Machine Learning Applied to Airspeed Prediction During Climb

R. Alligier, D. Gianazza, N. Durand
ENAC, MAIAA, F-31055 Toulouse, France
Univ. de Toulouse, IRIT/APO, F-31400 Toulouse, France

***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 $(\widehat{cas}, \widehat{M})$ in this database, we learn a model able to predict the (cas, 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 developped 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

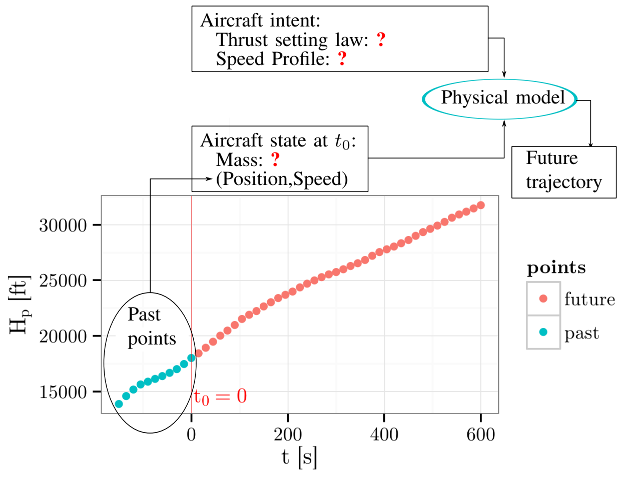

Figure 1: The ground-based aircraft climb prediction problem.

some of the unknown parameters of the point-mass model from the data that is already available today, typically from the observed radar tracks of past and current flights.

Applying Machine Learning techniques on the trajectory prediction problem is not a new idea. A decade ago, [17] has applied artificial neural network on this problem. It has also been investigated more recently using different Machine Learning techniques ([18], [19], [20]). With these approaches, the obtained model directly predicts the trajectory. It is a black-box hiding what comes from the aircraft performances and what comes from the airline procedures, *i.e.* the way it is operated. In this context, the originality of our work is that we keep the physical model in the loop. The physical 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 $(cas_{\mathrm{ref}}, M_{\mathrm{ref}})$ 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 cas till the transition altitude is reached, then it climbs at a constant Mach M. Although BADA associates one (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 (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_{cas} and h_{M}. In previous paper [1], 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 [21]. In the current paper, we want to predict the speed profile. To do so, we build a set of examples by adjusting a $\left(\widehat{cas}, \widehat{M}\right)$ 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 (cas, M) values from the past points of a new input trajectory.

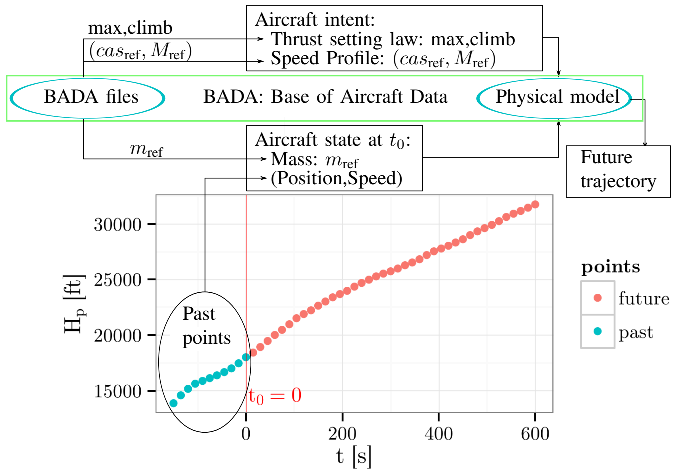

Figure 2: Baseline method : the BADA prediction of the future aircraft climb

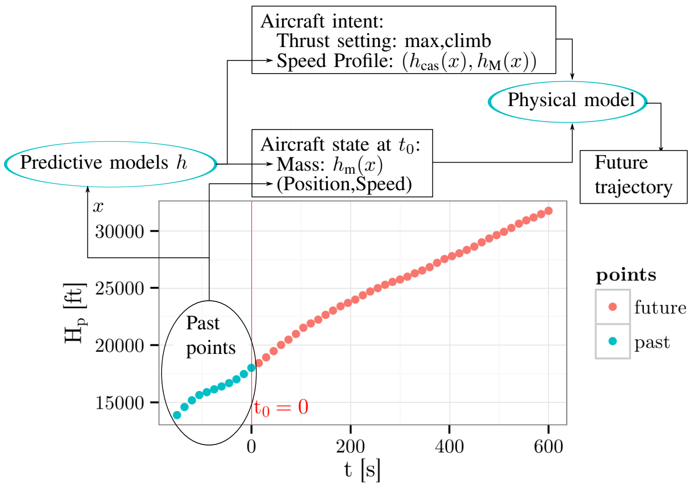

Figure 3: This figure describes how the future trajectory is predicted by applying Machine Learning techniques. When a fresh trajectory is observed, the input variables x are computed from the observed points and the predictive models h_{m}, h_{cas} and h_{M} are used to predict the mass and the speed profile.

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.

I. 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 [22], [23].

As explained in the previous section, we want to predict a variable y, here the adjusted $\widehat{cas}$ and $\widehat{M}$ 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). The value $R(h)$ is also called the *expected risk*.

$$R(h) = E_{(X,Y)} \left[L\left(h(X), Y\right) \right] \tag{1}$$

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

$$R_{\text{empirical}}(h, S) = \frac{1}{|S|} \sum_{(x,y) \in S} L\left(h(x), y\right). \tag{2}$$

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"[1] 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 $R_{\text{empirical}}(., S)$. 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 $\underset{h \in H}{min}\ R(h)$ is. However, the larger H is, the larger the gap between $R(h_S)$ and $\underset{h \in H}{min}\ R(h)$ 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 $\mathcal{A}$. Let us denote $\mathcal{A}[S]$ the prediction model found by algorithm $\mathcal{A}$ when minimizing $R_{\text{empirical}}(., S)$[2], considering a set of examples S.

The *empirical risk* $R_{\text{empirical}}(\mathcal{A}[S], S)$ is not a suitable estimation of $R(\mathcal{A}[S])$: the law of large numbers does not apply here because the predictor $\mathcal{A}[S]$ 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 $\mathcal{A}[S_T]$, the model learned on the training set S_T. For that purpose, one can compute the holdout validation error Err_{val} as defined by the equation below:

$$Err_{\text{val}}(\mathcal{A}, S_T, S_V) = R_{\text{empirical}}(\mathcal{A}[S_T], S_V). \tag{3}$$

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 $(S_i)_{1 \leqslant i \leqslant k}$. Let us denote $S_{-i} = S \backslash S_i$. In this method, k trainings are performed in order to obtain the k predictors $\mathcal{A}[S_{-i}]$. The mean of the holdout validation errors is computed, giving us the cross-validation estimation below:

$$CV(\mathcal{A}, S) = \sum_{i=1}^{k} \frac{|S_i|}{|S|} Err_{\text{val}}(\mathcal{A}, S_{-i}, S_i). \tag{4}$$

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 $Err_{\text{val}}(\mathcal{A}, S_T, S_V)$ is called *validation set* whereas in model assessment this set is called *testing set*.

C. Hyperparameter Tuning

Some learning algorithms have hyperparameters. These hyperparameters λ 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 $\mathcal{A}[S]$ might not be the global optimum for $R_{\text{empirical}}(., S)$, especially if the underlying optimization problem is handled by local optimization methods.

$\mathcal{A}_\lambda$. 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 λ 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*.

```
function TuneGrid($\mathcal{A}_\lambda$, $grid$)[$T$]
    $(T_T, T_V) \leftarrow split_{(80\%,20\%)}(T)$
    $\lambda^* \leftarrow \underset{\lambda \in grid}{argmin}\ Err_{\text{val}}(\mathcal{A}_\lambda, T_T, T_V)$
    return $\mathcal{A}_{\lambda^*}[T]$
end function
```

Algorithm 1: Hyperparameters tuning for an algorithm $\mathcal{A}_\lambda$ 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 1,013.25 hPa), rate of climb or descent $\frac{dH_p}{dt}$. The wind $W = (W_x, W_y)$ and temperature T at every trajectory point are interpolated from the weather datagrid. The temperature differential ΔT 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 ΔT(weather grid) and the wind along W_{along}(weather grid) at each altitude of the grid. This is done by using the V_{aXY}, the time, the latitude and the longitude of the considered point. W_{along} is the wind along the true air speed in the horizontal plane V_{aXY}.

All the computed quantities are summarized in Table I.

quantities	description
H_p	geopotential pressure altitude
V_g	Ground Speed
V_a	True Air Speed
V_{aXY}	True Air Speed in the (X,Y) plane
d_{air}	distance flown w.r.t. the air
d_{ground}	distance flown w.r.t. the ground
ΔT	temperature differential (cf. [28])
W	wind
W_{along}	wind along V_{aXY}
W_{across}	wind across V_{aXY}
W_Z	vertical wind
θ_c	drift angle
CAS	Calibrated Air Speed
Mach	Mach number
$1/r_{\text{sol}}$	curvature w.r.t. the ground
$1/r_{\text{air}}$	curvature w.r.t. the air
ϕ	bank angle
$e = V_a \frac{dV_a}{dt} + g_0 \frac{T}{T-\Delta T} \frac{dH_p}{dt}$	specific energy rate
$ew = e + \overrightarrow{W}.\overrightarrow{V_a}$	specific energy rate corrected from the wind effect
ΔT (*weather grid*)	temperature differential on a grid of different H_p
W_{along} (*weather grid*)	wind along V_{aXY} on a grid of different H_p
$\hat{m}_{LS}$	estimated mass from past points using least square method [21]
e_{LS}	root mean square error obtained on the past points using the least square method
$\hat{m}_{AD}$	estimated mass from past points using adaptive method [29]
RFL	Requested Flight Level
Speed	requested speed
distance	distance between airports
AO	aircraft operator
DEP	departing airport
ARR	arrival airport

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

B. Filtering Climb Segments

Our dataset includes all flights departing from Paris-Orly (LFPO) or Paris-Charles de Gaulle Airport (LFPG). Needless to say, this approach can be replicated to other airports.

The trajectories are filtered so as to keep only the climb segments. An additional 80 seconds is clipped from the beginning and end of each segment so as to remove climb/cruise or cruise/climb transitions. It is worth noticing that the trajectories have not been filtered on the speed profile but only on the rate of climb and the altitude.

C. Building the Sets of Examples

The climb segments are sampled every 15 seconds. From these sampled segments, we build examples containing exactly 51 points. In these examples, the first 11 points (past trajectory) are used to predict the mass and the speed profile. The remaining points (future trajectory) are used to compute the error between the predicted and actual trajectory.

From one sampled climb segment we build as many examples as we can. For instance, from a sampled climb segment

[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 11^{th} 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 18,500 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.

type	number of climbing segments	number of examples
A319	1863	15702
A320	5729	65514
A321	1866	21789
A332	1475	28629
B737	344	2178
B744	350	2750
B772	910	8525
E145	851	8310
F100	660	7430

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 intentis 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 cas and the M. The aircraft climbs at a constant CAS equals to cas till the transition altitude $H_{p_{\text{trans}}}(cas, M)$ 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 κ are physical constants.

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

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

$$\begin{aligned} \Phi(cas, M) =& \sum_{i=1}^{n} \left(V_a(cas, M, H_{p_i}, T_i) - V_{ai}\right)^2 \\ =& \sum_{i/H_{p_i} \leqslant H_{p,trans}(cas,M)} \left(f(cas, H_{p_i}, T_i) - V_{ai}\right)^2 \\ &+ \sum_{i/H_{p,trans}(cas,M) < H_{p_i}} \left(M\sqrt{\kappa RT_i} - V_{ai}\right)^2 \end{aligned}$$

As the aircraft is climbing, the H_{p_i} is an increasing sequence of altitude. In order to solve this, we split the domain in sub-domains specified below.

$$\begin{aligned} \Omega =& O_M \cup O_{cas} \cup \bigcup_{k=1}^{n-1} O_k \cup \bigcup_{k=1}^{n} F_k \\ \text{with } & O_M = {H_{p_{\text{trans}}}}^{-1}\left(]0; H_{p1}[\right) \\ & O_{cas} = {H_{p_{\text{trans}}}}^{-1}\left(]H_{p_n}; +\infty[\right) \\ & O_k = {H_{p_{\text{trans}}}}^{-1}\left(\left]H_{p_k}; H_{p_{k+1}}\right[\right) \\ & F_k = {H_{p_{\text{trans}}}}^{-1}\left(H_{p_k}\right) \end{aligned}$$

The minimum $\left(\widehat{cas}, \widehat{M}\right)$ on the domain Ω is also a minimum of one sub-domain. Then $\left(\widehat{cas}, \widehat{M}\right)$ is the minimum among the minimum of each sub-domain.

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

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

III. Applying Machine Learning to our Problem

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

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 $\frac{dH_p}{dt}$, $\frac{d^2H_p}{dt^2}$, $\frac{dW_{\text{along}}}{dt}$, $\frac{dV_g}{dt}$, $\frac{dV_a}{dt}$, $\frac{dW_Z}{dt}$, $\frac{dCAS}{dt}$ and $\frac{dMach}{dt}$. The latter quantities and the quantities in Table I between H_p and *ew* are computed on the 11 past points. Thus, these 27 quantities gives us 297 variables. However $d_{\text{air}11}$ and $d_{\text{ground}11}$ are always equal to zero, then only 295 are included in x. The other quantities in Table I are also included in x. ΔT (*weather grid*) and W_{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 $g_i = \frac{\partial L(\hat{y}, y_i)}{\partial \hat{y}} \left(h\left(x_i\right), y_i\right)$. A Machine Learning algorithm is applied to a modified Machine Learning problem where the set of examples is $(x_i, g_i)_{1 \leqslant i \leqslant n}$. Then the model g obtained is used to update the model h: the updated model is $h^{+1}(x) = h(x) - \rho g(x)$, where ρ is a constant minimizing the empirical risk. In the next iteration we consider h^{+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 $x_j \leqslant s$. Though, the J leaves describe a partition $(R_j)_{1 \leqslant j \leqslant J}$ of the input space. Each region R_j is associated to a constant γ_j and when x falls into R_j, then γ_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 $\exp\left(x_j\right)$ 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 GBM$_{(m,J,\nu)}$ this algorithm, where m is the number of boosting iterations, J is the number of leaves of the tree and ν 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. ν is the learning rate. The hyperparameters grid used for this algorithm is presented in Table III.

method	hyperparameter grid
GBM$_{(m,J,\nu)}$	$m = \{1000, 1500, 2000\}$ $J = \{3, 5, 10, 15\}$ $\nu = \{0.001, 0.0025, 0.005, 0.01, 0.025, 0.05\}$

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 partitionned, denoting λ the hyperparameter vector. Our set of examples S is partitioned in 10 folds $(S_i)_{1 \leqslant i \leqslant 10}$. 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 $TuneGrid(\mathcal{A}_\lambda, grid)[S_{-i}]\,(S_i)$ (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 $(S_i)_{1 \leqslant i \leqslant 10}$ 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 have examples independent from the ones in $S_{-i} = S \backslash S_i$.

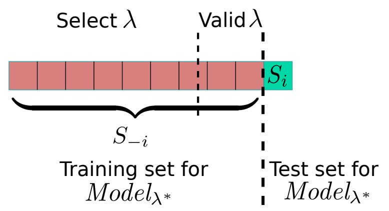
Select λ
Valid λ
S_i
S_{-i}
Training set for $Model_{\lambda^*}$
Test set for $Model_{\lambda^*}$

Figure 4: Cross-validation for model assessment, with an embedded holdout validation for hyperparameter tuning.

A. Prediction of the speed profile (cas, M)

The results obtained with the Machine Learning algorithm are reported in Table IV. In this table we compare the speed profile $V_a\left(cas, M, H_{p_i}, T_i\right)$ to the observed speed $V_{a\,i}$ on points i with $t_i \geqslant 0$. We have tested different settings: "adj" denotes the adjusted values $\left(\widehat{cas}, \widehat{M}\right)$, "ref" denotes the baseline values given by BADA $(cas_{\text{ref}}, M_{\text{ref}})$, "mean" denotes the values given by the mean of the adjusted values $\left(\overline{\widehat{cas}}, \overline{\widehat{M}}\right)$ and "GBM" denotes the predicted values $(cas_{\text{GBM}}, M_{\text{GBM}})$. 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 (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

type	speed	mean	stdev	mean abs	RMSE	max abs
A319	ref	3.77	20.7	15.8	21	123
A319	mean	1.19	20.4	14.7	20.5	119
A319	GBM	0.412	12.5	8.2	12.5	103
A319	adj	0.0259	7.85	5	7.85	102
A320	ref	2.34	21	15.3	21.2	134
A320	mean	1.1	21.2	14.9	21.2	129
A320	GBM	0.57	12.6	8	12.6	114
A320	adj	0.0262	7.71	4.68	7.71	124
A321	ref	3.16	22.9	17.7	23.1	115
A321	mean	1.09	23.1	17.6	23.1	112
A321	GBM	0.546	13.5	9.02	13.5	117
A321	adj	0.0285	7.83	4.87	7.83	94.6
A332	ref	-8.71	17.6	16.4	19.6	115
A332	mean	0.118	16.4	11.7	16.4	115
A332	GBM	0.648	11.6	7.4	11.6	103
A332	adj	0.0233	7.03	4.37	7.03	81.7
B737	ref	8.56	17.5	13.7	19.4	110
B737	mean	0.739	16.5	12.1	16.5	109
B737	GBM	0.391	12.4	8.17	12.4	112
B737	adj	-0.00503	6.82	4.54	6.82	93.4
B772	ref	-14.4	16.7	19.2	22.1	83.5
B772	mean	0.298	14	9.97	14	91.1
B772	GBM	0.66	11	7.18	11	83.6
B772	adj	0.0429	7.17	4.33	7.17	78.2
B744	ref	-28.7	21.2	32.2	35.6	76.2
B744	mean	1.01	20.8	16.1	20.8	90.7
B744	GBM	0.705	14.7	9.94	14.7	71
B744	adj	0.121	10.9	6.32	10.9	71.1
E145	ref	69.2	31.7	69.2	76.1	166
E145	mean	1.82	29	24.1	29	93.3
E145	GBM	1.12	16.2	12.2	16.2	81.2
E145	adj	-0.0446	8.26	5.85	8.26	65.3
F100	ref	36.8	19.5	36.9	41.6	168
F100	mean	0.646	19.4	14	19.4	132
F100	GBM	0.654	12.7	8.44	12.7	91.1
F100	adj	0.0178	5.78	3.64	5.78	71

Table IV: These statistics, in knots, are computed on the differences between the predicted speed and the observed speed $V_a\left(cas, M, H_{p_i}, T_i\right) - V_{a_i}$ for i such as $t_i \geqslant 0$ (*i.e.* $i \geqslant 11$).

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 $\widehat{m}$ is adjusted on future points and a model predicting this adjusted mass $\widehat{m}$ 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 $\widehat{cas}$ and $\widehat{M}$. 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}, h_{cas} and h_{M}.

C. Prediction of the Future Altitude

The results obtained with our methods are described in Table V. 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

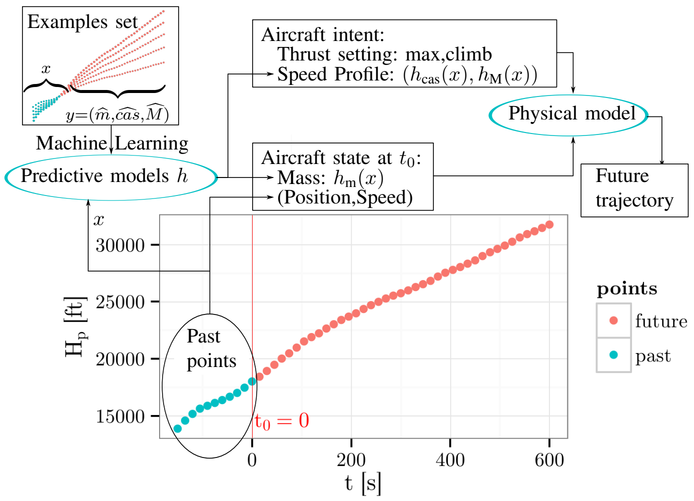

Figure 5: This figure describes how the future trajectory is predicted by applying Machine Learning techniques. The examples set is used to build predictive models $(h_{\text{m}}, h_{\text{cas}}, h_{\text{M}})$. When a fresh trajectory is observed, the input variables x are computed from the observed points and the predictive models are used to predict the mass and the speed profile.

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 $BADA_{\text{ref}}$, is the line with "ref" for the mass and the speed. Our approach, referred as $BADA_{\text{GBM}}$, 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 $BADA_{\text{ref}}$, 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 $BADA_{\text{GBM}}$. If we compare this latter to the $BADA_{\text{ref}}$ 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.

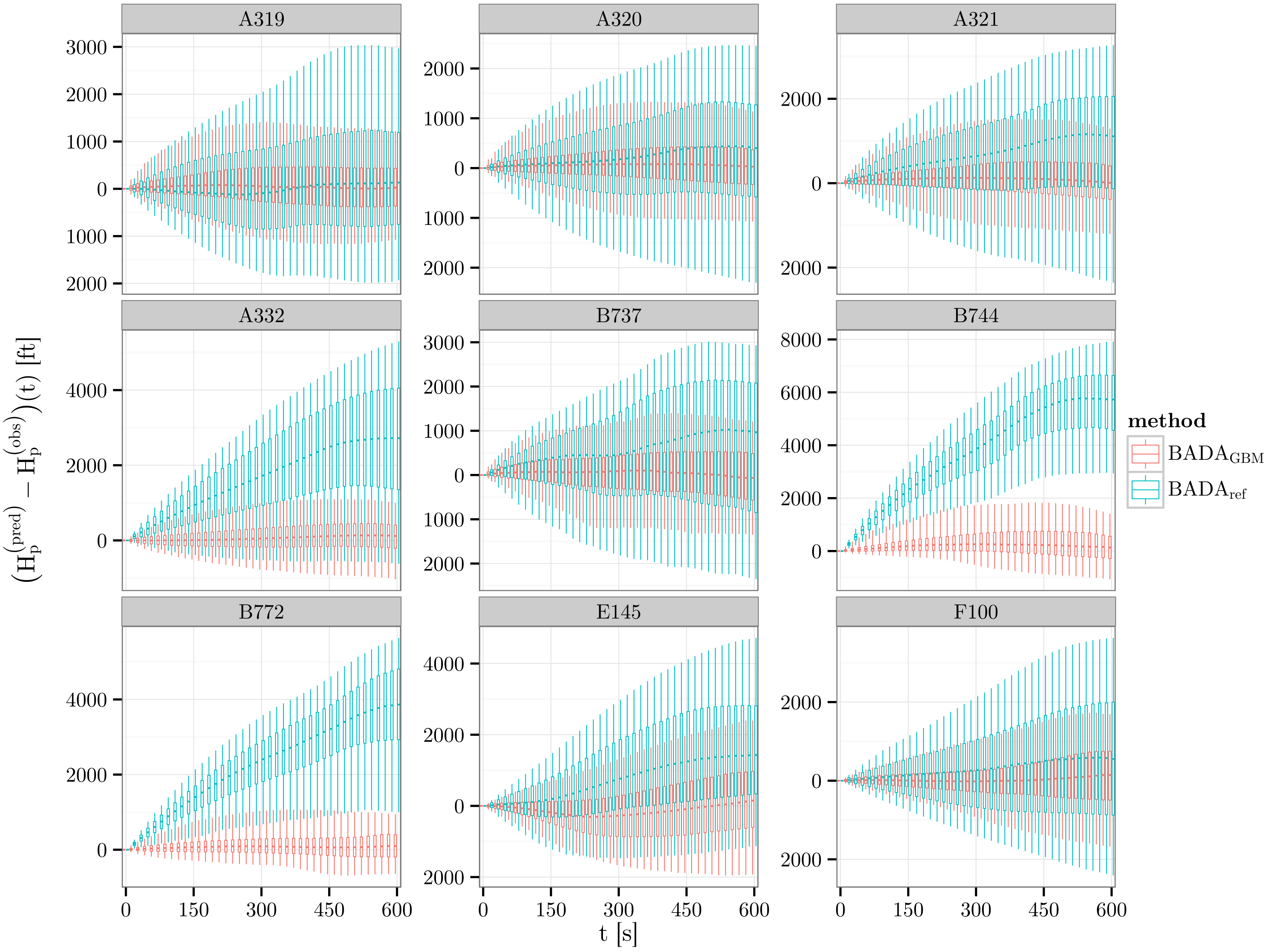

Figure 6: This figure portrays the error on the altitude obtained by $BADA_{\text{ref}}$ and $BADA_{\text{GBM}}$ for nine different aircraft types. Each time step, a boxplot shows the 5 %, 25 %, 50 %, 75 % and 95 % quantiles. The whiskers (resp. box) of the boxplot contains 90 % (resp. 50 %) of the data.

However, the reduction obtained on the altitude by using the predicted speed profile is large only for the E145 and the F100. The impact of the speed error reduction is hidden by other sources of error. Firstly, the weather model and the BADA model are not perfect. Secondly, we have assumed a max climb thrust setting which might not be a relevant assumption for all the climb trajectories. Thirdly, the mass used is a predicted mass. Even if the $\left(\widehat{cas}, \widehat{M}\right)$ values were perfectly predicted, the trajectory prediction will not be perfect. The error made in this perfect case can be read at the line with "GBM" for the mass and "adj" for the speed. Thus, even with an RMSE around 8 kts on the speed profile, the RMSE on the altitude is reduced but not greatly reduced.

type	mass	speed	mean	stdev	mean abs	rmse	max abs
A319	ref	ref	274	1472	1176	1497	5315
A319	GBM	ref	237	772	605	808	5350
A319	GBM	mean	47.1	767	575	769	5478
A319	GBM	GBM	42.1	725	532	726	5529
A319	GBM	adj	19.6	607	452	608	5720
A320	ref	ref	290	1420	1165	1449	5753
A320	GBM	ref	187	715	553	739	6815
A320	GBM	mean	45.3	718	534	719	6707
A320	GBM	GBM	23.5	681	490	681	7193
A320	GBM	adj	-0.895	523	389	523	6202
A321	ref	ref	863	1683	1588	1891	6154
A321	GBM	ref	33.1	783	571	784	4627
A321	GBM	mean	37	782	570	783	4642
A321	GBM	GBM	22.1	774	554	774	4418
A321	GBM	adj	-15.8	584	421	584	5569
A332	ref	ref	2622	1820	2783	3192	6769
A332	GBM	ref	-107	673	479	682	5217
A332	GBM	mean	66	664	469	667	4997
A332	GBM	GBM	70.4	651	460	654	4934
A332	GBM	adj	40.7	572	393	574	4795
B737	ref	ref	606	1750	1619	1852	4157
B737	GBM	ref	-40.8	796	616	797	3672
B737	GBM	mean	-51.7	796	617	797	3667
B737	GBM	GBM	-52	804	629	805	3645
B737	GBM	adj	-86.6	814	616	818	4216
B744	ref	ref	5558	1646	5580	5797	10183
B744	GBM	ref	12.4	844	649	844	3495
B744	GBM	mean	100	842	646	848	3372
B744	GBM	GBM	142	778	586	790	3342
B744	GBM	adj	103	748	547	755	3656
B772	ref	ref	3728	1413	3750	3987	7145
B772	GBM	ref	-80.2	534	425	540	3513
B772	GBM	mean	99.5	502	379	512	3647
B772	GBM	GBM	112	500	385	512	3446
B772	GBM	adj	65.7	453	334	458	3316
E145	ref	ref	1623	1801	1909	2425	7428
E145	GBM	ref	1667	2064	2032	2653	8280
E145	GBM	mean	548	2115	1741	2185	7289
E145	GBM	GBM	190	1314	1010	1327	5378
E145	GBM	adj	68.7	750	562	753	5858
F100	ref	ref	556	1879	1616	1959	6539
F100	GBM	ref	642	1229	1166	1387	4940
F100	GBM	mean	193	1209	993	1225	5425
F100	GBM	GBM	102	1022	793	1027	4490
F100	GBM	adj	43.3	732	543	734	4587

Table V: These statistics, in feet, are computed on the differences between the predicted altitude and the observed altitude $\left(H_p^{(pred)} - H_p^{(obs)}\right)$ at the time $t = 600$ s.

Figure 6 portrays the error on the altitude at different time horizons. More specifically, boxplots are presented at each time step. The whiskers (resp. box) of the boxplot contains 90 % (resp. 50 %) of the data.

CONCLUSION

To conclude, let us summarize our approach and findings, before giving a few perspectives on future works. In this article we have described a way to predict the future speed profile. Using Machine Learning and a set of examples, we have built models predicting the values (cas, M) of a CAS/Mach speed profile. Using real Mode-C radar, this approach has been tested on the 9 different aircraft types. In order to evaluate the accuracy of the Machine Learning method, a cross-validation is used. When compared to the reference speed profiles provided by BADA, the RMSE on the speed is reduced by at least by 36 % using GBM, a Machine Learning method. Concerning the E145, this RMSE is reduced by 79 %.

In order to predict the future trajectory, this approach is used in conjunction with a similar approach described in [1]. This latter is used to predict the mass. Then using the predicted mass and speed profile, the BADA physical model is used to compute the predicted future trajectory with a 10 minutes horizon. The RMSE on the future altitude is reduced by at least 45 %. This reduction reaches 87 % for the B772.

From an operational point of view, the resulting improvement in the climb prediction accuracy would certainly benefit air traffic controllers, especially in the vertical separation task as shown in [34]. Furthermore, even if it was not computed in our study, the proposed method probably reduces the along track error and the Top Of Climb prediction error.

We only have considered a CAS/Mach speed profile. However, as said before, the climbing trajectories in our data does not always follow a CAS/Mach speed profile. In order to improve the trajectory prediction, we might consider other climb procedures. For future work, we might consider a procedure in which the aircraft climbs at a constant CAS cas_1 till the altitude $H_{p_{cas}}$, then accelerates/decelerates till the CAS reaches cas_2 and finally follows a (cas_2, M) CAS/Mach speed profile. However, in order to use this, we would have to adapt our method to predict $\left(cas_1, H_{p_{cas}}, cas_2, M\right)$.

REFERENCES

[1] R. Alligier, D. Gianazza, and N. Durand. Machine learning and mass estimation methods for ground-based aircraft climb prediction. *Intelligent Transportation Systems, IEEE Transactions on*, submitted.

[2] SESAR Consortium. Milestone Deliverable D3: The ATM Target Concept. Technical report, 2007.

[3] H. Swenson, R. Barhydt, and M. Landis. Next Generation Air Transportation System (NGATS) Air Traffic Management (ATM)-Airspace Project. Technical report, National Aeronautics and Space Administration, 2006.

[4] X. Prats, V. Puig, J. Quevedo, and F. Nejjari. Multi-objective optimisation for aircraft departure trajectories minimising noise annoyance. *Transportation Research Part C*, 18(6):975–989, 2010.

[5] G. Chaloulos, E. Crück, and J. Lygeros. A simulation based study of subliminal control for air traffic management. *Transportation Research Part C*, 18(6):963–974, 2010.

[6] James K Kuchar and Lee C Yang. A review of conflict detection and resolution modeling methods. *Intelligent Transportation Systems, IEEE Transactions on*, 1(4):179–189, 2000.

[7] Lucia Pallottino, Eric M Feron, and Antonio Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *Intelligent Transportation Systems, IEEE Transactions on*, 3(1):3–11, 2002.

[8] J. M. Alliot, Hervé Gruber, and Marc Schoenauer. Genetic algorithms for solving ATC conflicts. In *Proceedings of the Ninth Conference on Artificial Intelligence Application*. IEEE, 1992.

[9] N. Durand, J.M. Alliot, and J. Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.

[10] Nicolas Durand and Jean-Marc Alliot. Ant colony optimization for air traffic conflict resolution. In *8th USA/Europe Air Traffic Management Research and Developpment Seminar*, 2009.

[11] C. Vanaret, D. Gianazza, N. Durand, and J.B. Gotteland. Benchmarking conflict resolution algorithms. In *International Conference on Research in Air Transportation (ICRAT), Berkeley, California, 22/05/12-25/05/12*, page (on line), http://www.icrat.org, may 2012. ICRAT.

[12] Study of the acquisition of data from aircraft operators to aid trajectory prediction calculation. Technical report, EUROCONTROL Experimental Center, 1998.

[13] ADAPT2. aircraft data aiming at predicting the trajectory. data analysis report. Technical report, EUROCONTROL Experimental Center, 2009.

[14] R. A. Coppenbarger. Climb trajectory prediction enhancement using airline flight-planning information. In *AIAA Guidance, Navigation, and Control Conference*, 1999.

[15] J. Lopez-Leones, M.A. Vilaplana, E. Gallo, F.A. Navarro, and C. Querejeta. The aircraft intent description language: A key enabler for air-ground synchronization in trajectory-based operations. In *Proceedings of the 26th IEEE/AIAA Digital Avionics Systems Conference*. DASC, 2007.

[16] J. Lopes-Leonés. *The Aircraft Intent Description Language*. PhD thesis, University of Glasgow, 2007.

[17] Y. Le Fablec. *Prévision de trajectoires d'avions par réseaux de neurones*. PhD thesis, Thèse doctorat informatique de l'INPT, 1999.

[18] K. Tastambekov, S. Puechmorel, D. Delahaye, and C. Rabut. Aircraft trajectory forecasting using local functional regression in sobolev space. *Transportation Research Part C: Emerging Technologies*, 39(0):1 – 22, 2014.

[19] M. Ghasemi Hamed. *Méthodes non-paramétriques pour la prévision d'intervalles avec haut niveau de confiance: application à la prévision de trajectoires d'avions*. PhD thesis, Thèse doctorat informatique de l'INPT, 2014.

[20] Marko Hrastovec and Franc Solina. Machine learning model for aircraft performances. In *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd*, pages 8C4–1. IEEE, 2014.

[21] R. Alligier, D. Gianazza, and N. Durand. Ground-based estimation of aircraft mass, adaptive vs. least squares method. In *10th USA/Europe Air Traffic Management Research and Developpment Seminar*, 2013.

[22] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[23] C. M Bishop. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[24] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk minimization. *Pattern Recogn. Image Anal.*, 1(3):284–305, 1991.

[25] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[26] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208. ACM, 1999.

[27] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143. Morgan Kaufmann, 1995.

[28] A. Nuic. User manual for base of aircarft data (bada) rev.3.9. Technical report, EUROCONTROL, 2011.

[29] C. Schultz, D. Thipphavong, and H. Erzberger. Adaptive trajectory prediction algorithm for climbing flights. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, August 2012.

[30] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38(4):367 – 378, 2002.

[31] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[32] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[33] G. Ridgeway. Generalized boosted models: A guide to the gbm package. *Update*, 1:1, 2007.

[34] C. Schultz, D. Thipphavong, and H. Erzberger. Adaptive trajectory prediction algorithm for climbing flights. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, August 2012.

Biographies

Richard Alligier received his Ph.D. (2014) degree in Computer Science from the "Institut National Polytechnique de Toulouse" (INPT), his engineer's degrees (IEEAC, 2010) from the french university of civil aviation (ENAC) and his M.Sc. (2010) in computer science from the University of Toulouse. He is currently assistant professor at the ENAC in Toulouse, France.

David Gianazza received his two engineer degrees (1986, 1996) from the french university of civil aviation (ENAC) and his M.Sc. (1996) and Ph.D. (2004) in Computer Science from the "Institut National Polytechnique de Toulouse" (INPT). He has held various positions in the french civil aviation administration, successively as an engineer in ATC operations, technical manager, and researcher. He is currently associate professor at the ENAC, Toulouse.

Nicolas Durand graduated from the Ecole polytechnique de Paris in 1990 and the Ecole Nationale de l'Aviation Civile (ENAC) in 1992. He has been a design engineer at the Centre d'Etudes de la Navigation Aérienne (then DSNA/DTI R&D) since 1992, holds a Ph.D. in Computer Science (1996) and got his HDR (french equivalent of tenure) in 2004. He is currently professor at the ENAC/MAIAA lab.