Machine Learning Applied to Airspeed Prediction During Climb
Richard Alligier, David Gianazza, Nicolas Durand

To cite this version:
D Seminar, Jun 2015, Lisboa, Portugal. <hal-01168664>

HAL Id: hal-01168664
https://hal-enac.archives-ouvertes.fr/hal-01168664
Submitted on 29 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract—In this paper, we apply Machine Learning methods to improve the aircraft climb prediction in the context of ground-based applications. Mass and speed intent are key parameters for climb prediction. As they are considered as competitive parameters by many airlines, they are currently not available to ground-based trajectory predictors. Consequently, most predictors today use reference parameters that may be quite different from the actual ones.

In our most recent paper ([1]), we have demonstrated that Machine Learning techniques provide a mass estimation significantly more precise than two state-of-the-art mass estimation methods.

In this paper, we apply similar techniques to the speed intent. We first build a set of examples by adjusting CAS/Mach speed profile to each climb trajectory in our database. Then, using the adjusted values (\(\hat{\text{CAS}}, \hat{\text{M}}\)) in this database, we learn a model able to predict the (\(\text{CAS}, \text{M}\)) values of a new trajectory, using its past points as input.

We apply this technique to actual Mode-C radar data and we consider 9 different aircraft types. When compared with the reference speed profiles provided by BADA, the reduction of the speed RMSE ranges from 36 % to 79 %, depending on the aircraft type. Using the predicted mass and speed profile, BADA is used to compute the predicted future trajectory with a 10 minute horizon. When compared with BADA used with the reference parameters, the reduction of the future altitude RMSE ranges from 45 % to 87 %.

Keywords: aircraft trajectory prediction, speed intent, BADA, Machine Learning

INTRODUCTION

Trajectory prediction is a key feature to most Air Traffic Management and Control (ATM/ATC) operational concepts. The role of trajectory prediction is even more important in the future concepts and decision support tools envisioned in the European SESAR program ([2]) and its U.S. counterpart NextGen ([3]).

With the implementation of a data-link between aircraft and ground-based systems, one could think that the on-board trajectory prediction could be downloaded to the ground-based system. However, for some applications the ground-based trajectory prediction is still more relevant. Some of the most recent algorithms designed to solve ATM/ATC problems do require to test a large number of alternative trajectories and it would be impractical to download them all from the aircraft.

As an example of such algorithms, in ([4]) an iterative quasi-Newton method is used to find trajectories for departing aircraft, minimizing the noise nuisance. Another example is ([5]) where Monte Carlo simulations are used to estimate the risk of conflict between trajectories in a stochastic environment. Some of the automated tools currently being developed for ATM/ATC can detect and solve conflicts between trajectories (see ([6]) for a review). These algorithms may use Mixed Integer Programming ([7]), Genetic Algorithms ([8], [9]), Ant Colonies ([10]), or Differential Evolution or Particle Swarm Optimization ([11]) to find optimal solutions to air traffic conflicts.

To be efficient, all these methods require a fast and accurate trajectory prediction, and the capability to test a large number of “what-if” trajectories. Such requirements forbid the sole use of on-board trajectory prediction, which is certainly the most accurate, but which is not directly available to ground systems.

Most trajectory predictors rely on a point-mass model to describe the aircraft dynamics. The aircraft is simply modeled as a point with a mass, and the second Newton’s law is applied to relate the forces acting on the aircraft to the inertial acceleration of its center of mass. Such a model is formulated as a set of differential algebraic equations that must be integrated over a time interval in order to predict the successive aircraft positions, knowing the aircraft initial state (mass, current thrust setting, position, velocity, bank angle, etc.), atmospheric conditions (wind, temperature), and aircraft intent (thrust profile, speed profile, route).

Unfortunately, the data that is currently available to ground-based systems for trajectory prediction purposes is of fairly poor quality. The speed intent and aircraft mass, being considered competitive parameters by many airline operators, are not transmitted to ground systems. The actual thrust setting of the engines (nominal, reduced, or other, depending on the throttle’s position) is unknown. Weather and Radar data are uncertain. The problem of unknown parameters such as the mass, thrust law, and target speeds, is of particular importance when predicting the aircraft climb. Figure 1 illustrates the climb prediction problem, when using a physical model of the aircraft dynamics.

Some studies ([12], [13], [14]) detail the potential benefits that would be provided by additional or more accurate input data. In other works, the aircraft intent is formalized through the definition of an Aircraft Intent Description Language ([15], [16]) that could be used in air-ground data links to transmit some useful data to ground-based applications. All the necessary data required to predict aircraft trajectories might become available to ground systems someday. In the meantime, by applying Machine Learning techniques, we propose to learn
of trajectories. Then, using Machine Learning techniques on a profile. To do so, we build a set of examples by adjusting \cite{21}. In the current paper, we want to predict the speed than two mass estimation methods previously compared in predictive models. Using these techniques allows us to build the to improve trajectory prediction using Machine Learning
demonstrated that the model describes the aircraft performances and the data-driven models describe how the aircraft is operated i.e. the mass and the speed intent.

In current operation, the trajectory is predicted by using the reference mass and the reference \((\text{cas}_{\text{ref}}, M_{\text{ref}})\) values from the Eurocontrol Base of Aircraft Data (BADA) (see Figure 2). These values describe the speed profile of a climbing aircraft. The aircraft climbs at constant CAS (Calibrated Airspeed) equals to \text{cas} till the transition altitude is reached, then it climbs at a constant Mach \(M\). Although BADA associates one \((\text{cas}, M)\) value to each aircraft type, these values might be different among aircraft of the same type due to different cost-index for instance. In this paper, using all the information available, we want to predict a \((\text{cas}, M)\) value specific to the considered aircraft.

Figure 3 describes the approach developed in this paper to improve trajectory prediction using Machine Learning techniques. Using these techniques allows us to build the predictive models \(h_m, h_{\text{cas}}, \text{and } h_M\). In previous paper \cite{11}, we describe how the model \(h_m\) predicting the mass is obtained. We demonstrated that the model \(h_m\) was significantly more precise than two mass estimation methods previously compared in \cite{21}. In the current paper, we want to predict the speed profile. To do so, we build a set of examples by adjusting a \((\hat{\text{cas}}, \hat{M})\) speed profile to each trajectory of a recorded set of trajectories. Then, using Machine Learning techniques on this set of examples, we learn a model able to predict \((\text{cas}, M)\) values from the past points of a new input trajectory.

The rest of this paper is organized as follows: Section \(I\) presents some useful Machine Learning notions that help understanding the methodology applied in our work. Section \(II\) details the data used in this study. The application of Machine Learning techniques to our speed profile prediction problem is described in section \(III\) and the results are shown and discussed in section \(IV\) before the conclusion.

\section{Machine Learning}
This section describes some useful Machine Learning notions and techniques. For a more detailed and comprehensive description of these techniques, one can refer to \cite{22, 23}.

As explained in the previous section, we want to predict a variable \(y\), here the adjusted \(\hat{\text{cas}}\) and \(\hat{M}\) values of a given
trajectory, from a vector of explanatory variables $x$, which in our case is the data extracted from the past trajectory points and the weather forecast. This is typically a regression problem. Naively said, we want to learn a function $h$ such that $y = h(x)$ for all $(x, y)$ drawn from the distribution $(X, Y)$. Actually, such a function does not exist, in general. For instance, if two ordered pairs $(x, y_1)$ and $(x, y_2)$ can be drawn with $y_1 \neq y_2$, $h(x)$ cannot be equal to $y_1$ and $y_2$ at the same time. In this situation, it is hard to decide which value to give to $h(x)$.

A way to solve this issue is to use a real-valued loss function $L$. This function is defined by the user of function $h$. The value $L(h(x), y)$ models a cost for the specific use of $h$ when $(x, y)$ is drawn. With this definition, the user wants a function $h$ minimizing the expected loss $R(h)$ defined by equation (1).

$$R(h) = E_{(X,Y)}[L(h(X),Y)]$$

However, the main issue when choosing a function $h$ minimizing $R(h)$ is that we do not know the joint distribution $(X, Y)$. We only have a set of examples of this distribution.

### A. Learning from examples

Let us consider a set of $n$ examples $S = \{(x_i, y_i)\}_{i=1}^n$ coming from independent draws of the same joint distribution $(X, Y)$. We can define the empirical risk $R_{\text{empirical}}$ by the equation below:

$$R_{\text{empirical}}(h, S) = \frac{1}{|S|} \sum_{(x,y) \in S} L(h(x), y).$$

Assuming that the values $(L(h(x), y))_{(x,y) \in S}$ are independent draws from the same law with a finite mean and variance, we can apply the law of large numbers giving us that $R_{\text{empirical}}(h, S)$ converges to $R(h)$ as $|S|$ approaches $\infty$.

Thereby, the empirical risk is closely related to the expected risk. So, if we have to select $h$ among a set of functions $H$ minimizing $R(h)$, using a set of examples $S$, we select $h$ minimizing $R_{\text{empirical}}(h, S)$. This principle is called the principle of empirical risk minimization.

Unfortunately, choosing $h$ minimizing $R_{\text{empirical}}(h, S)$ will not always give us $h$ minimizing $R(h)$. Actually, it depends on the “size” of $H$ and the number of examples $|S|$ ([24], [25]). The smaller $H$ and the larger $|S|$ are, the more the principle of empirical risk minimization is relevant. When these conditions are not satisfied, the selected $h$ will probably have a high $R(h)$ despite a low $R_{\text{empirical}}(h, S)$. In this case, the function $h$ is overfitting the examples $S$.

These general considerations above have practical consequences on the use of Machine Learning. Let us denote $h_S$ the function in $H$ minimizing $R_{\text{empirical}}(\cdot, S)$. The expected risk using $h_S$ is given by $R(h_S)$. We use the principle of empirical risk minimization. As stated above, some conditions are required for this principle to be relevant. Concerning the size of the set of examples $S$: the larger, the better. Concerning the size of $H$, there is a tradeoff: the larger $H$ is, the smaller $\min_{h \in H} R(h)$ is. However, the larger $H$ is, the larger the gap between $R(h_S)$ and $\min_{h \in H} R(h)$ becomes. This is often referred to as the bias-variance tradeoff.

### B. Accuracy Estimation

In this subsection, we want to estimate the accuracy obtained using a Machine Learning algorithm $A$. Let us denote $A[S]$ the prediction model found by algorithm $A$ when minimizing $R_{\text{empirical}}(\cdot, S)$ considering a set of examples $S$.

The empirical risk $R_{\text{empirical}}(A[S], S)$ is not a suitable estimation of $R(A[S])$: the law of large numbers does not apply here because the predictor $A[S]$ is neither fixed nor independent from the set of examples $S$.

One way to handle this is to split the set of examples $S$ into two independent subsets: a training set $S_T$ and another set $S_V$ that is used to estimate the expected risk of $A[S_T]$, the model learned on the training set $S_T$. For that purpose, one can compute the holdout validation error $Err_{\text{val}}$ as defined by the equation below:

$$Err_{\text{val}}(A, S_T, S_V) = R_{\text{empirical}}(A[S_T], S_V).$$

Cross-validation is another popular method that can be used to estimate the expected risk obtained with a given learning algorithm. In a $k$-fold cross-validation method, the set of examples $S$ is partitioned into $k$ folds $(S_i)_{1 \leq i \leq k}$. Let us denote $S_{-i} = S \setminus S_i$. In this method, $k$ trainings are performed in order to obtain the $k$ predictors $A[S_{-i}]$. The mean of the holdout validation errors is computed, giving us the cross-validation estimation below:

$$CV(A, S) = \frac{1}{k} \sum_{i=1}^k |S_i| Err_{\text{val}}(A, S_{-i}, S_i).$$

This method is more computationally expensive than the holdout method but the cross-validation is more accurate than the holdout method ([26]). In our experiments, the folds were stratified. This technique is said to give more accurate estimates ([27]).

The accuracy estimation has basically two purposes: first, model selection in which we select the “best” model using accuracy measurements and second, model assessment in which we estimate the accuracy of the selected model. For model selection, the set $S_V$ in $Err_{\text{val}}(A, S_T, S_V)$ is called validation set whereas in model assessment this set is called testing set.

### C. Hyperparameter Tuning

Some learning algorithms have hyperparameters. These hyperparameters $\lambda$ are the parameters of the learning algorithm

---

1 The “size” of $H$ refers here to the complexity of the candidate models contained in $H$, and hence to their capability to adjust to complex data. As an example, if $H$ is a set of polynomial functions, we can define the “size” of $H$ as the highest degree of the functions contained in $H$. In classification problems, the “size” of $H$ can be formalized as the Vapnik-Chervonenkis dimension.

2 Actually, depending on the nature of the minimization problem and chosen algorithm, this predictor $A[S]$ might not be the global optimum for $R_{\text{empirical}}(\cdot, S)$, especially if the underlying optimization problem is handled by local optimization methods.
These parameters cannot be adjusted using the empirical risk because most of the hyperparameters are directly or indirectly related to the size of $H$. Thus, if the empirical risk was used, the selected hyperparameters would always be the ones associated to the largest $H$.

These hyperparameters allow us to control the size of $H$ in order to deal with the bias-variance tradeoff. These hyperparameters can be tuned using the holdout method on a validation set for accuracy estimation. In order to find $\lambda$ minimizing the accuracy estimation, we used a grid search which consists in an exhaustive search on a predefined set of hyperparameters. The Algorithm 1 is a learning algorithm without any hyperparameters. In this algorithm, 20% of the training set is held out as a validation set.

Algorithm 1: Hyperparameters tuning for an algorithm $A_\lambda$ and a set of examples $T$ (training set).

II. DATA USED IN THIS STUDY

A. Data Pre-processing

Recorded radar tracks from Paris Air Traffic Control Center are used in this study. This raw data is made of one position report every 1 to 3 seconds, over two months (July 2006, and January 2007). In addition, the wind and temperature data from Météo France are available at various isobar altitudes over the same two months.

The raw Mode-C altitude\(^3\) has a precision of 100 feet. Raw trajectories are smoothed using splines. Basic trajectory data is made of the following fields: aircraft position ($X,Y$ in a projection plane, or latitude and longitude in WGS84), ground velocity vector $V_g = (V_x, V_y)$, smoothed altitude ($H_p$, in feet above isobar 1013.25 hPa), rate of climb or descent $\frac{dH_p}{dt}$. The wind $W = (W_x, W_y)$ and temperature $T$ at each trajectory point are interpolated from the weather datagrid. The temperature differential $\Delta T$ is computed at each point of the trajectory.

Using the position, velocity and wind data, we compute the true air speed $V_a$. The successive velocity vectors allow us to compute the trajectory curvature at each point. The aircraft bank angle is then derived from true airspeed and the curvature of the air trajectory.

Along with these quantities derived from the Mode-C radar data and the weather data, we have access to some quantities in the flight plan like the Requested Flight Level for instance.

With the weather datagrid, we have also computed the temperature differential $\Delta T$ (weather grid) and the wind along $W_{\text{along}}$ (weather grid) at each altitude of the grid. This is done by using the $V_{a,XY}$, the time, the latitude and the longitude of the considered point. $W_{\text{along}}$ is the wind along the true air speed in the horizontal plane $V_{a,XY}$.

All the computed quantities are summarized in Table 1.

Table I: This table summarizes the quantities available in our study.

<table>
<thead>
<tr>
<th>quantities</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_p$</td>
<td>geopotential pressure altitude</td>
</tr>
<tr>
<td>$V_g$</td>
<td>Ground Speed</td>
</tr>
<tr>
<td>$V_a$</td>
<td>True Air Speed</td>
</tr>
<tr>
<td>$V_{a,XY}$</td>
<td>True Air Speed in the $(X,Y)$ plane</td>
</tr>
<tr>
<td>$d_{\text{air}}$</td>
<td>distance flown w.r.t. the air</td>
</tr>
<tr>
<td>$d_{\text{ground}}$</td>
<td>distance flown w.r.t. the ground</td>
</tr>
<tr>
<td>$\Delta T$</td>
<td>temperature differential (cf. [28])</td>
</tr>
<tr>
<td>$W$</td>
<td>wind</td>
</tr>
<tr>
<td>$W_{\text{along}}$</td>
<td>wind along $V_{a,XY}$</td>
</tr>
<tr>
<td>$W_{\text{across}}$</td>
<td>wind across $V_{a,XY}$</td>
</tr>
<tr>
<td>$W_z$</td>
<td>vertical wind</td>
</tr>
<tr>
<td>$\phi$</td>
<td>bank angle</td>
</tr>
<tr>
<td>$\text{CAS}$</td>
<td>Calibrated Air Speed</td>
</tr>
<tr>
<td>$Mach$</td>
<td>Mach number</td>
</tr>
<tr>
<td>$1/\rho_{\text{sol}}$</td>
<td>curvature w.r.t. the ground</td>
</tr>
<tr>
<td>$1/\rho_{\text{air}}$</td>
<td>curvature w.r.t. the air</td>
</tr>
<tr>
<td>$\hat{m}_{\text{LS}}$</td>
<td>estimated mass from past points using least square method [21]</td>
</tr>
<tr>
<td>$\epsilon_{\text{LS}}$</td>
<td>root mean square error obtained on the past points using the least square method</td>
</tr>
<tr>
<td>$\hat{m}_{\text{AD}}$</td>
<td>estimated mass from past points using adaptive method [29]</td>
</tr>
<tr>
<td>$\text{RFL}$</td>
<td>Requested Flight Level</td>
</tr>
<tr>
<td>$\text{Speed}$</td>
<td>requested speed</td>
</tr>
<tr>
<td>$\text{distance}$</td>
<td>distance between airports</td>
</tr>
<tr>
<td>$\text{AO}$</td>
<td>aircraft operator</td>
</tr>
<tr>
<td>$\text{DEP}$</td>
<td>departing airport</td>
</tr>
<tr>
<td>$\text{ARR}$</td>
<td>arrival airport</td>
</tr>
</tbody>
</table>

\(3\)This altitude is directly derived from the air pressure measured by the aircraft. It is the height in feet above isobar 1013.25 hPa.
containing 54 points, we can build 4 examples containing exactly 51 successive points. These 4 examples share the 48 points in the middle of the climb segment. Once these examples are built, we only keep the examples with the 11th point at an altitude superior to 15,000ft for the B744 aircraft type and 18,000ft for all the other aircraft types. Using this method, we have considered 9 aircraft types and we have built one set of examples for each aircraft type. Some of the chosen aircraft types are very different: the E145 is a short haul aircraft with a 285,700 kg reference mass while the B744 is a long haul aircraft with a 285,700 kg reference mass. Looking at Table II we see the size of the different sets.

<table>
<thead>
<tr>
<th>type</th>
<th>number of climbing segments</th>
<th>number of examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>A319</td>
<td>1863</td>
<td>15702</td>
</tr>
<tr>
<td>A320</td>
<td>5729</td>
<td>65514</td>
</tr>
<tr>
<td>A321</td>
<td>1866</td>
<td>21789</td>
</tr>
<tr>
<td>A332</td>
<td>1475</td>
<td>26269</td>
</tr>
<tr>
<td>B737</td>
<td>344</td>
<td>2178</td>
</tr>
<tr>
<td>B744</td>
<td>350</td>
<td>2750</td>
</tr>
<tr>
<td>B772</td>
<td>910</td>
<td>8525</td>
</tr>
<tr>
<td>E145</td>
<td>851</td>
<td>8310</td>
</tr>
<tr>
<td>F100</td>
<td>660</td>
<td>7430</td>
</tr>
</tbody>
</table>

Table II: Size of the different sets. Only the climbing segments generating at least one example in our final examples set are counted here.

D. Adjusting the Speed Profile to Observed Points

We want to learn the future speed profile. \( x \) is all the information we have at \( t_0 \) and before. The future speed intent \( y \). From a set of examples \((x, y)\), we learn a model \( h \) predicting the speed profile from \( x \). The predicted speed profile \( h(x) \) will be hopefully equal to the actual speed intent \( y \).

However, the actual speed intent \( y \) is not available in our data, thus we do not have the set of examples \((x, y)\), yet. We have to extract it from the observed trajectory. In BADA, the speed intent is characterized by two values, the \( \text{cas} \) and the \( M \). The aircraft climbs at a constant CAS equals to \( \text{cas} \) till the transition altitude \( H_{\text{trans}}(\text{cas}, M) \) is reached. Then, the aircraft climbs with a constant Mach \( M \). The parametrized speed is given by the equation below where \( f \) is a function given by [28]. \( T \) is the temperature, \( R \) and \( \kappa \) are physical constants.

\[
V_a(\text{cas}, M, H_p, T) = \begin{cases} f(\text{cas}, H_p, T) & \text{if } H_p \leq H_{\text{trans}}(\text{cas}, M) \\ M\sqrt{\kappa RT} & \text{otherwise} \end{cases}
\]

In this subsection, we describe how we have adjusted the parameters \((\text{cas}, M)\) in order to have the parametrized speed profile fit the observed speed. To do so, we find the parameter \((\hat{\text{cas}}, \hat{M})\) in \( \Omega = \mathbb{R}_+^* \times \mathbb{R}_+^* \) minimizing the function \( \Phi \) defined below.

\[
\Phi(\text{cas}, M) = \sum_{i=1}^{n} (V_a(\text{cas}, M, H_p, T_i) - V_{ai})^2
\]
\[
= \sum_{i/H_p \leq H_{\text{trans}}(\text{cas}, M)} (f(\text{cas}, H_p, T_i) - V_{ai})^2
\]
\[
+ \sum_{i/H_p, \text{trans}(\text{cas}, M) < H_p} (M\sqrt{\kappa RT} - V_{ai})^2
\]

As the aircraft is climbing, the \( H_p \) is an increasing sequence of altitude. In order to solve this, we split the domain in sub-domains specified below.

\[
\Omega = O_M \cup O_{\text{cas}} \cup \bigcup_{k=1}^{n-1} O_k \cup \bigcup_{k=1}^{n} F_k
\]

with \( O_M = H_{\text{trans}}^{-1}(\{0; H_p\}) \)

\( O_{\text{cas}} = H_{\text{trans}}^{-1}(\{H_{\text{trans}}(\text{cas}, +\infty)\}) \)

\( O_k = H_{\text{trans}}^{-1}(\{H_{pk}; H_{pk+1}\}) \)

\( F_k = H_{\text{trans}}^{-1}(H_{pk}) \)

The minimum \((\hat{\text{cas}}, \hat{M})\) on the domain \( \Omega \) is also a minimum of one sub-domain. Then \((\hat{\text{cas}}, \hat{M})\) is the minimum among the minimum of each sub-domain.

The minimum on \( F_k \) can be easily found because we have the constraint \( H_{\text{trans}} = H_{pk} \) allowing us to compute \( \hat{M} \) as a function of \( \text{cas} \). The other sub-domains are open sets, consequently the minimum on each sub-domain has to satisfy \( \nabla \Phi(\text{cas}, M) = 0 \).

Finally, if we want to learn the future speed intent, our \( y \) will be the \((\hat{\text{cas}}, \hat{M})\) adjusted on the 41 future points of one example.

III. Applying Machine Learning to Our Problem

We want to learn the speed intent \((\hat{\text{cas}}, \hat{M})\). To do so, we apply a Machine Learning method separately on \( \hat{\text{cas}} \) and then on \( \hat{M} \) giving us a model \( h_{\text{cas}} \) predicting \( \text{cas} \) and another model \( h_M \) predicting \( M \). The final predictive model \( h \) is \( h(x) = (h_{\text{cas}}(x), h_M(x)) \).

A. The Explanatory Variables

The explanatory variables \( x \) is a tuple composed of all the known variables at \( t_0 \) and before. We consider the derivatives \( \frac{dH_p}{dt}, \frac{d^2H_p}{dt^2}, \frac{dV_{\text{casu}}}{dt}, \frac{dV_g}{dt}, \frac{dV_c}{dt}, \frac{dV_E}{dt}, \frac{d\text{CAS}}{dt} \) and \( \frac{d\text{Mach}}{dt} \). The latter quantities and the quantities in Table I between \( H_p \) and \( V_E \) are computed on the 11 past points. Thus, these 27 quantities gives us 297 variables. However \( d_{\text{alt11}} \) and \( d_{\text{ground11}} \) are always equal to zero, then only 295 are included in \( x \). The other quantities in Table I are also included in \( x \).

\( \Delta T \) (weather grid) and \( W_{\text{along}} \) (weather grid) are computed on the last point of the past trajectory. These quantities are computed at the 10 different altitudes of the weather grid giving us 10 explanatory variables for each quantity. The 9 remaining quantities in Table I are not point-wise quantities, they are associated to the example giving us 9 variables to add
to $x$. The variables $AO$, $DEP$ and $ARR$ given by the flight plan are categorical variables whereas all the other variables in $x$ are numerical.

Finally, the tuple $x$ is composed of 324 explanatory variables.

### B. Gradient Tree Boosting

The stochastic gradient boosting tree algorithm was introduced in [30]. It applies functional gradient descent [31] using regression trees [32].

The functional gradient descent is a boosting technique. The model $h$ is iteratively improved. At each iteration we consider the gradient of the loss $g_i = \frac{\partial L(\hat{y}_i, y_i)}{\partial \hat{y}_i} (h(x_i), y_i)$. A Machine Learning algorithm is applied to a modified Machine Learning problem where the set of examples is $(x_i, g_i)_{1 \leq i \leq n}$. Then the model $g$ obtained is used to update the model $h$: the updated model is $h^{t+1}(x) = h(x) - \rho g(x)$, where $\rho$ is a constant minimizing the empirical risk. In the next iteration we consider $h^{t+1}$ instead of $h$.

In the Gradient Tree Boosting, the Machine Learning algorithm used in the functional gradient descent is a regression tree algorithm [32]. The model obtained by this algorithm is a binary tree representing a binary recursive partition of the input space. At each node, the input space is split in two according to a condition $x_j \leq s$. Though, the $J$ leaves describe a partition $(R_j)_{1 \leq j \leq J}$ of the input space. Each region $R_j$ is associated to a constant $\gamma_j$ and when $x$ falls into $R_j$, then $\gamma_j$ is predicted.

Regression trees have some advantages. This regression tree algorithm is insensitive to input monotonic transformations. Using $x_j$, log($x_j$) or exp($x_j$) leads to the same model. As a consequence, this algorithm is robust to outliers. It can easily handle categorical variables and missing values. However it is known to have a poor performance in prediction.

The latter drawback is very limited when used in combination with functional gradient descent as it is done in the gradient tree boosting algorithm. In our experiments we used the gbm package ([33]) in the R software. This algorithm optimizes the risk given by a quadratic loss $L(\hat{y}, y) = (\hat{y} - y)^2$. Let us note $\text{GBM}_{(m,J,\nu)}$ this algorithm, where $m$ is the number of boosting iterations, $J$ is the number of leaves of the tree and $\nu$ is the shrinkage parameter. The obtained model is a sum of regression trees. $J$ allows us to control the interaction between variables, as we have $J-1$ variables at most in each regression tree. $\nu$ is the learning rate. The hyperparameters grid used for this algorithm is presented in Table III.

<table>
<thead>
<tr>
<th>method</th>
<th>hyperparameter grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{GBM}_{(m,J,\nu)}$</td>
<td>$m = {1000, 1500, 2000}$, $J = {3, 5, 10, 15}$, $\nu = {0.001, 0.0025, 0.005, 0.01, 0.025, 0.05}$</td>
</tr>
</tbody>
</table>

Table III: Grid of hyperparameters used in our experiments.

### IV. RESULTS AND DISCUSSION

All the statistics presented in this section are computed using a stratified 10-fold cross-validation embedding the hyperparameter selection. Figure 4 illustrates how the data is partitioned, denoting $\lambda$ the hyperparameter vector. Our set of examples $S$ is partitioned in 10 folds $(S_i)_{1 \leq i \leq 10}$. The hyperparameters used to learn from $S_{-i}$ are selected using 20% of the fold $S_{-i}$ as a validation set. The model learned with these hyperparameters on $S_{-i}$ is then used to predict the mass on the test set $S_i$. Obviously, the intersection of the training set $S_{-i}$ and the test set $S_i$ is empty: they do not share any example trajectory. Overall, our set of predicted values (masses or altitudes) is the concatenation of the ten $\text{TuneGrid}(A_\lambda, \text{grid})[S_{-i}](S_i)$ (see algorithm 1). Therefore, all the statistics presented in this section are computed on test sets.

For the 9 aircraft types, each set of examples $S$ was split in 10 folds $(S_i)_{1 \leq i \leq 10}$ in a particular way. This split was made using the climbing segments. Thus, all the examples generated by one climbing segment are only in one fold $S_i$. It guarantees that the fold $S_i$ has examples independent from the ones in $S_{-i} = S \setminus S_i$.

![Figure 4: Cross-validation for model assessment, with an embedded holdout validation for hyperparameter tuning.](image)

#### A. Prediction of the speed profile ($\text{cas}, M$)

The results obtained with the Machine Learning algorithm are reported in Table IV. In this table we compare the speed profile $V_s(\text{cas}, M, H_{p_i}, T_i)$ to the observed speed $V_{as}$ on points $i$ with $t_i \geq 0$. We have tested different settings: "adj" denotes the adjusted values $\left(\hat{\text{cas}}, \hat{M}\right)$, "ref" denotes the baseline values given by BADA ($\text{cas}_{\text{ref}}, M_{\text{ref}}$), "mean" denotes the values given by the mean of the adjusted values $\left(\bar{\text{cas}}, \bar{M}\right)$ and "GBM" denotes the predicted values $\left(\text{cas}_{\text{GBM}}, M_{\text{GBM}}\right)$. The "adj" setting result of the adjustment described in II-D. It is the lowest RMSE we can get using a CAS/Mach speed profile.

According to this table, using the baseline, the RMSE is around 20 kts for all the aircraft types except for the B744, the E145 and the F100. For these two aircraft types, the RMSE is reduced by at least 41% with the "mean" method. With this method, the RMSE reduction is less spectacular for the six other aircraft types with nearly no reduction for some aircraft types. This indicates that the reference speed parameters are in accordance with mean parameters of the observed trajectories for these aircraft types. However, even for these aircraft types, the prediction can still be improved by predicting a ($\text{cas}, M$) value specific to the considered aircraft. This is what is done when a GBM model is used. When compared to the "mean" method, the GBM method reduces the RMSE by at least 21%.

#### B. Computing the Predicted Trajectory Using Machine Learning and BADA

In order to actually predict a trajectory using the BADA model and assuming a max climb thrust, one still has to specify
a mass and a speed profile. Both are usually unknown from ground systems. In our experiment, we want to evaluate the impact of the predicted mass and the predicted speed profile on the trajectory prediction. The prediction of the mass was introduced in a previous paper [1] using a similar approach: a mass \( \hat{m} \) is adjusted on future points and a model predicting this adjusted mass \( \hat{m} \) from known variables is learned using GBM. This approach has been demonstrated more accurate than mass estimation methods introduced in [34], [21]. In the current paper, we adjust a CAS/Mach speed profile, and we learn two GBM models predicting the adjusted values \( \hat{cas} \) and \( \hat{M} \). The overall process to predict the future trajectory is described in Figure 5. The mass and the speed profile are specified using the three predictive models \( h_m \), \( \hat{h}_{cas} \) and \( h_M \).

C. Prediction of the Future Altitude

The results obtained with our methods are described in Table IV. The “ref” parameter is the BADA reference parameter. The “GBM” parameter for the mass is obtained by using the model \( h_m \). The line with “GBM” for the mass and “adj” for the speed cannot be used in an operational context as it uses adjusted values. This line is only used for comparison purpose. All the other lines can be used in an operational context. The baseline method, referred as \( BADA_{ref} \), is the line with “ref” for the mass and the speed. Our approach, referred as \( BADA_{GBM} \), is the line with “GBM” for the mass and the speed. These two setups can be used in an operational context: they use only the information available at the time the prediction is computed.

When compared with the baseline \( BADA_{ref} \), the use of the predicted mass and the reference speed profile reduce the RMSE on the altitude by at least 29% for any aircraft type except the E145. Note that for this latter, the gap between the reference speed profile and the observed speed profile is fairly high with a RMSE of 76.1 kts while it is around 20 kts for the other aircraft types (see Table IV).

If we consider the “mean” speed, the RMSE on the altitude is noticeably reduced only for the E145 and the F100. This was expected because these two aircraft types have the largest RMSE on the speed when using the “ref” parameter.

Using the predicted speed profile, we consider the \( BADA_{GBM} \). If we compare this latter to the \( BADA_{ref} \) setup, the RMSE on the altitude is reduced by at least 45% for all the aircraft types, including the E145. This reduction reaches 87% for the B772.
Figure 6: This figure portrays the error on the altitude obtained by $BADA_{\text{ref}}$ and $BADA_{\text{GBM}}$ for nine different aircraft types. Each time step, a boxplot shows the 5 %, 25 %, 50 %, 75 % and 95 % quantiles. The whiskers (resp. box) of the boxplot contains 90 % (resp. 50 %) of the data.
Table V: These statistics, in feet, are computed on the differences between the predicted and observed altitude, $(H_{p}^{\text{pred}} - H_{p}^{\text{obs}})$, at the time $t = 600$ s. Figure 6 portrays the error on the altitude at different time steps. The whiskers (resp. box) of the boxplot contains 90 % (resp. 50 %) of the data.

**REFERENCES**


