLeT` framework may not be able to learn them effectively.

A neural network is designed to handle nonlinearities in the input features, by using nonlinear activation functions. In recent years, deep learning approaches have found success in many prediction tasks, from computer vision and natural language processing to healthcare and environmental sciences. Among these deep learning models, long short-term memory (LSTM) is one type of recurrent neural network (RNN) that is widely used for time series prediction. Many variations of LSTM model had been developed to handle varieties of time series and sequence prediction tasks. There has also been growing interest recently to apply deep learning methods to the hurricane trajectory forecasting problem [59, 69, 3]. However, there are several limitations to these approaches. First, they are mostly designed to generate short-range forecasts only. Second, they only utilizes historical observation data, which may not be sufficient to overcome the error propagation problem. For applications such as hurricane prediction, the historical data alone is insufficient to capture the current and future environmental conditions that affect its trajectory path. Similar to the approach described in

Chapter 3, we will use the forecasts generated from an ensemble of statistical and dynamical models to generate more accurate trajectory predictions for hurricanes. Third, most of the previous deep learning approaches for hurricane prediction are designed in a batch learning setting.

To address these issues and research question **RQ5**, a novel LSTM-based framework, called `DTP` (**D**eep **T**rajectory **P**rediction) is proposed to aggregate the multi-model ensemble forecasts with a temporal attention-like mechanism. LSTM is a type of Recurrent Neural Network (RNN) that can effectively model time series data with long-term dependencies. The proposed architecture consists of two stages. The first stage consists of a set of LSTM based layers called model performance layers to learn the performance of the individual ensemble members. The second stage is the prediction layer, which uses the output from the previous stage to generate the final multi-lead time predictions. The proposed framework allows us to learn the nonlinear relationships among the ensemble member forecasts as well as the temporal autocorrelations of the predictions. The proposed framework generates its multi-step forecasts by utilizing the outputs of multi-model ensemble members and their corresponding attention weights. As the `DTP` algorithm is trained in a batch mode, it cannot capture concept drift present in the data. To overcome this limitation, I extended the `DTP` framework to `ODTP` (**O**nline **D**eep **T**rajectory **P**rediction), which is an online learning formulation to address the concept drift issue. The proposed frameworks were applied to real-world hurricane data to predict future trajectory path of the hurricane for lead times up to 48 hours. Experimental results showed that `ODTP` can achieve better performance than `DTP`, and generally outperforms other baseline approaches.

6.1 Related Works

Deep learning has been widely used in hurricane trajectory prediction tasks, because of its ability to effectively model the complex non-linear relationships. In order to generate future tracks, existing methods usually use historical trajectory data, climate/meteorological

data or the output of physical models as input features of their prediction models. [69] applied sparse RNN with flexible topology for hurricane trajectory predictions. To generate the predictions, they found the most similar hurricanes to the target hurricane and trained their RNN model with Genetic Algorithm (GA). Alemany et al. [3] utilized Long Short-Term Memory (LSTM) with gridded data to forecast hurricane trajectory. These approaches are only interested in 6-hour short-range forecasts using historical trajectories. Because the historical data cannot capture current and future environmental conditions that affect the path of a hurricane, its performance is often poor and can only be used to generate short-range forecasts. Methods using meteorological data are usually based on deep learning techniques, such as, the generalized adversarial networks (GAN) [77] and convolutional LSTM (ConvLSTM) [54]. Since these models are usually trained using coarse-scale images (e.g., $0.5^\circ \times 0.5^\circ$), their prediction errors are still relatively large. Methods that use physical model output tend to produce more reliable long-range forecasts [29, 82]. As the physical models take current environmental conditions as input and simulate future environmental conditions, their outputs can be used to generate reliable long-range predictions.

6.2 Problem Formulation

For brevity, we present the nonlinear trajectory prediction task in the context of hurricane trajectory forecasting problem. The problem formulation and methodology is applicable to other domains with similar properties. Consider a set of hurricanes, $\{h_1, h_2, \ldots, h_C\}$, ordered by their start times. Assuming there are n_i data points (time steps) associated with hurricane h_i, then $N = \sum_{i=1}^{C} n_i$ is the total number of time steps in the hurricane dataset. Assume T is the forecast horizon, i.e., maximum lead-time of the forecasting task, and M is the total number of ensemble member forecasts. Let $\mathcal{X} \in \mathbb{R}^{2\times M\times T\times N}$ be the set of trajectory forecasts generated by the ensemble of dynamical (physical) models, where each $\mathcal{X}^t \in \mathbb{R}^{2\times M\times T}$ corresponds to the hurricane trajectory forecasts generated at time step t. Let $\tilde{n}_i$ be the accumulated number of data points from hurricane h_1 to h_i, i.e. $\tilde{n}_i = \sum_{j=1}^{i} n_j$.

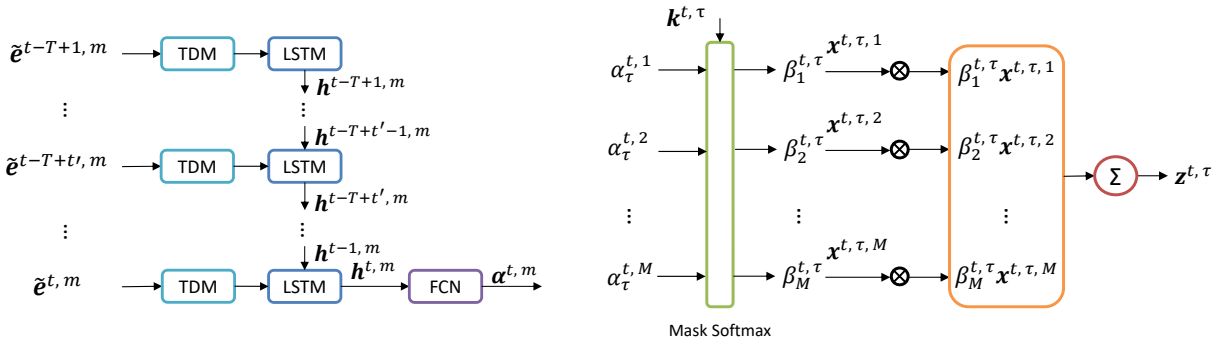
Thus, $\{\mathcal{X}^j \mid \tilde{n}_{i-1} < j \leq \tilde{n}_i\}$ is the set of trajectory and intensity data points associated with hurricane h_i. Let $\mathcal{Y} \in \mathbb{R}^{2\times T\times N}$ to be the corresponding ground truth locations for $\mathcal{X}$, and $\mathbf{Y}^t \in \mathbb{R}^{2\times T}$ to be the ground truth locations for for all lead times at each time step t. The trajectory forecasts of the ensemble members for lead time τ at time t is denoted as $\mathbf{X}^{t,\tau} \in \mathbb{R}^{2\times M}$, with the corresponding ground truth location $\mathbf{y}^{t,\tau} \in \mathbb{R}^2$. Let $\mathcal{E} \in \mathbb{R}^{M\times T\times N}$ to be the distance errors corresponding to trajectory forecasts $\mathcal{X}$. At each time step t, $\mathbf{E}^t \in \mathbb{R}^{M\times T}$ is the distance errors for all the ensemble members at all lead times computed based on their ground truth locations. The distance error for ensemble member m at time t with lead time τ is denoted as $e^{t,\tau,m} = \mathit{distance}(\mathbf{x}^{t,\tau,m}, \mathbf{y}^{t,\tau}) \in \mathbb{R}$. Let $\tilde{\mathbf{e}}^{t,m} = [e^{t-T,T,m}, e^{t-T+1,T-1,m}, \ldots, e^{t-1,1,m}]$ be the distance errors at time t associated with the multi-lead time forecasts generated by the ensemble member m.

Note that the forecast dataset $\mathcal{X}$ and the corresponding distance error $\mathcal{E}$ contain substantial amount of missing values. This is one of the major challenges in using the multi-model ensemble hurricane trajectory data [82], in which the feature length varies from one hurricane to another. Let $\mathcal{K} \in \{0,1\}^{M\times T\times N}$ be the mask values corresponding to trajectory forecasts set $\mathbf{X}$. If the mask value is equal to 1, then the corresponding ensemble member forecast is available; otherwise, the corresponding forecast is missing. At each time step t, let $\mathbf{K}^t \in \{0,1\}^{M\times T}$ be the mask values associated with all the ensemble members for all lead times. The mask value of an ensemble member m for forecasts generated at time t with lead time τ is denoted as $k^{t,\tau,m} \in \{0,1\}$.

6.3 Methodology

6.3.1 `DTP` Framework

This section presents an overview of the proposed `DTP` framework as shown in Figure 6.1. In the first stage shown in Figure 6.1(a), a set of LSTMs were trained to learn an embedding of the ensemble members based on their model performance. Specifically, the distance errors associated with each ensemble member were used as input to the LSTM. Since the distance



(a) Model performance layer

(b) Prediction layer

Figure 6.1: The figure illustrate the DTP framework. Figure (a) shows model performance layer. Figure (b) shows prediction layer.

errors contain missing values, I implemented an approach known as Temporal Decay Memory (TDM) for imputing the missing values before providing the data as input to the LSTMs. In the second stage shown in Figure 6.1(b), the outputs of model performance layer were combined to generate attention-like weights for each ensemble member forecast. The final multi-lead time predictions are computed based on the attention weights and multi-model ensemble forecasts. The detailed architecture of proposed DTP framework is discussed in the following sections.

6.3.1.1 Temporal Decay Memory

In 2018, Kim and Chi [55] proposed an approach Temporal Belief Memory (TBM) for imputing missing values in data. It is a memory module that has two gating units, a missing gate m and a belief gate b. Inspired by this approach, I designed a method called Temporal Decay Memory (TDM) to impute the distance errors in hurricane trajectory forecasts. The architecture of TDM method is shown in Figure 6.2. It contains one gating unit, which is the missing gate m. The missing gate $m \in \{0, 1\}$ indicates if the value is missing or not. If the value is missing, then $m = 1$, otherwise $m = 0$. If the value is observed ($m = 0$), the observed value is directly passed to the output. If the value is missing ($m = 1$), the value passed to the output is a combination of the last observation value $\mathbf{x}_l$, the mean value of all

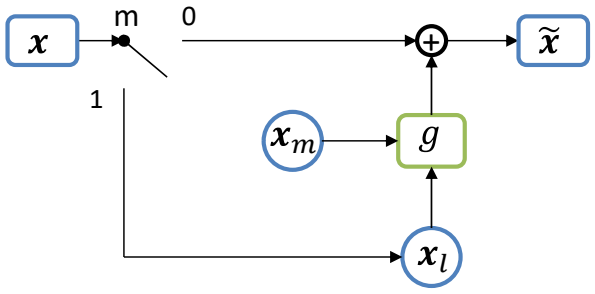


Figure 6.2: Temporal Decay Memory for data imputation.

observations $\mathbf{x}_m$ and the time interval Δt between the current time and the last observation time. Unlike the TBM method, we do not replace the missing value by the last observation $\mathbf{x}_l$ nor the mean observation value $\mathbf{x}_m$. In TDM, we use a function g to calculate the imputed missing value based on $\mathbf{x}_l$ and $\mathbf{x}_m$. Note that the influence of the last observed value $\mathbf{x}_l$ decreases as the time interval Δt increases. The function g is calculated as follows:

$$\tilde{\mathbf{x}} = g(\Delta t, \mathbf{x}_l, \mathbf{x}_m) = e^{-\Delta t/\lambda}\mathbf{x}_l + (1 - e^{-\Delta t/\lambda})\mathbf{x}_m \tag{6.1}$$

where λ is the hyperparameter for the decay rate.

The overall implementation of TDM can be expressed by using Equation 6.2. If $m = 0$, then $\tilde{\mathbf{x}} \to \mathbf{x}$. However, if $m = 1$ and $\Delta t \to 0$, then $\tilde{\mathbf{x}} \to \mathbf{x}_l$, whereas if $m = 1$ and $\Delta t \to \infty$, then $\tilde{\mathbf{x}} \to \mathbf{x}_m$. The hyperparameter λ can be determined by minimizing the imputation loss of the random sampled data points.

$$\tilde{\mathbf{x}} = TDM(m, \Delta t, \mathbf{x}_l, \mathbf{x}_m) = (1 - m)\mathbf{x} + m\left(e^{-\Delta t/\lambda}\mathbf{x}_l + (1 - e^{-\Delta t/\lambda})\mathbf{x}_m\right) \tag{6.2}$$

TDM was developed to impute the missing values in the distance error $\mathcal{E}$. When the forecast $\mathbf{x}$ of an ensemble member is missing, the corresponding distance error $\mathbf{e}$ is also unavailable. Since LSTM requires a complete set of (non-missing) input values, we need to impute the missing values in distance error $\mathbf{e}$ first using TDM before providing them to the LSTM. While TDM can effectively impute the missing values in distance error for computing a feature embedding of the models, using it to directly impute the trajectory forecasts of the ensemble members is not optimal since the current location can be very different than the most recently observed location as well as the mean location at all times.

6.3.1.2 Model Performance Layer

The model performance layer aims to learn the performance of the individual ensemble members, as well as the the temporal dependencies between different lead times. Let $\tilde{\mathbf{e}}^{t,m} \in \mathbb{R}^T$ denotes the distance errors of the forecasts generated by ensemble member m with respect to the predicting location at time t. The model performance layer takes the sequence $\{\tilde{\mathbf{e}}^{t-T+1,m}, \tilde{\mathbf{e}}^{t-T+2,m}, \ldots, \tilde{\mathbf{e}}^{t,m}\}$ as input and output a final hidden state $\mathbf{h}^{t,m}$ using the stacked LSTM network. The outputs of LSTM can be considered as a feature embedding of the ensemble model m at time t. The hidden state $\mathbf{h}^{t,m}$ is calculated as a function of the previous hidden state $\mathbf{h}^{t-1,m}$ and $\tilde{\mathbf{e}}^{t,m}$ using the following equation:

$$\mathbf{h}^{t,m} = LSTM\left(\tilde{\mathbf{e}}^{t,m}, \mathbf{h}^{t-1,m}\right) \tag{6.3}$$

The long term memory is maintained by the cell sate vector $\mathbf{c}_t$ in the LSTM, which is updated using the following equation:

$$\mathbf{c}_t = \mathbf{i}_t \circ \mathbf{c}_{t-1} + \mathbf{f}_t \circ \tilde{\mathbf{c}}_t \tag{6.4}$$

where $\mathbf{i}_t$ is the input gate and $\mathbf{f}_t$ is the forget gate. The gates control the amount of change to the cell state vector $\mathbf{c}_t$. The input gate $\mathbf{i}_t$, forget gates $\mathbf{f}_t$ and cell input activation vector $\tilde{\mathbf{c}}_t$ are calculated a follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i\tilde{\mathbf{e}}^{t,m} + \mathbf{U}_i\mathbf{h}^{t-1,m} + \mathbf{b}_i\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f\tilde{\mathbf{e}}^{t,m} + \mathbf{U}_f\mathbf{h}^{t-1,m} + \mathbf{b}_f\right) \\
\tilde{\mathbf{c}}_t &= \tanh\left(\mathbf{W}_c\tilde{\mathbf{e}}^{t,m} + \mathbf{U}_c\mathbf{h}^{t-1,m} + \mathbf{b}_c\right)
\end{aligned} \tag{6.5}$$

Next, the output gate's activation vector $\mathbf{o}_t$ and hidden state $\mathbf{h}^{t,m}$ can be calculated as follows:

$$\begin{aligned}
\mathbf{h}^{t,m} &= \mathbf{o}_t \circ \tanh\left(\mathbf{c}_t\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o\tilde{\mathbf{e}}^{t,m} + \mathbf{U}_o\mathbf{h}^{t-1,m} + \mathbf{b}_o\right)
\end{aligned} \tag{6.6}$$

In the above equations, the operator $\circ$ denotes the Hadamard product, $\sigma(\cdot)$ denotes a sigmoid activation function, and $\tanh(\cdot)$ denotes a hyperbolic tangent function.

A fully connected network (FCN) takes the hidden states $\mathbf{h}^{t,m}$ as input to generate a multi-lead time performance vector $\boldsymbol{\alpha}^{t,m} \in \mathbb{R}^T$. The relationship between the input and output can be expressed as follows:

$$\boldsymbol{\alpha}^{t,m} = FCN(\mathbf{h}^{t,m}) \tag{6.7}$$

6.3.1.3 Prediction Layer

In this layer, a temporal attention mechanism is used to automatically generate the attention weights for all the ensemble members across all lead times. Based on the outputs of the model performance layer, a multi-lead time performance vector $\alpha^{t,m}$ is obtained for each model m at time t. The attention weights for all ensemble members across all lead times can be generated using a masked softmax layer. The masked softmax function will ensure that the weights of the missing values are set to zero. The attention weight $\boldsymbol{\beta}^{t,\tau} \in \mathbb{R}^M$ for all ensemble members at time t with lead time τ can be calculated based on the following equation:

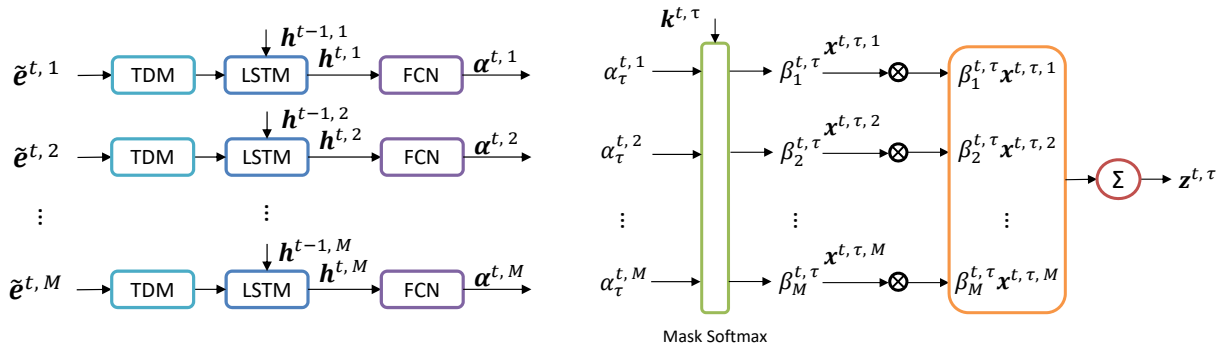
$$\begin{aligned} \beta_m^{t,\tau} &= \frac{exp(a_\tau^{t,m})}{\sum_{i=1}^{M} exp(a_\tau^{t,i})}, \\ \text{where} \quad a_\tau^{t,m} &= \begin{cases} \alpha_\tau^{t,m}, \text{if } k^{t,\tau,m} = 1 \\ -\infty, \text{otherwise} \end{cases} \end{aligned} \tag{6.8}$$

Finally, the multi-lead time predictions at time step t can be computed as a linear combination of the ensemble member forecasts.

$$\mathbf{z}^{t,\tau} = \sum_{i=1}^{M} \beta_i^{t,\tau} \mathbf{x}^{t,\tau,i} \tag{6.9}$$

6.3.2 `ODTP` Framework

The `DTP` framework described in the previous subsection is trained in a batch mode. Due to concept drift, such a model can easily become outdated when applied to real-world problems such as hurricane prediction. To overcome this limitation, the `DTP` framework is extended to



(a) Model performance layer

(b) Prediction layer

Figure 6.3: The figure illustrate the ODTP framework. Figure (a) shows model performance layer. Figure (b) shows prediction layer.

an online learning approach called ODTP, which allows the model to be continuously updated as new observation data become available.

The proposed ODTP framework is shown in Figure 6.3. Similar to the DTP framework, ODTP also contains two layers—the model performance layer and the prediction layer. Unlike the model performance layer in DTP that employs a fixed sequence of length T to train the LSTM model, ODTP considers only a sequence of length 1 to update the model incrementally. Furthermore, for each given sequence of length T, the hidden state in DTP is initialized to zero. In contrast, the hidden state of the LSTM cell is inherited from its previous time step. This strategy allows the hidden state for each model to be continuously updated in an online fashion. In addition, online learning is applied to update the model to fit the new observations. Following the strategy described in Chapter 3, to alleviate the error propagation problem, the algorithm will backtrack to its previous T time steps and restart the update from the time step $t-T$ and incrementally update the model until the current time step t. The online learning with backtrack and restart strategy adopted by ODTP framework can thus help to adapt to concept drift and overcome the error propagation issue in multi-lead time forecasting problems. The overall proposed framework is shown in Figure 6.4, with the pseudocode given in Algorithm 4.

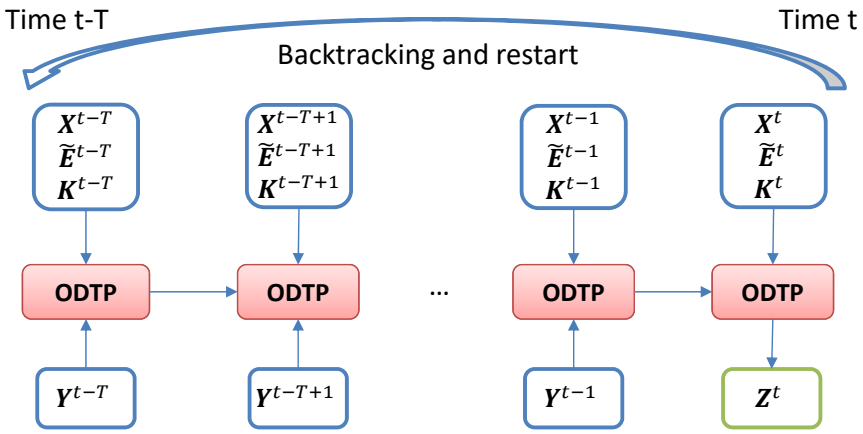


Figure 6.4: The ODTP framework using backtracking and restart strategy.

Input: Hyperparameter $\boldsymbol{\Theta}$ for ODTP model M
Output: Forecasts $\mathbf{z}$
Initialize: Pre-train model $M^{(0)}$ using training dataset;
for t = *1,2, . . . ,* N **do**
  Observe the trajectory location at time t
  /* Backtracking and restart step */
  for $t' = t-T, t-T+1, \ldots, t-1$ **do**
    Update model $M^{(t')}$ using backpropagation with all observed trajectory locations till current time t
  end
  $M^{(t)} \leftarrow M^{(t-1)}$
  /* Prediction step */
  for $\tau = 1, 2, \cdots, T$ **do**
    Generate trajectory predictions $\mathbf{z}^{t,\tau}$ using model $M^{(t)}$
  end
end

Algorithm 4: Proposed ODTP framework

6.4 Experiments

The hurricane best track (ground truth) data and NHC official forecasts are available from the NHC website[1], while the ensemble member forecasts were downloaded from the Hurricane Forecast Model Output website at University of Wisconsin-Milwaukee[2]. According to NHC, 46 models were used in the preparation of their official forecasts. However, only 27 of them have data available from the year 2012 to 2020 at the UWM website. We will use the

[1]https://www.nhc.noaa.gov
[2]http://derecho.math.uwm.edu/models

forecasts from these 27 models as ensemble members in our experiments. We collected the hurricane trajectory data from year 2012 to 2020. The final dataset contains 336 tropical cyclones and total 7364 observations at 6 hour interval. Each tropical cyclones has a average length of 21.9 time steps. For ensemble members with 12-hourly interval, we performed linear interpolation to impute the missing values so that every ensemble member has 6-hourly forecasts. The hurricane data from 2012 to 2017 (208 tropical cyclones) were used for training and validation while those from 2018 to 2020 (128 tropical cyclones) were used for testing.

6.4.1 Baseline and Evaluation Metrics

We compared our `DTP` framework against the following baseline methods:

1. **Ensemble mean**: This method simply calculates the mean value of the ensemble member outputs at each lead time as its predictions.

2. **Persistence**: This method assumes the moving speed at each time step is equal to moving speed at its previous time step. Then, the new location can be easily calculated for each lead time.

3. **Passive-Aggressive(PA)** [21]: This is a well-known online learning algorithm that updates the weights based on newly observed data points.

4. **ORION** [84]: This is an online multi-task learning algorithm for multi-lead time forecasting.

5. **OMuLeT**: This is the online learning algorithm described in the Chapter 3 for trajectory prediction.

6. **NHC**: This is the gold standard, corresponding to the official forecasts generated by NHC.

Online model	Trajectory error (in n mi)			
Lead Time	12	24	36	48
Ens Mean	23.30	36.34	50.22	65.03
Persistance	34.84	88.89	155.87	229.63
NHC	24.59	38.49	52.17	65.74
PA	23.30	36.34	50.23	64.80
ORION	23.37	36.36	50.21	65.00
OMuLeT	**22.33**	**35.33**	48.97	63.77
LSTM	41.64	94.50	160.35	232.80
DTP	23.20	36.08	49.72	64.40
ODTP	22.90	35.50	**48.85**	**63.27**

Table 6.1: Trajectory and intensity forecast errors for different methods at varying lead times from 12 to 48 hours.

7. **LSTM**: This is the vanilla LSTM architecture trained on the historical trajectories with a window size of 48 hours and 6-hour interval.

For a fair comparison, the model is trained on the same training set. The performance of the forecasts can be evaluation by the Mean Distance Error (MDE). The mean distance error for lead time τ can be defined as MDE_τ in the following equation:

$$\text{MDE}_\tau = \frac{1}{\sum_t k^{t,\tau}} \sum_{t,k^{t,\tau}=1} distance(\mathbf{z}^{t,\tau}, \mathbf{y}^{t,\tau}) \tag{6.10}$$

6.4.2 Performance Comparison

The results comparing the hurricane trajectory prediction errors for various methods from 12 hour to 48 hour forecast lead times are shown in Table 6.1. Note that the results reported here for `OMuLeT` are slightly different from the ones given in Section 3.4.1 since the latter were based on models trained and tested on hurricane data from the years up to 2018 only. The performance comparison results shown in the table can be summarized as follows. First, observe that the performance of the persistence and vanilla LSTM methods are significantly worse than other approaches. This is because both methods use only historical trajectories, which are insufficient to generate accurate trajectory forecasts. Second, the performance

of DTP framework is slightly worse than OMuLeT framework, though they both outperform other baselines including the NHC official forecasts. Furthermore, ODTP can further improve the performance of DTP in all lead times, which suggests the advantages of using an online learning approach to adapt the model to changes in the distribution of the data. Specifically, the 48-hour forecast error is decreased from 64.40 n mi to 63.27 n mi. Experiment results also show that ODTP outperform all other online linear models such as PA, ORION, and OMuLeT at 36 hour or more forecast lead times. This suggests the benefits of using a nonlinear model to capture the relationships in the multi-lead time forecasts generated at different time steps.

6.4.2.1 Case Study

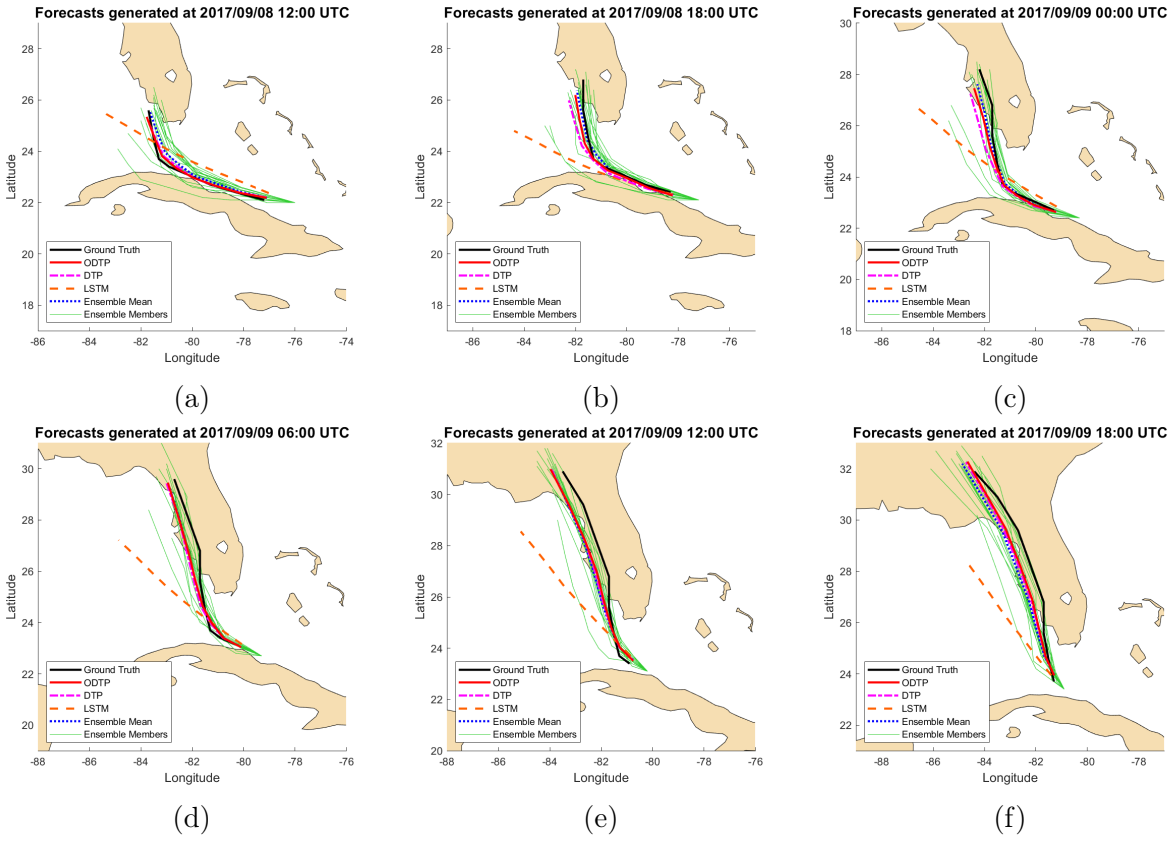


Figure 6.5: Comparison of 48-hour forecasts for Hurricane Irma from 2017/09/08 to 2017/09/09 by different methods.

Figure 6.5 shows an example of the trajectory forecasts for Hurricane Irma from Septem-

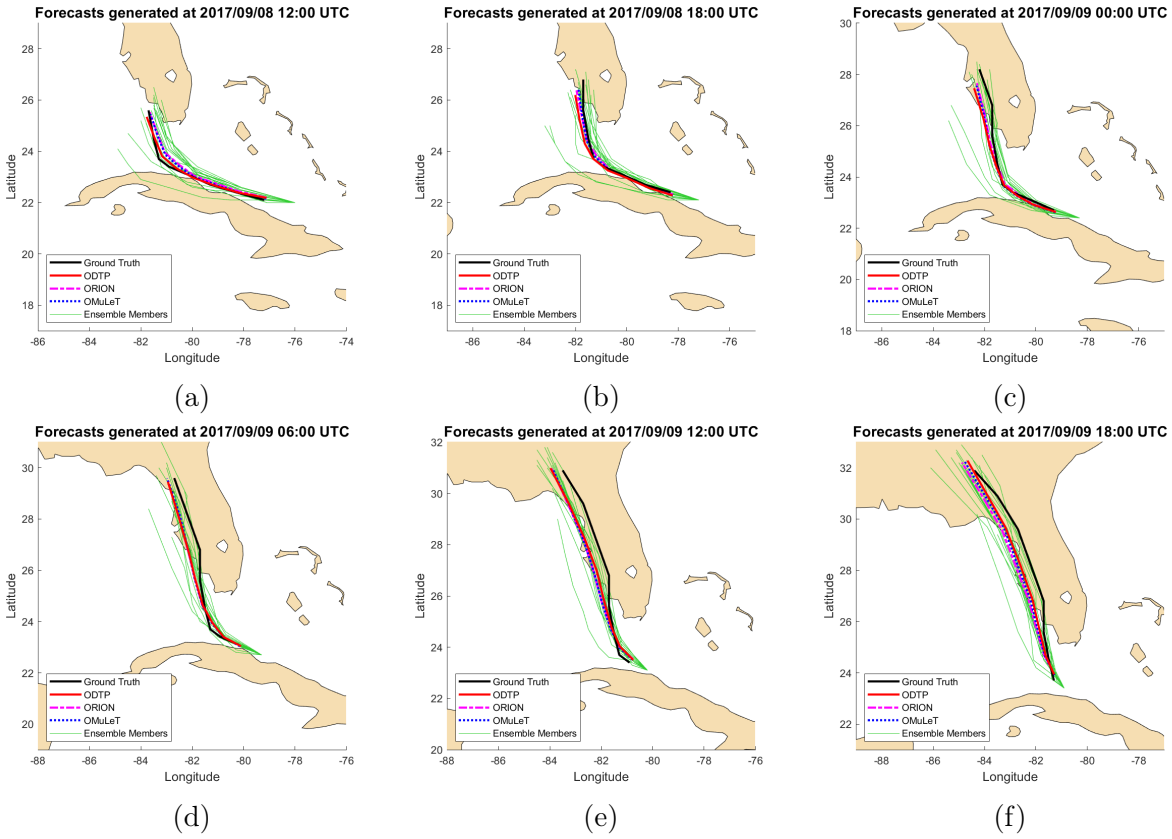


Figure 6.6: Comparison of 48-hour forecasts for Hurricane Irma from 2017/09/08 to 2017/09/09 by different methods.

ber 8 to September 9, 2017. The plots compare the forecasts generated by ODTP against DTP and several other batch learning algorithms. The results suggest that the vanilla LSTM can only predict short-term forecasts accurately, but not for longer lead times. In most of the cases, the forecasts generated by ODTP algorithm are closest to the best track compared to other baseline methods. Although the trajectory forecasts generated by the ensemble members vary quite significantly, ODTP can learn the appropriate attention weights to the ensemble members to make the prediction more accurate. Furthermore, ODTP also produces more accurate predictions than DTP, which can be clearly seen from Figures 6.5(b) and (c). This verifies the importance of using an online learning approach for trajectory prediction. Figure 6.6 compares ODTP algorithm against all other online learning baseline algorithms. As can be seen from the figure, online learning algorithms generate quite comparable multi-lead time forecasts. However, as shown in Table 6.1, there is still improvements in ODTP compared

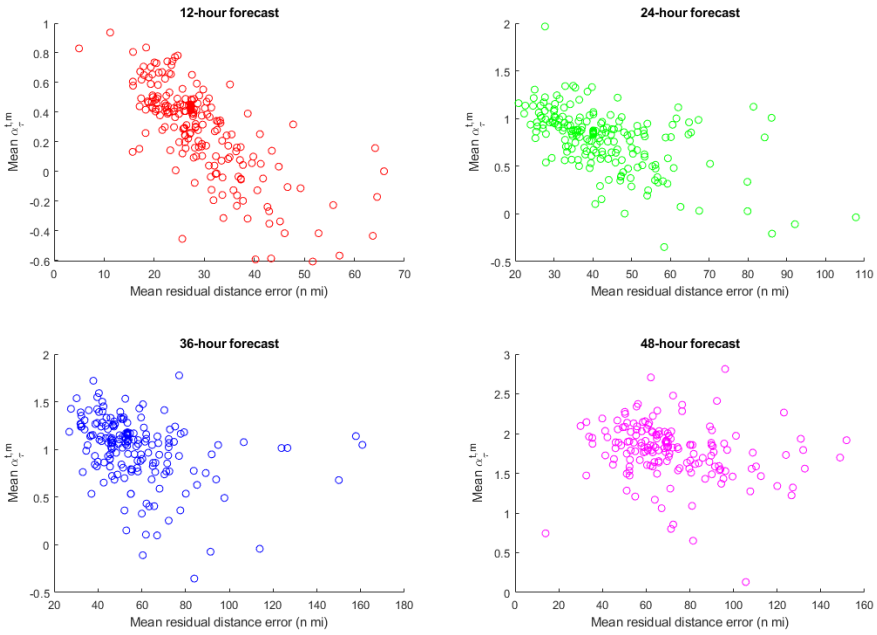


Figure 6.7: Mean residual distance error $e^{t-\tau,m,\tau}$ vs. mean $\alpha_\tau^{t,m}$ of all the time steps within one hurricane for physical model AVNO. The correlations scores are -0.7294, -0.5489, -0.3457, -0.2133 for 12-hour, 24-hour, 36-hour, 48-hour lead time forecasts, respectively.

to other online linear models.

6.4.2.2 Analysis of the Model Performance Layer

Based on the discussion in Section 6.3.1.2, we expect the performance layer should be able to learn an embedding of the ensemble members based on their model performance. The embeddings are then used to generate appropriate attention weights for the ensemble members. To verify this, we will analyze the relationship between the input and output of the model performance layer. For each physical model m, the input of the LSTM in model performance layer is the distance error $\tilde{\mathbf{e}}^{t,m}$ while its output corresponds to the embedding vector, $\boldsymbol{\alpha}^{t,m}$. Figure 6.7 shows the scatter plots of mean residual distance error $e^{t-\tau,m,\tau}$ against the mean vector $\alpha_\tau^{t,m}$ for all the time steps in a given hurricane for the AVNO model. The correlation between the mean distance errors and mean vector $\alpha_\tau^{t,m}$ are -0.7372, -0.5440, -0.3450, -0.2128 for 12-hour, 24-hour, 36-hour, 48-hour lead time forecasts, respectively. Even though the output $\boldsymbol{\alpha}^{t,m}$ also depends on the long-term memory in LSTM, it is clear that there is a

significant negative relationship between the residual distance error $e^{t-\tau,m,\tau}$ and the output score $\alpha_\tau^{t,m}$. The larger the residual error, the smaller the embedding vector $\alpha_\tau^{t,m}$. This agrees with our expectation that $\alpha_\tau^{t,m}$ captures the performance of the ensemble member m.

6.5 Conclusions

In this chapter, I proposed an LSTM based trajectory forecasting framework called `DTP` and its online counterpart, `ODTP`. Unlike existing RNN based approaches for hurricane trajectory prediction, the proposed frameworks aim to produce accurate long-range forecasts by leveraging the outputs generated from an ensemble of statistical and dynamical models. To handle the missing value problem, a novel TDM (Temporal Decay Memory) structure was developed. Both frameworks were applied to real-world hurricane dataset to predict the hurricane trajectory for up to 48 hours lead time. Experimental results showed that `ODTP` can achieve better performance than `DTP`, and generally outperforms other baseline approaches.

CHAPTER 7

CONCLUSIONS AND FUTURE WORKS

In this chapter, a summary of the thesis contributions are presented along with suggestions for future research.

7.1 Summary of Thesis Contributions

In this thesis, a family of online learning algorithms was developed to handle a variety of trajectory location prediction and state prediction tasks. These algorithms were designed to overcome the various limitations of existing approaches, including handling multi-lead time predictions, missing values, ordinal-valued state predictions, etc.

First of all, to address research question **RQ1**, I developed a framework called `OMuLeT` for multi-lead time location prediction. The proposed framework could improve trajectory prediction by combining outputs generated from an ensemble of prediction models in an online fashion. In order to generate accurate long range predictions, `OMuLeT` employs a backtracking with restart strategy to incrementally update the model weights when new observation data become available. It can also handle the varying feature length issue with a weight re-normalization strategy.

Second, I proposed the `OOR` framework to address research question **RQ2**, which is to handle the state prediction task with an ordinal target variable. `OOR` employs an ordinal loss function in its formulation to process the ordinal variable and generate ordinal predictions. To address the research question **RQ3**, the framework was extended to `OOQR`, which accommodates a quantile loss function to improve its prediction accuracy for high/low ordinal category values. Furthermore, using an ϵ-insensitive loss function, the `OOR-`ϵ and `OOQR-`ϵ frameworks were developed to simultaneously generate real-valued and ordinal-valued state predictions.

Third, I introduced a joint learning framework called `JOHAN` for the simultaneous location

and state prediction of a moving object in order to address research question **RQ4**. JOHAN utilizes an exponentially-weighted quantile loss function in its formulations for location and state predictions, in which the hyperparameter of the quantile loss function is updated jointly in real-time.

Finally, I proposed a LSTM based approach called DTP for research question **RQ5**. DTP approach aims to produce accurate long-range predictions uses the location predictions generated from an ensemble of predicting models. In order to solve the concept drift problem in the trajectory prediction task, I developed an online implementation of the DTP framework called ODTP to further improve the prediction performance.

All of the developed approaches were successfully applied to the hurricane prediction task. The OMuLeT approach was used to predict the future hurricane trajectory. Experimental results showed that OMuLeT significantly outperformed various baseline methods, including the official trajectory forecasts produced by NHC. The OOR/OOQR framework was used to predict the ordinal-valued hurricane categories. Experimental results suggested that OOR generates more accurate predictions of the hurricane categories compared to several baseline methods. In addition, a variation of the framework called OOQR can further improve its accuracy in predicting high category hurricanes. The JOHAN framework can generate hurricane trajectory and intensity predictions simultaneously. Experimental results demonstrated that JOHAN can further improve hurricane intensity predictions and achieves superior performance in terms of identifying high category hurricanes approaching the land. The DTP/ODTP are non-linear models that were used to predict the trajectory paths of hurricanes. Experimental results showed that DTP outperforms various batch learning algorithms whereas its online version, called ODTP, can further boost the performance and generates more accurate long range predictions than other baseline methods.

7.2 Future Works

Although the proposed frameworks showed great promise in terms of addressing the hurricane prediction problem, there are several potential directions for future research that can be pursued. This section outlines some of the potential future works.

Application to other Domains In this thesis, I mainly applied the proposed frameworks to hurricane prediction tasks. However, it is clear that many of these frameworks are also applicable to other domains, in which useful features beyond historical trajectory (or state) information are available. For example, the `OMuLeT` framework can be used to forecast vehicle trajectory, utilizing a set of multi-lead time location predictions from various real-time features available. The `OOR`/`OOR-`ϵ frameworks can also be applied to other ordinal prediction tasks.

LSTM for Trajectory Location and State Prediction The `DTP`/`ODTP` frameworks were developed and evaluated for trajectory location predictions. Extending the frameworks to a joint learning approach for location and state prediction is a potential future research direction. In particular, how to combine the joint trajectory/state learning into deep learning framework is an interesting problem that has not been sufficiently addressed.

Deep Learning Approach for Hurricane Trajectory Prediction with Image Data In this thesis, we developed an LSTM-based algorithms `DTP`/`ODTP`, which uses the outputs from an ensemble of statistical/dynamical models. In recent years, deep learning approaches, such as convolutional neural networks, have been successfully applied to satellite imagery data [54, 39, 38] for hurricane location predictions. Merging my proposed frameworks with trajectory prediction based on satellite imagery data will be an interesting direction to pursue.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Assessing the U.S. Climate in 2018. *National Centers for Environmental Information (NCEI)*, 2019.

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, pages 961–971, 2016.

[3] S. Alemany, J. Beltran, A. Perez, and S. Ganzfried. Predicting hurricane trajectories using a recurrent neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 468–475, 2019.

[4] A. Asahara, K. Maruyama, A. Sato, and K. Seto. Pedestrian-movement prediction based on mixed Markov-chain model. In *Proceedings of the 19th ACM SIGSPATIAL*, pages 25–33. ACM, 2011.

[5] M. Awad and R. Khanna. Support vector regression. In *Efficient learning machines*, pages 67–80. Springer, 2015.

[6] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

[7] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[8] E. S. Blake and D. A. Zelinsky. Tropical Cyclone Report: Hurricane Harvey. *National Hurricane Center*, 2018.

[9] E. S. Blake, C. W. Landsea, and E. J. Gibney. *The deadliest, costliest, and most intense United States tropical cyclones from 1851 to 2010 (and other frequently requested hurricane facts)*. National Weather Service, National Hurricane Center, 2011.

[10] A. Bolbol, T. Cheng, I. Tsapakis, and J. Haworth. Inferring hybrid transportation modes from sparse GPS data using a moving window SVM classification. *Computers, Environment and Urban Systems*, 36(6):526–537, 2012.

[11] D. Brown. Tropical Cyclone Intensity Forecasting: Still a Challenging Proposition. *National Hurricane Center*, 2017.

[12] J. P. Cangialosi. National Hurricane Center forecast verification report: 2019 hurricane season, 2020.

[13] H. Cao, N. Mamoulis, and D. W. Cheung. Mining frequent spatio-temporal sequential patterns. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[14] H. Cao, N. Mamoulis, and D. W. Cheung. Discovery of periodic patterns in spatiotemporal sequences. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):453–467, 2007.

[15] J. S. Cardoso, J. F. P. da Costa, and M. J. Cardoso. Modelling ordinal relations with SVMs: An application to objective aesthetic evaluation of breast cancer conservative treatment. *Neural Networks*, 18(5-6):808–817, 2005.

[16] H. Cheng, P.-N. Tan, J. Gao, and J. Scripps. Multistep-Ahead Time Series Prediction. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 765–774, 2006.

[17] W. Chu and S. S. Keerthi. Support vector ordinal regression. *Neural computation*, 19 (3):792–815, 2007.

[18] U. Costliest. Tropical Cyclones Tables Updated. *National Hurricane Center*, 2018.

[19] T. S. Cox, C. S. Hoi, C. K. Leung, and C. R. Marofke. An accurate model for hurricane trajectory prediction. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 534–539. IEEE, 2018.

[20] K. Crammer and Y. Singer. Pranking with ranking. In *Advances in neural information processing systems*, pages 641–647, 2002.

[21] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[22] C. Davis, W. Wang, S. S. Chen, Y. Chen, K. Corbosiero, M. DeMaria, J. Dudhia, G. Holland, J. Klemp, J. Michalakes, et al. Prediction of landfalling hurricanes with the advanced hurricane WRF model. *Monthly weather review*, 136(6):1990–2005, 2008.

[23] M. DeMaria and J. Kaplan. A statistical hurricane intensity prediction scheme (SHIPS) for the Atlantic basin. *Weather and Forecasting*, 9(2):209–220, 1994.

[24] M. DeMaria, M. Mainelli, L. K. Shay, J. A. Knaff, and J. Kaplan. Further improvements to the statistical hurricane intensity prediction scheme (SHIPS). *Weather and Forecasting*, 20(4):531–543, 2005.

[25] H. Dikkers and L. Rothkrantz. Support vector machines in ordinal classification: An application to corporate credit scoring. *Neural Network World*, 15(6):491, 2005.

[26] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[27] O. M. Doyle, E. Westman, A. F. Marquand, P. Mecocci, B. Vellas, M. Tsolaki, I. Kłoszewska, H. Soininen, S. Lovestone, S. C. Williams, et al. Predicting progression of Alzheimer's disease using ordinal regression. *PloS one*, 9(8):e105542, 2014.

[28] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1996.

[29] E. Eslami, Y. Choi, Y. Lops, and A. Sayeed. A Deep Learning Driven Improved Ensemble Approach for Hurricane Forecasting. *2019 ESIP Winter Meeting*, 2019.

[30] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[31] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. Advances in knowledge discovery and data mining. 1996.

[32] F. Fernandez-Navarro, P. Campoy-Munoz, C. Hervas-Martinez, X. Yao, et al. Addressing the EU sovereign ratings using an ordinal regression approach. *IEEE transactions on cybernetics*, 43(6):2228–2240, 2013.

[33] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 63–72. ACM, 1999.

[34] S. J. Gaffney, A. W. Robertson, P. Smyth, S. J. Camargo, and M. Ghil. Probabilistic clustering of extratropical cyclones using regression mixture models. *Climate dynamics*, 29(4):423–440, 2007.

[35] S. Gao, P. Zhao, B. Pan, Y. Li, M. Zhou, J. Xu, S. Zhong, and Z. Shi. A nowcasting model for the prediction of typhoon tracks based on a long short term memory neural network. *Acta Oceanologica Sinica*, 37(5):8–12, 2018.

[36] G. Georgoulas, S. Kolios, P. Karvelis, and C. Stylios. Examining nominal and ordinal classifiers for forecasting wind speed. In *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, pages 504–509. IEEE, 2016.

[37] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD*, pages 330–339. ACM, 2007.

[38] S. Giffard-Roisin, M. Yang, G. Charpiat, B. Kégl, and C. Monteleoni. Deep Learning for Hurricane Track Forecasting from Aligned Spatio-temporal Climate Datasets. In *Modeling and decision-making in the spatiotemporal domain NIPS workhop*, 2018.

[39] S. Giffard-Roisin, M. Yang, G. Charpiat, B. Kégl, and C. Monteleoni. Fused deep learning for hurricane track forecast from reanalysis data. In *Climate Informatics Workshop Proceedings 2018*, 2018.

[40] S. Giffard-Roisin, M. Yang, G. Charpiat, C. Kumler Bonfanti, B. Kégl, and C. Monteleoni. Tropical cyclone track forecasting using fused deep learning from aligned reanalysis data. *Frontiers in Big Data*, 3:1, 2020.

[41] S. Gopalakrishnan, Q. Liu, T. Marchok, D. Sheinin, N. Surgi, R. Tuleya, R. Yablonsky, and X. Zhang. Hurricane Weather Research and Forecasting (HWRF) model scientific documentation. *L Bernardet Ed*, 75:7655, 2010.

[42] J. S. Greenfeld. Matching GPS observations to locations on a digital map. In *81th annual meeting of the transportation research board*, volume 1, pages 164–173, 2002.

[43] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li. Incremental support vector learning for ordinal regression. *IEEE Transactions on Neural networks and learning systems*, 26(7):1403–1416, 2014.

[44] J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 35–42. ACM, 2006.

[45] J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of motion patterns in spatio-temporal data sets. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 250–257. ACM, 2004.

[46] P. A. Gutiérrez, S. Salcedo-Sanz, C. Hervás-Martínez, L. Carro-Calvo, J. Sánchez-Monedero, and L. Prieto. Ordinal and nominal classification of wind speed from synoptic pressurepatterns. *Engineering Applications of Artificial Intelligence*, 26(3):1008–1015, 2013.

[47] J. E. Hershberger and J. Snoeyink. *Speeding up the Douglas-Peucker line-simplification algorithm*. University of British Columbia, Department of Computer Science, 1992.

[48] T. F. Hogan, M. Liu, J. A. Ridout, M. S. Peng, T. R. Whitcomb, B. C. Ruston, C. A. Reynolds, S. D. Eckermann, J. R. Moskaitis, N. L. Baker, et al. The navy global environmental model. *Oceanography*, 27(3):116–125, 2014.

[49] H. Jeung, H. T. Shen, and X. Zhou. Convoy queries in spatio-temporal databases. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1457–1459. IEEE, 2008.

[50] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[51] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296. IEEE, 2001.

[52] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404. IEEE, 2017.

[53] K.-j. Kim and H. Ahn. A corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach. *Computers & Operations Research*, 39(8):1800–1811, 2012.

[54] S. Kim, H. Kim, J. Lee, S. Yoon, S. E. Kahou, K. Kashinath, and M. Prabhat. Deep-hurricane-tracker: Tracking and forecasting extreme climate events. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1761–1769. IEEE, 2019.

[55] Y.-J. Kim and M. Chi. Temporal Belief Memory: Imputing Missing Data during RNN Training. In *In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-2018)*, 2018.

[56] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[57] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 140–149. IEEE, 2008.

[58] J.-G. Lee, J. Han, X. Li, and H. Gonzalez. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

[59] R. S. Lee and J. N. Liu. Tropical cyclone identification and tracking system using integrated neural oscillatory elastic graph matching and hybrid RBF network track mining techniques. *IEEE Transactions on Neural Networks*, 11(3):680–689, 2000.

[60] W.-C. Lee and J. Krumm. Trajectory preprocessing. In *Computing with spatial trajectories*, pages 3–33. Springer, 2011.

[61] X. Li, J. Han, S. Kim, and H. Gonzalez. Roam: Rule-and motif-based anomaly detection in massive moving object data sets. In *SDM*, pages 273–284. SIAM, 2007.

[62] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734, 2010.

[63] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD*, pages 1099–1108. ACM, 2010.

[64] W. Mathew, R. Raposo, and B. Martins. Predicting future locations with hidden Markov models. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 911–918. ACM, 2012.

[65] P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 42(2):109–127, 1980.

[66] N. Meratnia and A. Rolf. Spatiotemporal compression techniques for moving point objects. In *International Conference on Extending Database Technology*, pages 765–782. Springer, 2004.

[67] A. Mohamed and K. Schwarz. Adaptive Kalman filtering for INS/GPS. *Journal of geodesy*, 73(4):193–203, 1999.

[68] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD*, pages 637–646. ACM, 2009.

[69] M. Moradi Kordmahalleh, M. Gorji Sefidmazgi, and A. Homaifar. A Sparse Recurrent Neural Network for Trajectory Prediction of Atlantic Hurricanes. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 957–964. ACM, 2016.

[70] M. Mudigonda, S. Kim, A. Mahesh, S. Kahou, K. Kashinath, D. Williams, V. Michalski, T. O'Brien, and M. Prabhat. Segmenting and tracking extreme climate events using neural networks. In *Deep Learning for Physical Sciences (DLPS) Workshop, held with NIPS Conference*, 2017.

[71] W. Y. Ochieng, M. A. Quddus, and R. B. Noland. Map-matching in complex urban road networks. 2003.

[72] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 863–868. ACM, 2008.

[73] M. Pérez-Ortiz, M. Cruz-Ramírez, M. D. Ayllón-Terán, N. Heaton, R. Ciria, and C. Hervás-Martínez. An organ allocation system for liver transplantation based on ordinal regression. *Applied Soft Computing*, 14:88–98, 2014.

[74] O. Pink and B. Hummel. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 862–867. IEEE, 2008.

[75] M. A. Quddus, R. B. Noland, and W. Y. Ochieng. A high accuracy fuzzy logic based map matching algorithm for road transport. *Journal of Intelligent Transportation Systems*, 10(3):103–115, 2006.

[76] B. J. Reich, M. Fuentes, et al. A multivariate semiparametric Bayesian spatial modeling framework for hurricane surface wind fields. *The Annals of Applied Statistics*, 1(1):249–264, 2007.

[77] M. Rüttgers, S. Lee, S. Jeon, and D. You. Prediction of a typhoon track using a generative adversarial network and satellite images. *Scientific reports*, 9(1):1–15, 2019.

[78] S. Stewart and R. Berg. Tropical Cyclone Report: Hurricane Florence. *National Hurricane Center*, 2019.

[79] P.-N. Tan et al. *Introduction to data mining*. Pearson Education India, 2007.

[80] G. A. Vecchi, M. Zhao, H. Wang, G. Villarini, A. Rosati, A. Kumar, I. M. Held, and R. Gudgel. Statistical–dynamical predictions of seasonal North Atlantic hurricane activity. *Monthly Weather Review*, 139(4):1070–1082, 2011.

[81] P. Vickery, P. Skerlj, and L. Twisdale. Simulation of hurricane risk in the US using empirical track model. *Journal of structural engineering*, 126(10):1222–1237, 2000.

[82] D. Wang, B. Liu, P.-N. Tan, and L. Luo. OMuLeT: Online Multi-Lead Time Location Prediction for Hurricane Trajectory Forecasting. In *Proceedings of 34th AAAI Conference on Artificial Intelligence*, 2020.

[83] S. Wang and R. Toumi. Recent migration of tropical cyclones toward coasts. *Science*, 371(6528):514–517, 2021.

[84] J. Xu, P.-N. Tan, and L. Luo. ORION: Online Regularized multI-task regressiON and its application to ensemble forecasting. In *ICDM*, pages 1061–1066. IEEE, 2014.

[85] X. Yang, L. Tang, X. Zhang, and Q. Li. A data cleaning method for big trace data using movement consistency. *Sensors*, 18(3):824, 2018.

[86] H. Yin and O. Wolfson. A weight-based map matching method in moving objects databases. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 437–438. IEEE, 2004.

[87] J. J.-C. Ying, W.-C. Lee, and V. S. Tseng. Mining geographic-temporal-semantic patterns in trajectories for location prediction. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):2, 2013.

[88] G. Yuan, S. Xia, L. Zhang, Y. Zhou, and C. Ji. Trajectory outlier detection algorithm based on structural features. *Journal of Computational Information Systems*, 7(11): 4137–4144, 2011.

[89] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL*, pages 99–108. ACM, 2010.

[90] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD*, pages 316–324. ACM, 2011.

[91] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Trans. Knowl. Data Eng.*, 25(1):220–232, 2013.

[92] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on knowledge and data engineering*, 25 (10):2390–2403, 2012.

[93] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on knowledge and data engineering*, 25 (10):2390–2403, 2013.

[94] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li. iBAT: detecting anomalous taxi trajectories from GPS traces. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 99–108. ACM, 2011.

[95] Z. Zhang and T. Krishnamurti. A perturbation method for hurricane ensemble predictions. *Monthly Weather Review*, 127(4):447–469, 1999.

[96] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. GeoLife2. 0: a location-based social networking service. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pages 357–358. IEEE, 2009.

[97] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding transportation modes based on GPS data for web applications. *ACM Transactions on the Web (TWEB)*, 4 (1):1, 2010.

[98] Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.