Title

Predicting Go-around Occurrence with Input-Output Hidden Markov Model

Journal

Proceedings of the 9th International Conference on Research in Air Transportation (ICRAT 2020)

Authors

Dai, Lu
Liu, Yulin
Hansen, Mark

Publication Date

2020-09-01

Peer reviewed

Predicting Go-around Occurrence with Input-Output Hidden Markov Model

Lu Dai, Yulin Liu, Mark Hansen
Department of Civil and Environmental Engineering
University of California, Berkeley
Berkeley, California, United States
dailu@berkeley.edu, liuyulin101@berkeley.edu, mhansen@ce.berkeley.edu

***Abstract*—In this work, we propose a probabilistic graphical model – Input-Output Hidden Markov Model (IO-HMM) – to make sequential predictions of go-around probabilities for a flight approaching its destination airport. We compare the performance of the IO-HMM against four popular machine learning models trained at every nautical mile to the landing runway threshold on a collection of metrics. Our experiment with approximately 100,000 flights in the JFK airport suggests that the IO-HMM in general outperforms other models due to its capability of capturing the inherent temporal structure of the entire flight sequence.**

Keywords - go-around prediction; sequence classification; imbalanced data; machine learning; Input-Output Hidden Markov Model

I. INTRODUCTION

A go-around is an aborted landing of an aircraft that is on the final approach due to adverse conditions such as wind shear, runway incursion, and unstabilized approach. While a go-around often reduces risk compared to proceeding with the landing, it also significantly degrades the efficiency of the airspace system with respect to airport throughput [1], flight on-time performance [2], air traffic controllers' workload [3] and noise abatement [4]. However, if the probability of a go-around can be predicted in advance and be imparted to operators (e.g. air traffic controllers and pilots) in-time, more proactive mitigations can be taken to alleviate the operational risks and increase airport efficiency as go-arounds might be averted. Therefore, this paper aims to provide a go-around prediction tool that can incorporate the operational conditions and environmental measures for making the real-time prediction of go-arounds. This work also enables a real-time system-wide safety assurance that is applicable to NASA's In-Time System-Wide Safety Assurance (ISSA) strategic trust initiative and builds a practical safety-enhancing risk detection tool for air navigation service providers (ANSPs), airports, airlines, and other ISSA stakeholders.

There is little work on predicting go-arounds in the NAS system. MITRE [5] made the first attempt to predict go-arounds present to departing aircraft by introducing the arrival-departure window (ADW). The ADW is a region along the approach path that must be free of arrival aircraft before a departure is released. It is generated by running a Monte Carlo simulation millions of times for each intersecting or converging scenario. A go-around is thus classified if the Near mid-air collision is detected through simulation. Dai et al. [6] developed a trajectory-based go-around detection algorithm to label go-around flights for JFK arrivals. They identified features that affect go-around occurrence using principal component logistic regression and conducted a counterfactual analysis to quantify the contribution of different features. They found that visibility, ceiling, and flying the glideslope path are the most salient determinants in reducing go-arounds. Bro [7] investigate the utility of neural networks in predicting go-arounds using 2000 hours of general aviation training flight data. In their work, the features are extracted to create a *snapshot* of an aircraft's parameters at 200 feet above ground level on approach. However, a flight approach procedure is a time-series sequence. Prediction problems involving sequentially structured data cannot be effectively dealt with by models which only take one snapshot of features in the process. A predictive model for go-arounds should leverage time series of relevant data. In this paper, go-around prediction is formulated as a sequence prediction problem that predicts the next go-around label of a sequence based on the previously observed information.

There are two common approaches to sequence prediction: a) use point features extracted from the flight sequence to build corresponding machine learning models at every timestamp; and b) use sequence features for temporal models that are learned from the inherent temporal structures of the entire flight sequence. For the applications of using multiple static models, Martinez et al. [8] apply gradient boosting methods to predict the actual runway occupancy times at Vienna airport. For each flight sequence, the instant in which the prediction is made has been set to [10, 9.5, 9, …, 2.5, 2] nautical miles before the landing runway threshold – every 0.5nm between the landing runway threshold and 10nm from that. With the similar regime, Dai and Hansen [9] apply regularized linear models and random forest model to predict the Runway Occupancy Buffer (ROB), which is the time difference between the runway threshold crossing time of the trailing aircraft and the runway exit time of the leading aircraft, when the trailing aircraft is at [10, 9, …, 3, 2] nm before the landing runway threshold.

Among many temporal models, Hidden Markov Model (HMM) . was first introduced by Rabiner and Juang [10] as generative models to classify speech signals and have since become a standard method for sequential predictions. In aviation research, HMM has gained more attention in recent years, especially in predicting flight trajectories. Ayhan and Samet [11] propose an HMM to predict a full 4D trajectories, given the observed weather

This research is sponsored by NASA (grant number: NNA16BE49C).

cube sequence (temperature and wind). In this model, the position of the aircraft is regarded as hidden states, and the weather cube observed around the track point is a realization of such a hidden state. To evaluate airport system operational behavior for safety control, Rodriguez-Sanz et al. [12] has applied the HMM framework to access airspace-airside turnaround operations . This paper defines hidden states according to performance thresholds (e.g., delay, throughput) and expert knowledge, but lacks sensitivity analysis for the selected threshold values to validate model robustness.

The IO-HMM is a special graphical model and was firstly proposed by Bengio and Frasconi [13] for learning problems involving sequentially structured data. While it is originated from HMM, it has a much more flexible architecture, which can be interpreted as a statistical model for target propagation [14] based on the Expectation-Maximization (EM) algorithms. HMMs adjust their parameters using unsupervised learning, whereas the IO-HMM uses EM in a supervised fashion. Experiments on artificial tasks [15] have shown that IO-HMM, which uses EM recurrent learning, can deal with time dependencies more effectively than backpropagation through time and other alternative algorithms. In this paper, we implement a variant of the IO-HMM that allows us to fully exploit both input and output portions of the flight sequence data, as required by the go-around prediction task.

The main contributions of this paper are as follows: First, this research is the first attempt for the real-time go-around prediction task using sequential modeling techniques. Second, we develop an input-output hidden Markov model (IO-HMM) to make ***sequential predictions*** of the go-around probabilities for a flight. Third, we benchmarked the IO-HMM performance against a wide range of machine learning models (random forest, logistic regression, etc.) trained at every nautical mile independently on a collection of metrics, including sensitivity, precision, etc.

This paper is organized as follows. Section II introduces the go-around detection algorithm and feature engineering. Section III describes formalized the methodology we use, and Section VI uses the real-world dataset to show the experimental result of the go-around prediction. Conclusions and future work are proposed in Section V.

II. Go-Around Detection and Feature Engineering

A. Identifying Go-arounds

We obtained raw flight trajectory data that do not specifically label go-around flights. Therefore, to predict a go-around occurrence, we first apply the trajectory-based go-around detection algorithm introduced by Dai et al. [6] to the JFK arrival flights from July 1st to December 24th in 2018. Each flight trajectory will be processed and must meet the criteria described in the paper to be considered as a go-around. After data cleaning and matching, we have identified and validated 371 go-arounds within the period, which accounts for 0.371% of total JFK arrivals. Notice that the algorithm can not only detect the go-around flights as in binary responses but also identify when and where the go-around occurred. Therefore, for each labeled go-around flight, we used the timestamp/trackpoint at the start of ascent as the initial time/location for the go-around procedure, and further truncated the trajectory after that point. As a result, each flight, with a sequence of track points, will have a sequence of go-around labels as either $G_{NGA} = [0,0,0,\dots,0]$ or $G_{GA} = [0,0,\dots,0,1]$, where G_{NGA} is the label sequence for non-go-around flights, and G_{GA} is for the go-around flights.

B. Subsampling Flight Sequence

In the original dataset, the flight tracks were recorded at (roughly) every 5 seconds, which constitutes a dense representation of the trajectories in the terminal airspace. To simplify the analysis, we have applied linear extrapolation to discretize the flight trajectories at every nautical mile away from the landing runways, and only kept the trajectories within the 10-nautical-mile terminal region (Flights beginning apparent go-arounds outside of the 10nm range will not be considered). We hereafter define those points at every nautical mile away from the landing runway thresholds as *information cutoff gates*, and we will only derive features at those gates. Therefore, each flight trajectory considered to be a go-around can be represented by *at most* ten track points, which ends before the flight altitude increases. Only the last closest information gate is labeled as $G_d = 1$. For example, consider a flight that initiated go-around at 5.3 nautical miles from the landing runway threshold. This flight sequence spans from 10 nautical miles to 6 nautical miles, which means that 6-nautical-mile to 10-nautical-mile information gates will be considered for feature engineering. Information for 1-nautical-mile to 5-nautical-mile gates will not be considered.

C. Data Source and Features

We limit the scope of the study to the JFK airport, and our datasets come from three sources ranging from July 1st to December 24th in 2018, except for four days with incomplete data.

The first dataset is retrieved from the Integrated Flight Format (IFF) and Reduced Data (RD) summary of the NASA Sherlock Data Warehouse. Fields of interest include flight summary (e.g., aircraft type, origin, destination), and track points (timestamp, latitude, longitude, altitude, ground speed, etc.). The RD summary and the IFF data have been further processed and merged on a daily basis for each flight arriving at JFK. This dataset has also been used to derive flight approach features, and in-trail separation features described in TABLE I. Interested readers may refer to [6] for details.

The second dataset, airport surface detection equipment Model X (ASDE-X) data, allows us to determine the position of aircraft and ground support equipment in the airport surface area. Each record of raw surface track data contains the timestamp, latitude, longitude, altitude, and groundspeed. This dataset will be used to derive surface operation features – runway occupancy time and the counts of objects on the runway – as illustrated in [9].

The third dataset, which comes from FAA aviation system performance metrics (ASPM), provides airport level configuration and weather information every quarter-hour. We use this dataset to derive the airport and weather features by

matching flight-level operation with the 15-minute interval weather information according to the time at which aircraft reaches a certain distance from the landing runway threshold.

TABLE I. Model Variables and Summary Statistics

Category	Variable	Description	Mean
Flight-specific Characteristics	Airline	1 if flight i is operated by an international airline, 0 otherwise	0.21
	Body	1 if flight i is wide-body aircraft, 0 otherwise	0.24
	Runway	Dummy variable for landing runway of flight i	
Go-around Clustering Effect Features	GaGap	The minimal time interval between the approaching time of flight i and the initiation time of all (other) go-arounds occurred in the past 24 hours (in minutes)	712.61
	GaCnt	The number of go-arounds occurred in the past 30 minutes	0.06
Flightpath Profile Features	Angle	Angle with the Extended Runway Centerline (in degree)	7.99
	AltDev	Absolute altitude deviation from 3-degree glideslope (in 100 feet)	1.52
	Speed	Flight groundspeed (in knots)	163.19
	Energy	Kinetic energy height (in feet)	2841.45
Flight In-trail Relationship Features	LOS	The loss of separation between leading and trailing flight i (in nautical miles)	0.07
	SpeedDiff	Groundspeed difference between leading and trailing flight i (in knots)	20.92
	AltDiff	The altitude difference between leading and trailing flight i (in 100 feet)	13.30
	NoLead	1 if there is no leading aircraft in front of flight i within 10-minute landing sequence, 0 otherwise	0.13
Airport and Weather Condition Features	ArrQue	Difference between airport supplied arrival rate (AAR) and the number of intended landing aircraft (counts)	-2.04
	DepQue	Difference between airport supplied departure rate (ADR) and the number of intended depart aircraft (counts)	1.24
	Hour	Dummy variable for each hour of the day in local time	-
	RwyChange	1 if the used runway configuration is changed, otherwise 0	0.08
	Wind	Wind speed where the headwind component is subtracted (in knots)	5.72
	Visibility_i	Discretized visibility (i = 1,2,3; intervals are [0, 3], [3, 5], [5, 10] in miles)	-
	Ceiling_i	Discretized ceiling (i =1, 2, 3, 4; intervals are [0, 5), [5, 10), [10, 30), [30, 100] (in 100 feet)	-
	VMC	1 for VMC, 0 for IMC	0.86
Surface Operation Features	ROB	The predicted time difference between the runway threshold crossing time of the trailing aircraft and the runway exit time of the leading aircraft (seconds)	49.78
	RwyCnt	The number of aircraft and vehicles appearing on the landing runway (counts)	2.04

According to the literature, theoretical studies, and data availability, we derive two groups of features at each information cutoff gate – ***static features*** and ***dynamic features***. The static features, which contain aircraft type, operated airline, and landing runways, will not change during the final approach; the dynamic features, such as flight altitude and speed, vary along with the approach. These features, summarized in TABLE I. , are standardized to handle varying magnitudes of the feature values and improve the convergence speed of the model. After data cleaning and matching, there are 100,032 arrivals in the JFK airport during the analysis period.

D. Problem Statement

For each flight, given the sequence of the *realized* track points and their associated feature vectors, we predict whether it will initiate a go-around prior to the next track point. This task can, therefore, be viewed as a "many-to-one" sequential prediction problem, where "many" suggests that the observed feature space could involve more than one feature vector, and "one" indicates that our final prediction based on those "many" features will be one label indicating whether it is a go-around or not.

III. METHODOLOGY

As the flight approaches to the airport, our goal is to predict whether the flight will initiate a go-around or not, given the information we can gather from the *realized* flight trajectory. Such a conditional dependence structure between random variables is usually expressed in the probabilistic graphical model, which uses a graph-based representation as to the foundation for encoding the distribution over a multi-dimensional space. Two types of graphical representations of distributions are commonly used - Bayesian networks and Markov random fields. Both HMMs and RNNs are special cases of Bayesian networks, of which the network structure is a directed acyclic graph. A toy graph example of a standard HMM is shown in Figure 1. , where nodes represent variables, and edges represent transitions and dependence relationships. Specifically, the white nodes in the Figure suggest hidden states which are typically not observed, and the blue nodes represent the visible output emitted by those hidden states. The underlying probabilistic transitions between hidden states are termed as transition probability and are denoted as edges between the white nodes. The transitions between hidden states and output states are called emission probability and are represented by the edges between white and blue nodes. In this paper, we propose a novel input-output hidden Markov model (IO-HMM), an extension to the HMM that can better capture the sequential structure inherent in our problem to model and predict the go-around occurrence for an approaching flight.

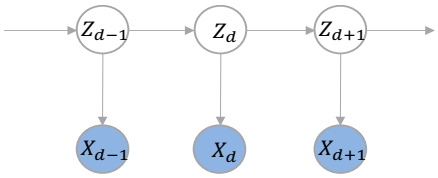

Figure 1. HMM Architecture

A. IO-HMM

1) Model architecture

In order to deal with the go-around prediction problem, we regard the flight approach procedure as a discrete state dynamical process based on the following state-space description:

$$z_d = f(z_{d-1}, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}) \quad (1)$$

$$[\boldsymbol{x_d}, G_d] = g(z_d, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}) \quad (2)$$

Equation (1) describes the transition function between different states, where $z_d \in \{1, 2, \dots, s\}$ is a discrete hidden state variable that encodes symbols representing flight approach status (locations, speeds, etc.), s is the total number of hidden states, which is set *a priori* for IO-HMM. Researchers typically associate those hidden states with semantics. In our specific context, those hidden states can be representations of how stable the flight approaches are.

$\boldsymbol{u_d}$ is the input variable at information cutoff gate d, including contextual information that can be known before the aircraft reaches the information gate d, such as flight-specific characteristics (e.g., operated airline, aircraft type, landing runway), weather conditions (e.g., visibility, ceiling, wind speed), and airport information (e.g., airport arrival rate, the change of runway configuration). Incorporating the input vector, which is the specialty of the IO-HMM, the model can leverage distributed implementation and sharing parameters across multiple flight subgroups (e.g., flight sequences with the same aircraft type). In other words, flight sequences in the same subgroup have similar geographical and structural contextual variables, share the same estimated parameters, thus will have a high probability of being "clustered" in the same approach states.

Equation (2) describes the emission function among the output variables $[\boldsymbol{x_d}, G_d]$ and the state variables z_d. To be more specific, in the equation, $[\boldsymbol{x_d}, G_d]$ are both the output variable at information cutoff gate d. $\boldsymbol{x_d}$ contains dynamic features described in TABLE I. such as flight altitude and loss of separation. G_d is the go-around label obtained from the go-around detection algorithm in Section II.A. According to equation (2), the G_d is determined by the current state of the system z_d, input features at the current information cutoff gate $\boldsymbol{u_d}$and the lag-one output features in the previous information cutoff gate $\boldsymbol{x_{d-1}}$. The output variables are available only after the aircraft passes information cutoff gate d. In contrast to the input variables, the output variables contain information that is not available at the transition to a new approach state. In other words, output variables can be observed when training the models but must be inferred when we predict the go-around probability. As dynamic features from the previous information cutoff gates also contribute to part of the context information for predicting what will happen to the aircraft in the next information cutoff gate, we link the lag-one output variables $\boldsymbol{x_{d-1}}$ to the next information gate by incorporating $\boldsymbol{x_{d-1}}$ in the input vector layer.

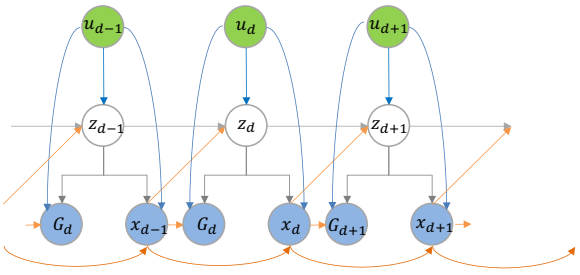

Figure 2. IO-HMM Architecture

Such discrete state dynamical system defined by equation (1) (2) can be modeled by the graph depicted in Figure 2. , in which the white nodes represent hidden state variables z_d, the green nodes represent observed input variables $\boldsymbol{u_d}$, and the blue nodes contain output variables $\boldsymbol{x_d}$ and G_d.

2) Model specification

In this paper, we assume a multinomial distribution for the state variable z_d and use a Bayesian network to characterize the probabilistic dependencies among these state variables, input variables, and output variables. The IO-HMM captures the dynamics of the Markovian chain by using the current inputs and current state distribution to estimate the state variable and the output variable distributions for the next information cutoff gate. According to the connections shown in the IO-HMM graph shown in Figure 2. , three probability models need to be specified: an initial probability model which outputs a vector of initial state probabilities at the start of sequence, a transition probability model conditioned on the input sequence which outputs a square matrix of state transition probabilities at each information cutoff gate, and an emission probability (a.k.a. output probability) model governing the distribution of the output variables—including go-around probability--at a particular time given the hidden state and input features. Model details and formulas are described in the following sections.

a) Initial model: We develop a multinomial logistic regression model to estimate the initial probability parameters $\widehat{\boldsymbol{\theta}}_{\boldsymbol{initial}}$.

$$P(z_1 = i \mid \boldsymbol{u_1}; \boldsymbol{\theta_{initial}}) = \frac{\exp(\theta_{initial_i} \cdot \boldsymbol{u_1})}{\sum_k^s \exp(\theta_{initial_k} \cdot \boldsymbol{u_1})} \quad (3)$$

where i is the state label at the initial ($d = 1$ represents the first information cutoff gate of the sequence, which is 9nm before the landing threshold) information cutoff gate; s is the total number of hidden states.

b) Transition model: At information gate d, the hidden state z_d is related to the input features u_d, lag-one output features x_{d-1}, and the previous hidden state z_{d-1}, subject to some Gaussian noise. The multinomial logistic regression model is used to estimate the transition probability parameters $\widehat{\boldsymbol{\theta}}_{\boldsymbol{trans}}$.

$$P(z_d = j \mid z_{d-1} = i, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) = \frac{\exp(\theta_{trans}{}_i^j \cdot)}{\sum_k^s \exp(\theta_{trans}{}_i^k \cdot \begin{bmatrix} \boldsymbol{u_d} \\ \boldsymbol{x_{d-1}} \end{bmatrix})} \quad (4)$$

where i is the state label at the previous information cutoff gate $d-1$, j is the state label at the current information cutoff gate d; s is the total number of hidden states. The estimated $\widehat{\boldsymbol{\theta}}_{\boldsymbol{trans}} = \begin{bmatrix} \theta_{trans}{}_1^1 & \cdots & \theta_{trans}{}_1^s \\ \vdots & \ddots & \vdots \\ \theta_{trans}{}_s^1 & \cdots & \theta_{trans}{}_s^s \end{bmatrix}$, $\widehat{\boldsymbol{\theta}}_{\boldsymbol{trans}} \in \mathbb{R}^{s \times s}$is the transition probability matrix for the approach state transited from the previous state. The transition probabilities are heterogeneous and depend on contextual information u_d. This can improve the accuracy of state inference.

c) Emission model: To gain interpretability, we choose linear models for continuous outputs represented as Gaussian random variables ($\boldsymbol{x_d}$), thus:

$$P(\boldsymbol{x_d}|\, z_d = j,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{\left(\boldsymbol{x_d} - \boldsymbol{\theta^c_{emis_j}} \cdot \begin{bmatrix} \boldsymbol{u_d} \\ \boldsymbol{x_{d-1}} \end{bmatrix}\right)^2}{2\sigma_j^2} \right) \quad (5)$$

where j is the state label at the current information cutoff gate d. The estimated $\widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis}} = \left[\widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis_1}} \quad \cdots \quad \widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis_s}}\right]$, $\widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis}} \in \mathbb{R}^{m\times s}$ is the emission coefficient matrix where its column $\boldsymbol{\theta^c_{emis_j}}$ denotes the coefficients of m output variables in the linear model when the hidden state is j. σ_j represents the standard deviation of the linear model when the hidden state is j. Therefore, the number of coefficients to be estimated in the emission probability model is equal to the product of the number of hidden states s and the number of output variables m.

For binary random variable G_d, the logistic regression model is used as the output emission model. The probability is as follow:

$$P(G_d = 1|\, z_d = j,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}}) = \frac{1}{1 + \exp\left(-\boldsymbol{\theta^G_{emis_j}} \cdot \begin{bmatrix} \boldsymbol{u_d} \\ \boldsymbol{x_{d-1}} \end{bmatrix}\right)} \quad (6)$$

When implementing the maximum likelihood method, a Gaussian may be fit onto a single data point and lead to a singular covariance matrix. Whenever the covariance matrix is singular, the log-likelihood function will go to infinity. Thus, the maximization of the log-likelihood function is ill-posed. Ridge regularization [16] is employed to objective functions of both the linear regression and the logistic regression. The regularized term C_{ols}, C_{logit} will be fine-tuned from the range specified in TABLE II. using five-fold cross-validation.

Based on the model architecture and model specifications, the likelihood of a sequence in our IO-HMM can be written as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{u}) = \sum\nolimits_s [\, & P(z_1 \mid \boldsymbol{u_1}; \boldsymbol{\theta_{initial}}) \cdot \\ & \prod_{d=2}^{D} P(z_d \mid z_{d-1}, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) \cdot \\ & \prod_{d=1}^{D} P(\boldsymbol{x_d}|\, z_d,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) \cdot \\ & P(G_d = 1|\, z_d,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}})] \end{aligned} \quad (7)$$

The temporal dependency is captured by the transitions between the hidden approach states $\boldsymbol{z}$. The direct dependency between input features and output features can, in principle, compensate for some potential ineffectiveness of the hidden state learned for the current timestamp.

3) Model estimation

As illustrated in the previous section, the IO-HMM includes three groups of unknown parameters to be estimated: initial probability parameters $\boldsymbol{\theta_{initial}}$, transition probability parameters $\boldsymbol{\theta_{trans}}$, and emission probability parameters $\boldsymbol{\theta_{emis}}$. In this study, we implement the Expectation-Maximization (EM) algorithm to optimize the parameter set. Explicitly, in the E-step, we compute the expected value of the complete log-likelihood as in Equation (8), given the observed data and parameters estimated (or initialized) at the previous (or initialization) step. In the M-step, parameters are updated to maximize the expected data log-likelihood.

$$\begin{aligned} Q(\boldsymbol{\theta},\, \boldsymbol{\theta^r}) = & \sum_{i=1}^{s} \gamma_{i,1} ln\, P(z_1 = i \mid \boldsymbol{u_1}; \boldsymbol{\theta_{initial}}) + \\ & \sum_{d=2}^{D} \sum_{i=1}^{s} \sum_{j=1}^{s} \xi_{ij,d} \ln \mathrm{P}(z_d = j \mid z_{d-1} = i, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) \\ & + \sum_{d=1}^{D} \sum_{i=1}^{s} \gamma_{i,d} \left[\ln P(\boldsymbol{x_d}|\, z_d = j,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) \right. \\ & \left. + \ln P(G_d = 1|\, z_d = j,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}}) \right] \end{aligned} \quad (8)$$

where r represents the iteration; s is the total number of hidden states; D is the total number of information cutoff gates in each flight sequence; $\boldsymbol{u_d}, \boldsymbol{x_{d-1}}, \boldsymbol{x_d}$, G_d, and z_d are the input variables, lag-one output variables, output variables, go-around binary labels and hidden state variables at information cutoff gate d, which were introduced in Section III.A.a; $\boldsymbol{\theta_{initial}}, \boldsymbol{\theta_{trans}}, \boldsymbol{\theta^c_{emis}}, \boldsymbol{\theta^G_{emis}}$ are parameters to be estimated in initial probability model, transition probability model, and emission probability model which were discussed in Section III.A.b.

$\xi_{ij,d}$ is the posterior transition probability, which defines the probability of being in state i at information cutoff gate d and state j at information cutoff gate $d+1$. $\gamma_{i,d}$ is the posterior state probability for state i at information cutoff gate d. $\xi_{ij,d}$ and $\gamma_{i,d}$ are computed from forward probability α and backward probability β under the forward-backward algorithm [17], which involves three steps:

- Computing the forward probability which provides the probability of ending up in any particular state i given the first d observations in the sequence;
- Computing the backward probability which provides the probability of seeing the observations from information cutoff gate $d+1$ to the end given we are in a particular state at the time;
- These two sets of probabilistic distributions can then be combined to obtain the distribution over states at any specific point $d \in \{1, \cdots, D\}$ given the entire observations sequence.

$$\begin{aligned} \xi_{ij,d} = & P(z_d = i \mid z_{d-1} = j, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) \cdot \alpha_{i,d} \cdot \beta_{j,d} \cdot \\ & P(\boldsymbol{x_d}|\, z_d = j,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) \cdot \\ & P(G_d = 1|\, z_d = k,\, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}}) / \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{u}) \end{aligned} \quad (9)$$

$$\gamma_{i,d} = \frac{\alpha_{i,d}\beta_{i,d}}{\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{u})} \quad (10)$$

In summary, the forward-backward algorithm starts with some initial estimates of the IO-HMM parameters $\boldsymbol{\theta_0}$. We then iteratively run the Expectation-step and the Maximization-step. In the E-step, we compute the expected value of the complete data log-likelihood, the posterior state probability $\gamma_{i,d}$, and the posterior transition probability $\xi_{ij,d}$ for each training sequence, given the entire observed data sequence and parameters estimated at the previous (or initialized) step. In the M-step, we use the computed distribution over states to update the transition probability matrix and the emission likelihood matrix for maximizing the expected data likelihood.

B. Conventional Classification

TABLE II. Hyperparameters for Predictive Models

Model	Hyperparameter	Search Range
Logistic Regression	Penalty term for the l_2 regularization	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
SVM	Penalty term for the l_2 regularization	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
Random Forest Regression	The maximal depth of the tree	[5, 10, 15, 20]
	The minimal number of samples required to split an internal node	[2, 5, 10, 15, 20]
XGBoost	Learning rate	[10^{-6},10^{-5},10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.3, 0.8]
	Minimum loss reduction required to make a further partition on a leaf node of the tree	[1, 3, 5, 10, 15, 20]
	Maximum depth of a tree	[6, 10, 15, 20]
	The maximum number of steps we allow each leaf output to be. It might help when class is extremely imbalanced	[2, 6, 10, 15, 20]
	l_2 regularization term on weights	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
	Subsample ratio of the training instances	[0.2, 0.4, 0.6, 0.8, 0.9, 1]
IO-HMM	The l2 regularized term in linear regression	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
	The l2 regularized term in logistic regression	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
	The number of hidden states	[2, 3, 4, 5, 6]

To further evaluate the performance of the proposed IO-HMM model, we compare it against four classical machine learning algorithms using a similar modeling regime in [18]. The algorithms include logistic regression (LR), support vector machine (SVM), and two tree-based methods – random forest (RF) and extreme gradient boosting (XGBoost). We implement the linear SVM since nonlinear kernels require significantly more computational complexity to train with little to no gain in predictive performance on our large dataset. Since these four algorithms do not explicitly handle the sequential prediction tasks, we assume that each information cutoff gate is independent of each other and develop a model at each gate. Specifically, we have built models at 9-nautical-mile, 8-nautical-mile, …, 2-nautical-mile information gate, and each model only used the feature vector at which distance information gate it was in. The underlying question we wish to answer is that, by solely observing the current flight and weather conditions, how well a classical machine learning model learns to predict whether a flight is going to initiate a go-around in the future?

Lastly, to balance bias and variance, we have fine-tuned the hyper-parameters for each model using five-fold cross-validation. TABLE II. summarizes the descriptions of hyper-parameters and their ranges.

IV. Experimental Results

A. Handling Imbalanced Data

In our go-around dataset, we have 371 go-arounds out of 100,032 arrival flights. This dataset is extremely imbalanced with go-around flights significantly less than those non-go-around flights - only 0.3% of arrivals are go-around flights. With so few go-arounds relative to non-go-arounds, the learning algorithm is often unable to generalize the behavior of the go-around flights well, and tends to classify all observations as majority class; hence the algorithm performs poorly on the minority class. An effective way to make the classification dataset balanced is to downsample the majority class in the training set before learning a classifier. The assumption behind this strategy is that in the majority class, there are many redundant observations, and randomly removing some of them does not change the estimation of the within-class distribution.

TABLE III. Data Statistics

Dist. (nm)	Full dataset (G-A / arrivals)		Downsampled set (G-A / arrivals)	
	Train (80%)	*Test (20%)*	*Train*	*Test*
9	296 / 80,025	75 / 20,007	296 / 3,256	75 / 825
8	293 / 80,022	74 / 20,006	293 / 3,223	74 / 814
7	287 / 80,016	72 / 20,004	287 / 3,157	72 / 792
6	281 / 80,010	71 / 20,003	281 / 3,091	71 / 781
5	273 / 80,002	69 / 20,001	273 / 3,003	69 / 759
4	252 / 79,981	63 / 19,995	252 / 2,772	63 / 693
3	221 / 79,950	56 / 19,988	221 / 2,431	56 / 616
2	180 / 79,909	46 / 19,978	180 / 1,980	46 / 506

Specifically, the downsampling will be used as a pre-processing step to rebalance the two classes before any algorithm is applied. For each distance-variant dataset, we first split the data into a training set (80%) and testing set (20%), which are the 2nd and 3rd sub-column in TABLE III. Second, while the number of go-arounds in both the training set and the testing set remains unchanged, we randomly sample the non-go-around flights without replacement until the non-go-arounds to go-arounds ratio is 10 to 1 (269:1 in the full dataset). We do not set the two classes equally balanced; otherwise, models are prone to overfit and be much more sensitive to outliers due to the small number of data samples. After combining the downsampled majority class with the original minority class, we obtain the balanced dataset, which is reported as the 4th and 5th column in TABLE III. We train, validate, and test the aforementioned models on the full dataset and the downsampled dataset separately. We also apply another treatment to deal with these two imbalanced datasets – penalizing the misclassification on the minority class by an amount proportional to how under-represented it is.

B. Experimental Steps

After pre-processing the imbalanced data and splitting it into 80% of the training set and 20% of the testing set, we then use five-fold cross-validation on the training set to fine-tune the model parameters as shown in TABLE II. based on the F-score. F-score, which is the weighted harmonic mean of the precision and recall, is chosen to be the evaluation criteria through the experiments. Based on the accuracy measures in confusion matrix [19], $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$ are defined. We want to have high confidence that observations predicted as go-arounds are actually correct (high precision), as well as a high detection rate of the go-arounds (high recall). However, high precision comes at the cost of low recall and vice versa. Referring to [1] that the cost of miss detecting a go-around highly outweighs the cost of getting a false alarm warning in the system, we set the parameter, which determines the weight of recall in the F-score, as ***2***.

$$F_2 = \frac{(1+2^2)\cdot precision \cdot recall}{2^2 \cdot precision + recall} \tag{11}$$

Lastly, we fit models using selected parameters on the full training set and make predictions on the testing set. For the conventional classification, each kind of particular machine learning model is trained eight times separately over different distant-variant datasets, using only features up to the current nautical miles. For the sequential classification, the IO-HMM model is trained once for all the 80,025 flight sequences or the 3,256 downsampling flight sequences. In order to compare the model performance, five metrics will be reported: F2 score, the area under the receiver operating characteristic curve (AUC), precision, recall, and accuracy.

C. Model Performance

To save space, TABLE IV. reports the model performance on the testing set of the best baseline model (with maximum F2 score) and the IO-HMM, and the fine-tuned hyperparameters associated with the models are omitted. The corresponding metrics scores are comparable across different models, for a specific information cutoff gate, in the same dataset. For each model comparison, we shade the superior metric score.

For models trained on the full dataset, the IO-HMM consistently outperforms the other models except for the early stage of the flight approach. This suggests that incorporating temporal structures in the model can marginally improve go-around prediction performance. As for the machine learning models, the random forest model is comparable to the IO-HMM, yet it still slightly falls behind the IO-HMM, especially during the later stage of a flight approach process. The bootstrapping and oversampling techniques applied in RF help to overcome the imbalanced class issue and avoid overfitting, thus the model has better predictability.

With a less imbalanced dataset, all the five models have better predictability in terms of F2 score, precision, recall, and AUC. Nevertheless, there is no universal model that performs best for every distance-variant dataset. The performance of IO-HMM is poorer, probably due to the small size of the training data. Other research work [20] has found that the larger the training data, the better the estimation of the probability models (especially for the transition matrix) of the IO-HMM, and the lower the bias in the Markovian predicted residuals. Even though IO-HMM has temporal structures, the small training data makes it incapable of learning a hidden state effective for go-around prediction.

We also notice that the IO-HMM gives a consistent and monotonically increasing performance, in terms of F2 score, as the approach progresses, while other models do not. The machine learning models are trained separately using independent input features, and are incapable of learning a whole flight sequence that is inherently capturing the temporal dependency. For the IO-HMM using the full dataset, the most confident prediction is obtained when the flight is at 2nm before the runway threshold, with a 0.417 recall score and a 0.155 precision score, meaning that 41.7% of go-arounds are correctly classified, and 15.5% of the go-arounds predicted by IO-HMM are correct. We also investigate the predicted probabilities of particular flights labeled as go-arounds in the test set. We observe that the prediction result given by IO-HMM has probabilistic evolutionary changes – the predicted probabilities for these tested go-around flights monotonically increase as the approach progresses. The main reason is that most go-arounds happen closer to the airport. A more generalized investigation is needed to verify this property in further research work.

V. Conclusions and Future Work

For the go-around prediction task, our research reported in this paper is novel. We view it as a sequence classification problem and employ a temporal model – IO-HMM - to solve it. A set of machine learning models serve as the benchmarks of the go-around prediction task. Our extensive experimental result shows that the IO-HMM can capture the time dependency in the flight sequence, giving a consistent performance and probabilistic evolutionary change as the approach progresses.

The main challenge of this go-around prediction task is the imbalanced class distribution of the data. The go-around rate is often very low – FAA reports that the average go-around occurrence across the core 30 airports in the US from 2012 to 2018 is at a rate of 0.4% [21]. Thus, the dataset used for training and validation in this task is typically heavily imbalanced. Future work will be envisaged using specific imbalanced-class mathematical machinery and calibrating the balanced model to correct the bias introduced by downsampling. Thus, it will be able to produce predictions under the true distribution of the classes. Another possible extension to the go-around prediction problem is to apply the Recurrent neural network (RNN) methods such as the long short-term memory (LSTM) network. These kinds of models are also capable of automatically extracting features from past events. However, they apply a point-wise nonlinearity to the output at every timestamp, which Markovian models do not. Based on our findings, nonlinearity may make the networks more expressive and adaptive to learn from real-world data.

TABLE IV. Model Performance

Dist.	Model	Downsampling Dataset					Model	Full Dataset				
		F2 score	*Precision*	*Recall*	*AUC*	*Accuracy*		*F2 score*	*Precision*	*Recall*	*AUC*	*Accuracy*
9	**SVM**	0.610	0.313	0.800	0.849	0.820	**RF**	0.165	0.073	0.240	0.841	0.986
	IO-HMM	0.339	0.110	0.707	0.568	0.453	**IO-HMM**	0.148	0.067	0.213	0.777	0.978
8	**Logistic**	0.589	0.313	0.757	0.829	0.825	**XGBoost**	0.165	0.047	0.446	0.846	0.964
	IO-HMM	0.378	0.215	0.467	0.620	0.795	**IO-HMM**	0.149	0.062	0.230	0.711	0.984
7	**RF**	0.609	0.348	0.750	0.869	0.848	**SVM**	0.185	0.066	0.333	0.828	0.981
	IO-HMM	0.400	0.186	0.562	0.643	0.741	**IO-HMM**	0.153	0.077	0.203	0.728	0.988
6	**RF**	0.615	0.374	0.732	0.869	0.863	**SVM**	0.228	0.091	0.366	0.854	0.985
	IO-HMM	0.440	0.307	0.493	0.651	0.859	**IO-HMM**	0.230	0.092	0.368	0.808	0.985
5	**XGBoost**	0.617	0.425	0.696	0.873	0.885	**RF**	0.247	0.122	0.333	0.856	0.989
	IO-HMM	0.445	0.187	0.680	0.666	0.787	**IO-HMM**	0.255	0.100	0.417	0.828	0.986
4	**SVM**	0.652	0.395	0.778	0.879	0.870	**RF**	0.282	0.131	0.397	0.854	0.990
	IO-HMM	0.453	0.194	0.681	0.653	0.779	**IO-HMM**	0.291	0.117	0.463	0.836	0.989
3	**SVM**	0.663	0.407	0.786	0.883	0.875	**RF**	0.304	0.410	0.286	0.877	0.997
	IO-HMM	0.545	0.367	0.621	0.826	0.894	**IO-HMM**	0.307	0.141	0.435	0.839	0.990
2	**SVM**	0.648	0.407	0.761	0.885	0.875	**RF**	0.284	0.126	0.413	0.848	0.992
	IO-HMM	0.589	0.321	0.745	0.841	0.890	**IO-HMM**	0.312	0.155	0.417	0.861	0.990

REFERENCES

[1] J. Shortle and L. Sherry, "A Model for Investigating the Interaction Between Go-Arounds and Runway Throughput," in *2013 Aviation Technology, Integration, and Operations Conference*, (AIAA AVIATION Forum: American Institute of Aeronautics and Astronautics, 2013.

[2] T. Blajev and C. W. Curtis, "Go-Around Decision-Making and Execution Project," Flight Safety Foundation, 2017.

[3] R.-C. Jou, C.-W. Kuo, and M.-L. Tang, "A study of job stress and turnover tendency among air traffic controllers: The mediating effects of job satisfaction," *Transportation Research Part E: Logistics and Transportation Review,* vol. 57, pp. 95-104, 2013/10/01/ 2013, doi: https://doi.org/10.1016/j.tre.2013.01.009.

[4] X. Prats, V. Puig, J. Quevedo, and F. Nejjari, "Multi-objective optimisation for aircraft departure trajectories minimising noise annoyance," *Transportation Research Part C: Emerging Technologies,* vol. 18, no. 6, pp. 975-989, 2010/12/01/ 2010, doi: https://doi.org/10.1016/j.trc.2010.03.001.

[5] MITRE. "Decision Aid for Controllers Helps Prevent Near-misses." https://www.mitre.org/publications/project-stories/decision-aid-for-controllers-helps-prevent-near-misses (accessed May 31, 2020).

[6] L. Dai, Y. Liu, and M. Hansen, "Modeling Go-around Occurrence," presented at the Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019), Vienna, Austria, 2019.

[7] J. Bro, "FDM Machine Learning: An investigation into the utility of neural networks as a predictive analytic tool for go around decision making," *Journal of Applied Sciences and Arts,* 2017.

[8] D. Martinez, S. Belkoura, S. Cristobal, F. Herrema, and P. Wachter, "A Boosted Tree Framework for Runway Occupancy and Exit Prediction," presented at the Eighth SESAR Innovation Days, 2018.

[9] L. Dai and M. Hansen, "Real-time Prediction of Runway Occupancy Buffer," in *International Conference on Artificial Intelligence and Data Analytics for Air Transportation*, Singapore, 2020: IEEE Xplore Digital Library and Scopus.

[10] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *ieee assp magazine,* vol. 3, no. 1, pp. 4-16, 1986.

[11] S. Ayhan and H. Samet, "Aircraft Trajectory Prediction Made Easy with Predictive Analytics," presented at the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, 2016.

[12] Á. Rodríguez-Sanz, J. M. Cordero, B. R. Fernández, F. Gómez, Comendador, and R. A. Valdés, "Assessment of the Airport Operational Dynamics Using a Multistate System Approach," presented at the USA/Europe Air Traffic Management Research and Development Seminar, Vienna, Austria, 2019.

[13] Y. Bengio and P. Frasconi, "An Input Output HMM Architecture " 1995.

[14] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio, "Difference target propagation," in *Joint european conference on machine learning and knowledge discovery in databases*, 2015: Springer, pp. 498-515.

[15] Y. Bengio and P. Frasconi, "Credit assignment through time: Alternatives to backpropagation," in *Advances in Neural Information Processing Systems*, 1994, pp. 75-82.

[16] R. A. Thisted, *Ridge regression, minimax estimation, and empirical Bayes methods*. Department of Statistics, Stanford University, 1976.

[17] S.-Z. Yu and H. Kobayashi, "An efficient forward-backward algorithm for an explicit-duration hidden Markov model," *IEEE signal processing letters,* vol. 10, no. 1, pp. 11-14, 2003.

[18] Y. Liu, M. Hansen, D. J. Lovell, and M. O. Ball, "Predicting aircraft trajectory choice–a nominal route approach," in *Proceedings of the International Conference for Research in Air Transportation*, 2018.

[19] J. T. Townsend, "Theoretical analysis of an alphabetic confusion matrix," *Perception & Psychophysics,* vol. 9, no. 1, pp. 40-50, 1971.

[20] H. Alshraideh and G. Runger, "Process monitoring using hidden Markov models," *Quality and Reliability Engineering International,* vol. 30, no. 8, pp. 1379-1387, 2014.

[21] FAA, "Air traffic by the numbers," in "FAA Report," 2019.