Learning Air Traffic Controller Workload from Past Sector Operations

David Gianazza

Learning Air Traffic Controller Workload from Past Sector Operations

David Gianazza
ENAC, F-31055 Toulouse, France
Univ. de Toulouse, IRIT/APO, F-31400 Toulouse, France

***Abstract*—In this paper, we compare several machine learning methods on the problem of learning a model of the air traffic controller workload from historical data. This data is a collection of workload mesurements extracted from past sector operations and of ATC complexity measurements computed from radar records and airspace data (sector geometry).**

We assume that the workload is low when a given sector is collapsed with other sectors into a larger sector, normal when it is operated as is, and high when it is split into smaller sectors assigned to several working positions.

This learning problem is modeled as a classification problem where the target variable is a workload category (low, normal, high) and the explanatory variables are the air traffic control (ATC) complexity metrics.

Several classifiers are compared on this problem: linear discriminant analysis, quadratic discriminant analysis, naive Bayes classifiers, neural networks, and gradient boosted trees. The performance of these models is assessed on a separate test set. The best methods show a rate of correct predictions around 82%.

***Keywords:* Air traffic control, ATC complexity, workload, machine learning, airspace configuration**

Introduction

How can we predict the workload of air traffic controllers operating a given sector and facing a more or less dense and complex traffic? This question is of crucial importance to the safety of the air traffic management (ATM) system at large. Overloads might lead to potentially dangerous situations where some conflicts might not be detected in time by the controllers.

Predicting the worlkoad with good accuracy is also a question of efficiency. In day-to-day operations, the airspace is dynamically reconfigured according to the controller workload. Underloaded sectors are collapsed to form larger sectors, and overloaded sectors are split into several smaller sectors operated separately. When it is not possible to absorb the traffic simply by reconfiguring sectors, the traffic is delayed or rerouted so as to avoid the congested areas. This needs to be done well in advance, usually before the aircraft take off. Predicting with greater accuracy which ATC sectors shall be operated at what time and which of these sectors might get overloaded would improve the whole traffic regulation process. This requires a realistic and accurate workload model, which is the subject of this paper.

This problem has been addressed in several ways since the beginings of air traffic control (ATC). Depending on the context and purpose, one might count the movements on an airport, or the number of aircraft within the boundaries of an en-route sector, or the incoming flow of traffic over a time period. Such basic metrics – and the associated threshold values (capacities) – provide simple and straightforward answers to the question of deciding whether the controllers are experiencing a normal workload when handling given traffic, or if they are overloaded.

However, it has been aknowledged for a long time that simple metrics, such as *aircraft count*, do not adequately reflect the complexity of air traffic control. ATC complexity covers dynamic aspects relative to the traffic, static aspects relative to the sector geometry and route network, and aspects relative to the air traffic control procedures.

In this paper, we are interested in the relationship between ATC complexity and workload. Both of these concepts are loosely defined in the literature ([1]), and before building models relating one to the other, we need to quantify them. Many ATC complexity indicators have been proposed in the literature [1], [2], [3], and this paper proposes nothing new in that matter. Quantifying the controller's workload has been done through different kinds of measures: physical activity ([4], [5]), physiological indicators ([6], [7], [8]), or subjective ratings ([9], [10]). Some of these indicators are difficult to interpret, and others are subject to biases (such as the recency effect denounced in [7], and the possibility of raters errors in the case of "over-the shoulder workload ratings" [11]). Collecting these data requires heavy experimental setups, often resulting in relatively small datasets and potential overfitting issues when trying to adjust a model on too few examples.

In order to avoid these drawbacks, we have proposed in previous work ([12], [13], [14]) to use historical records of the past sector operations to quantify the workload. These records are available in large quantity, for a large number of sectors. The information that can be extracted from the past sector operations is the following: we can assume that the workload was normal when the sector was operated, low when it was merged with other sectors to form a larger sector, and high when it was split into smaller sectors assigned to several working positions.

Several approaches have been tried to build models relating ATC complexity to workload. For example, taskload models ([15], [16]) compute the cumulative time required to execute control tasks. Linear regression models such as the popular dynamic density models ([17], [9]) approximate subjective workload ratings by a linear combination of a number of ATC complexity measures. Other works use a neural network instead of a linear model ([10]) to approximate subjective ratings. In previous work, we also used neural networks, but

our target variable was the workload measured from the past sector operations instead of subjective ratings. Considering an initial set of 27 complexity metrics found in the literature, we selected a subset of relevant metrics for the purpose of building a model that could be used to predict future airspace configurations ([12], [13], [14]). We showed that this concept was feasible and could be used to forecast airspace configurations that were much more realistic than the actual sector opening schedules made by the Flow Management Positions ([18], [19]). The concept was demonstrated on a mock-up HMI using static data ([20])

The current paper further explores the relationship between complexity and workload by comparing the performances of several machine learning methods using a fixed set of ATC metrics as input and the workload observed from past sector operations as target variable. The problem of learning the workload is here considered as a classification problem, where the target variable is a workload category (low, normal, or high). The methods being compared are the following: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), naive Bayes classifier (NBayes), neural networks (NNet), and gradient boosted trees (GBM).

The remainder of this paper is organized as follows: Section I is a short introduction to machine learning. Section II describes the methods used in this study. The data and experimental setup are described in section III, and the results in section IV. The expected benefits of the proposed approach are discussed in section V. The paper concludes with a brief summary of our findings and the perspectives of future works.

I. A Short Introduction to Machine Learning

This section is a brief introduction to machine learning. The reader may refer to [21], [22], [23] for a more thorough view of this active research field.

Learning from data with a computer can be done in different ways, through supervised learning, unsupervised learning, or reinforcement learning. In reinforcement learning, a sofware agent takes actions in a given environment so as to maximize a cumulative reward. In supervised or unsupervised learning, given some features x of an observed phenomenon, the objective is to learn a model from a set of examples $(\mathrm{x}_1, \ldots, \mathrm{x}_N)$. Unsupervised learning considers the explanatory variables x either to produce clusters of data, or to estimate the probability density of x, using the examples $(\mathrm{x}_1, \ldots, \mathrm{x}_N)$. In supervised learning, we assume a relationship $\mathrm{y} = f(\mathrm{x})$ between x and a target variable y, and we use examples of the outputs $(\mathrm{y}_1, \ldots, \mathrm{y}_N)$ associated with the inputs $(\mathrm{x}_1, \ldots, \mathrm{x}_N)$ to learn a model h approximating f.

In this paper, supervised learning techniques are used to predict the workload from ATC complexity indicators. The target variable y is here a workload category (low, normal, or high) and the input x is a vector of complexity indicators computed from the traffic or the sector geometry.

Such learning problems where the target is a categorical variable are usually referred to as classification problems, as opposed to regression problems where y is a floating-point value or a vector of floats.

Given a *loss* function ℓ such that $\ell(\mathrm{y}, \widehat{\mathrm{y}})$ is the cost of the error between the computed output $\widehat{\mathrm{y}} = h(\mathrm{x})$ and the observed data $\mathrm{y} = f(\mathrm{x})$, our objective is to choose h minimizing the following risk (i.e. the expected loss), where X and Y are the random variables from which are drawn x and y:

$$\mathcal{R}(h) = \mathbb{E}_{X,Y}\big[\ell(Y, h(X))\big] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(\mathrm{y}, h(\mathrm{x}))\, \mathbb{P}_{X,Y}(d\,\mathrm{x}, d\,\mathrm{y}) \tag{1}$$

A. Learning from a finite dataset

In practice, the joint distribution of X and Y is not known and one can only approximate f using a set of examples $S = \{(\mathrm{x}_1, \mathrm{y}_1), \ldots, (\mathrm{x}_N, \mathrm{y}_N)\}$ of finite size.

The most straightforward idea is then to select the model h minimizing the following empirical risk:

$$\mathcal{R}_{emp}(h, S) = \frac{1}{|S|} \sum_{(\mathrm{x}_n, \mathrm{y}_n)\in S} \ell(\mathrm{y}_n, h(\mathrm{x}_n)) \tag{2}$$

Unfortunately, minimizing the empirical risk on S might not lead to the most desirable model. The selected model might fit the examples $\{(\mathrm{x}_1, \mathrm{y}_1), \ldots, (\mathrm{x}_N, \mathrm{y}_N)\}$ of S very well, while performing poorly on new instances of x[1].

Statistical models can be more or less "flexible" when fitting the data, depending on their analytical expression. For example, a linear model is much less likely to overfit the data than a polynomial of high degree. Selecting the best model among a collection of models of various "flexibilities" requires a *bias-variance tradeoff*. Simple models tend to have a high bias (i.e. they are far from truth) and a low variance (i.e. the response of the model is about the same, whatever the training set used to tune it). In contrast, complex models have low bias and high variance. A complex model tuned on too few examples tends to overfit these examples and to perform poorly on new inputs.

B. Model assessment and selection

There are several ways to control overfitting and to find a suitable bias-variance tradeoff. One can use an information theory criterion, such as AIC (Akaike's "An Information Criterion") or BIC (Schwartz's Bayesian Information Criterion). These asymptotic criteria add a penalty P depending on the model complexity to the empirical risk $\mathcal{R}_{emp}(h, S)$ defined in equation (2). The model having the lowest value of $\mathcal{R}_{emp}(h, S) + P$ is selected.

Another way to proceed is to assess empirically the generalization error. Let us denote $\mathcal{A}$ the algorithm used to learn a model from a dataset S. In holdout cross-validation, the initial dataset S is split into two sets: a training set S_T used to learn the models, and another set S_V used to assess the holdout validation error Err_{val} as defined by the equation below:

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

[1]For example, let us assume we fit a polynomial curve on 10 points. For this regression problem, a polynom of degree 9 will fit exactly the examples, but will give poor predictions at other points. For a classification problem, the same overfitting problem might occur when using a K-nearest-neighbours method with $K = 1$.

The model having the lowest holdout validation error is selected.

K-fold cross-validation is another popular empirical method, where the dataset S is partitioned into k folds $(S_i)_{1\leq i\leq k}$. Let us denote $S_{-i} = S\backslash S_i$. In this method, k separate predictors $\mathcal{A}[S_{-i}]$ are learned from the k training sets S_{-i}. The mean of the holdout validation errors is computed, giving us the cross-validation estimation below:

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

When used for model selection, cross-validation can be performed successively on a collection of models. The model having the best cross-validation error is selected.

C. Hyperparameter tuning

In many methods, the bias-variance tradeoff is controlled through one or several parameters. For example, one can think of the number of hidden units in a neural network, or the weight decay hyperparameter (see subsection II-C). Hyperparameter values can be selected through cross-validation.

Let us denote λ the vector of hyperparameters of an algorithm $\mathcal{A}_\lambda$. In this paper, a 10-fold cross-validation has been used to tune hyperparameters, as described in algorithm 1.

Algorithm 1 Hyperparameter tuning for an algorithm $\mathcal{A}_\lambda$ and a set of examples T (training set).

function TUNEGRID($\mathcal{A}_\lambda$,$grid$)[T]
  $\lambda^* \leftarrow \underset{\lambda\in grid}{argmin}\ CV_{10}(\mathcal{A}_\lambda, T)$
  return $\mathcal{A}_{\lambda^*}[T]$
end function

II. Methods used in this study

The following machine learning methods have been applied to our workload prediction problem: linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), naive Bayes classifier (NBayes), neural networks (NNet), and gradient boosted trees (GBM). These methods are described in the following sections.

We used the statistical software `R`, and more specifically the `lda` and `qda` functions of the `MASS` package, the `naiveBayes` function of package `e1071`, the `nnet` neural network of the `Nnet` library, and the `Xgboost` library for gradient boosted trees.

A. Linear and quadratic discriminant analysis

Linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) are methods modeling the joint probability density $p(\mathrm{x}, \mathrm{y})$. Since we are considering classification problems, where y is a category (or class) $\mathrm{y} \in \{\mathcal{C}_1, \dots \mathcal{C}_P\}$, we denote $p(\mathrm{x}, \mathcal{C}_k)$ the joint density, and we have:

$$p(\mathrm{x}, \mathcal{C}_k) = p(\mathrm{x}\mid \mathcal{C}_k)\ \mathbb{P}(\mathcal{C}_k)$$

The prior probability $\mathbb{P}(\mathcal{C}_k)$ can be assessed empirically by counting the occurences of class $\mathcal{C}_k$ in the data, or it can be defined by the user. In linear and quadratic discriminant analysis, a gaussian density is assumed for $p(\mathrm{x}\mid\mathcal{C}_k)$.

$$\begin{aligned}&\mathbb{P}(\mathcal{C}_k) = \pi_k \\ &p(\mathrm{x}|\mathcal{C}_\mathrm{k}) = (2\pi)^{-\frac{P}{2}}\ |\Sigma|^{-\frac{1}{2}}\ \exp\left\{-\frac{1}{2}(\mathrm{x}-\mu_k)^T\Sigma_k^{-1}(\mathrm{x}-\mu_k)\right\}\end{aligned} \tag{5}$$

In LDA, the covariance matrix $\Sigma_k = \Sigma$ is the same for all classes $\mathcal{C}_k$, whereas QDA assumes a different covariance matrix for each class.

With the above assumptions, the posterior probability $\mathbb{P}(\mathcal{C}_j|x)$ of each class j can be computed using the Bayes' theorem:

$$\mathbb{P}(\mathcal{C}_j|x) = \frac{p(x|\mathcal{C}_j)\,\mathbb{P}(\mathcal{C}_j)}{p(x)} = \frac{p(x|\mathcal{C}_j)\,\mathbb{P}(\mathcal{C}_j)}{\sum_k p(x|\mathcal{C}_k)\,\mathbb{P}(\mathcal{C}_k)}$$

It can be easily verified, by computing $\ln\frac{\mathbb{P}(\mathcal{C}_k|x)}{\mathbb{P}(\mathcal{C}_l|x)}$, that the gaussian assumption leads to a linear boundary between classes $\mathcal{C}_k$ and $\mathcal{C}_l$ when all classes share a same covariance matrix Σ (LDA). The quadratic term is maintained when we assume different covariance matrices for the classes (QDA).

Training LDA or QDA models is done by computing maximum-likelihood estimates for the parameters π_k, μ_k and Σ_k.

B. Naive Bayes classifier

The naive Bayes classifier computes the posterior probabilities $\mathbb{P}(\mathcal{C}_j|\,\mathrm{x})$ using the Bayes rule and assuming conditional independance of the explanatory variables. For an input vector $\mathrm{x} = (x_1, \dots, x_D)^T$ of dimension D, this conditionnal independance is expressed as follows:

$$p(\mathrm{x}\,|\mathcal{C}_j)\ =\ \prod_{d=1}^{D} p(x_d|\mathcal{C}_j)$$

In addition, the naive Bayes classifier generally assumes a gaussian distribution for each of these variables. With these assumptions, the model is very much the same as the LDA model seen in the previous section, except that the off-diagonal terms of the correlation matrix Σ are null.

C. Neural networks for classification

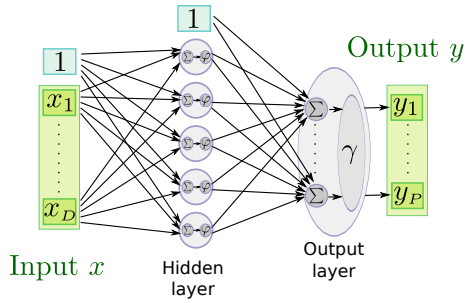

Figure 1: Example of a feed-forward network with one hidden layer

Figure 1 shows an example of a feed-forward neural network with one hidden layer such as the ones used in this study.

In our case, the output is a vector $\mathrm{y} = (y_1, y_2, y_3)$ modeling the probabilities to belong to the low, normal, or high workload categories.

In order to distinguish the output computed by the network from the target values found in our examples, we shall from now on denote $\widehat{\mathrm{y}}$ the computed output, and y the target value.

The network output can be interpreted as a vector of posterior class-membership probabilities, given the input x:

$$\widehat{\mathrm{y}} = \begin{pmatrix} \widehat{y}_1 \\ \vdots \\ \widehat{y}_P \end{pmatrix} = \begin{pmatrix} \mathbb{P}(\mathcal{C}_1|\mathrm{x}) \\ \vdots \\ \mathbb{P}(\mathcal{C}_P|\mathrm{x}) \end{pmatrix}$$

With this interpretation, and as the classes are non-overlapping, we must have:

$$\begin{aligned} &\widehat{y}_p \in [0,1], \quad \forall p \in \{1,\ldots,P\} \\ &\sum_{p=1}^{P} \widehat{y}_p = \sum_{p=1}^{P} \mathbb{P}\left(\mathcal{C}_p|\mathrm{x}\right) = 1 \end{aligned} \tag{6}$$

One way to ensure the conditions of equation (6) is to take $\widehat{\mathrm{y}} = \gamma(\mathrm{a})$ where γ is the *softmax* function described below:

$$\gamma \ : \begin{pmatrix} a_1 \\ \vdots \\ a_P \end{pmatrix} \longrightarrow \begin{pmatrix} \frac{\exp\{a_1\}}{\sum_{p=1}^{P}\exp\{a_p\}} \\ \vdots \\ \frac{\exp\{a_P\}}{\sum_{p=1}^{P}\exp\{a_p\}} \end{pmatrix} \tag{7}$$

In equation 7, each a_p is a weighted sum of the outputs of the hidden layer.

Each hidden unit j of the hidden layer computes a value $z_j = \varphi(a_j)$ where a_j is a weighted sum of $1, x_0, \ldots, x_D$. A common choice for φ is to take the sigmoid function $\varphi(a_j) = \frac{1}{1+e^{-a_j}}$, or the hyperbolic tangent function $\tanh$.

Training the neural network on a set of examples $S_T = \{(\mathrm{x}_1, \mathrm{y}_1), \ldots, (\mathrm{x}_N, \mathrm{y}_N)\}$ amounts to minimizing an error function depending on the network weights. These weights are tuned so as to minimize the error between the computed outputs $\widehat{\mathrm{y}}_1, \ldots, \widehat{\mathrm{y}}_N$ and the target values $\mathrm{y}_1, \ldots, \mathrm{y}_N$. The error function can be chosen according to the maximum-likelihood principle, or to the maximum posterior, for example.

In our classification problem, and under some normality assumptions, the maximum-likelihood principle leads to minimize the cross-entropy error function given in equation (8), where w is the vector of neural network weights.

$$\mathcal{E}(\mathrm{w}) = -\sum_{n=1}^{N}\sum_{p=1}^{P} y_{np} \ln \frac{\widehat{y}_{np}(\mathrm{w})}{y_{np}} \tag{8}$$

The maximum posterior principle leads to a regularized cross-entropy error, described in equation (9).

$$\mathcal{E}_\lambda(\mathrm{w}) = -\sum_{n=1}^{N}\sum_{p=1}^{P} y_{np} \ln \frac{\widehat{y}_{np}(\mathrm{w})}{y_{np}} \ + \ \lambda \sum_{j=1}^{D} w_j^2 \tag{9}$$

The regularization performed through the additional term in the right-hand side of equation (9) is known in the neural network literature as the *weight decay* method.

For the neural networks $\mathrm{NNet}_{N_h,\lambda}$, the hyperparameters allowing us to select a suitable bias-variance tradeoff are the number of hidden units N_h and the factor λ in equation (9).

D. Gradient boosting (GBM) applied to classification trees

The stochastic gradient boosting tree algorithm was introduced in [24], [25], [22]. It applies functional gradient descent to classification or regression trees ([26]).

The functional gradient descent is a *boosting* technique. The model h is iteratively improved. Denoting h_m the current model at iteration m, we consider the opposite gradient of the loss $g_i = -\frac{\partial \ell(\hat{y}, y_i)}{\partial \hat{y}}\left(h\left(x_i\right), y_i\right)$. A model g is then tuned to fit this opposite gradient, using a set of examples $(x_i, g_i)_{1 \le i \le n}$. The model h is then updated as follows : $h_{m+1}(x) = h_m(x) + \rho g(x)$, where ρ is a constant minimizing the empirical risk. The next iteration repeats the same procedure for h_{m+1} until a maximum number of iterations is reached. In stochastic gradient boosting, the dataset is randomly resampled at each iteration.

In the Gradient Tree Boosting, the machine learning algorithm boosted by the functional gradient descent is a classification or regression tree algorithm. Before continuing our description of gradient boosted trees, let us say a few words on classification and regression trees (CART) which were introduced by Breiman in [26]. In this algorithm, a binary tree is used to represent a binary recursive partition of the input space. At each node, the input space is split in two regions according to a condition $x_j \le s$. The J leaves of this tree describe a partition $(R_j)_{1 \le j \le J}$ of the input space. Each region R_j is associated to a constant γ_j. In the case of regression, it will be a constant float value (usually the average value of the examples in region R_j). In classification trees, γ_j will be a class (the most represented class among the examples in R_j). When the tree is used to make a prediction on a new input x, the value γ_j is returned when x falls into R_j.

CARTs have some advantages. For example, they are 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. This performance is greatly improved however when applying gradient boosting to CART.

In gradient boosted trees, the equation of the model update is the following, where ν is a *shrinkage* parameter:

$$h_m : \mathrm{x} \to h_{m-1}(\mathrm{x}) \ + \ \nu \sum_{R_j \in T_m} \gamma_{mj} \mathbb{1}_{R_j}(\mathrm{x}) \tag{10}$$

In our first experiments we used the `gbm` package in the `R` software. Due to major bugs[2] we switched to the `Xgboost` package.

We can denote $\mathrm{GBM}_{(m,J,\nu)}$ the gradient boosted tree 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 final model obtained after boosting is a sum of regression or classification trees. J allows us to control the interaction between variables, as we have $J-1$ variables at most in each tree. ν is the learning rate. In [22] (chap. 10), it is recommended to take small values for the shrinkage parameter

[2]The `gbm` software suffers from a memory leak when using multinomial distributions. This known bug has not been solved by the `gbm` developpers yet.

($\nu < 0.1$) and small values for J as well ($4 \leq J \leq 8$). The hyperparameter grid used for this algorithm and the others is presented in section III-D.

III. Data and experimental setup

A. Explanatory variables

In this study, ATC complexity indicators are used as inputs to our models. In previous works ([12], [28], [14]), we selected 6 basic complexity metrics among 27 metrics found in the literature. We used a principal component analysis to reduce the dimensionality of the inputs, and then selected the most relevant metrics related to the significant components. The 6 metrics that were found to be the most relevant *for the purpose of building airspace configuration prediction models* are the following:

- *vol*, the airspace volume of the considered ATC sector,
- *nb*, the number of aircraft within the sector boundaries at time t,
- *flow15*, the incoming traffic flow within the next 15 minutes,
- *flow60*, the incoming traffic flow within a 1 hour time horizon,
- *avg_vs*, the average absolute vertical speed of the aircraft within the sector,
- *inter_hori*, the number of speed vector intersections with an angle greater than 20 degrees.

These metrics are fairly simple and can be computed from radar tracks and static sector data (geometrical boundaries).

In the current paper, we have chosen to use the same metrics. We compare several machine learning methods on historical data from 2016. The ATC complexity metrics are standardized so as to obtain explanatory variables with mean 0 and standard deviation 1. These standardized variables are used as input vector x in our models.

B. Target variable

The target variable y we are trying to predict with our models is a workload category: *low*, *normal*, or *high*. In order to build our examples, we extracted this workload variable from historical airspace configuration data. In many cases, the workload in an ATC sector s at a past time t can be quantified by considering how it was operated at that time t. We simply make the following assumptions about the relationship between sector operation and workload:

- *Low workload* when sector s is collapsed with other sectors to form a larger sector operated on a working position,
- *Normal workload* when the sector s is operated as is,
- *High workload* when s is split into several smaller sectors operated on different working positions.

The other possible states – such as when a part of s is collapsed with one sector and another part is collapsed with another sector – are useless for quantifying the workload and are not used.

C. Datasets and selection procedure

The datasets used in this study are built from radar tracks and recorded sector operations from two weeks in October 2016, from the five french ATC control centers (Aix, Bordeaux, Brest, Paris, and Reims). For any given day of traffic in that period, only the ATC sectors operated at one moment or another during the day are used to build examples. Sectors that were not opened that day are not used to build examples.

For a given ATC sector, we first consider the time windows during which the sector was actually operated. Then, for each such time window of duration d minutes, we select all patterns $(\mathrm{x}_n, \mathrm{y}_n)$ within this window, and also all the patterns in the d minutes before and the d minutes after, within the limits of 0h-24h. We repeat this procedure for all time windows and ATC sectors. We then add randomly drawn samples (with equal probability) until the three classes "low", "normal", and "high" are equally represented in our data.

Our data is split into two subset: a *training set* made of the first week, from October 13^{th} to 19^{th}, and a *test set* made of the second week, from november 20^{th} to 26^{th}. The training set is used to tune models and the test set is used to assess the performance of tuned models. Our training set comprises $140\,168$ examples, with complexity and workload measurements from 511 different ATC sectors. The test set is made of $137\,564$ examples, from 513 ATC sectors.

For the methods requiring hyperparameter tuning, a 10-fold cross-validation is performed on the training set. This cross-validation is stratified so that every class is approximately equally represented in each fold. In addition, the folds are built so that the examples relative to a same ATC sector are not put in different folds.

The whole process can be seen as a nested cross-validation, with an outer holdout cross-validation used for model selection, and an inner 10-fold cross-validation operating on the training set for hyperparameter selection.

D. Hyperparameter grids

The hyperparameter selection of the inner cross-validation is performed on the training set, using function *TuneGrid* of algorithm 1 and the grids described in table I.

Method	Hyperparameter grid
LDA	-
QDA	-
NBayes	-
$\text{NNet}_{(n,\lambda)}$	$N_h = \{15, 20, 25\}$ $\lambda = \{10,1,1e\text{-}1,1e\text{-}2,1e\text{-}3,1e\text{-}5,0\}$
$\text{GBM}_{(m,J,\nu)}$	$m = \{5000, 6000, 7000\}$ $J = \{2, 3, 4\}$ $\nu = \{1e\text{-}4,\ 5e\text{-}4,\ 1e\text{-}3,1e\text{-}2,1e\text{-}1\}$

Table I: Grid of hyperparameters used in our experiments.

IV. Results

Table II shows the rates of correct predictions for the different machine learning methods used in this study. These rates are computed from the *test set*, not from the *training set* that was used to tune each model.

The column labeled "Overall" shows the rate of correct classifications over all three classes. The rates per class are given in the corresponding other columns (*low*, *normal*, and *high*).

Method	Overall	Low	Normal	High
LDA	71.4	71.6	70.1	72.4
QDA	74.8	81.8	63.8	79.7
NBayes	76.8	83.2	68.8	79.1
NNet	81.9	75.7	78.6	92.0
GBM	81.8	74.4	79.3	92.1

Table II: Correct classifications rates

In our previous studies [12], [13], we trained a neural network on data samples from June 2003. The overall rate of correct classifications was about 83%, and the class-specific rates were about 65% only for the "normal" workload class, and about 90% for the two other classes. The new results shown in table II for the neural network trained on 2016 data are consistent with these previous results, concerning the overall rate of correct predictions. Our new 2016 dataset is built so that the three classes are equally represented in the data, which was not the case in our previous works. This might explain why the class-specific rates for NNet in table II are slightly different from our previous results.

V. Expected benefits for the ATM

We have shown in the previous sections that machine learning techniques can be succesfully used to train models that predict the air traffic controller workload from a few complexity measures. What benefits for the Air Traffic Management (ATM) can be expected from this approach?

The current metrics for workload assessment, such as the aircraft counts in a sector or the incoming flows of traffic, are easy to understand and simple to use for the ATM operators. In comparison, machine learning methods may appear more sophisticated and complex. What they do exactly and how they work is not always easily perceived. However, once trained, the resulting model is just a relatively simple mathematical formula that can be used to produce workload estimates from any given input traffic.

Having workload estimates is not sufficient, though. One is usually interested in defining a threshold value for the workload, below which the situation is considered safe, and above which the excessive workload is considered as potentially hazardous. Flow management operators often use the *sector capacity* for that purpose, or *monitoring values* for peak and sustained traffic, for short-term ATFCM[3] measures. One usually defines at least one capacity value per sector (or a set of monitoring values), or several ones depending on various conditions of operation (e.g. military activity, weather, etc).

When considering the impact of ATC complexity on workload thresholds, one way to proceed is to modulate these threshold values, taking into account the level of complexity. However, ATC complexity is a rather elusive concept that depends on several interacting factors. The fact that complexity is not clearly defined and not quantified in a single metric makes the implementation of this approach difficult.

The reader might have noticed that the notion of capacity – as a threshold value for the incoming flows – does not appear in this paper, nor does the notion of peak or sustain monitoring values usually associated with the aircraft count metric. In our approach, the model outputs are workload probabilities and one can choose a probability value as the threshold indicating a potential overload. This probability threshold is the same for all sectors and it does not change with the ATC complexity, because the workload model itself takes the complexity factors as inputs and weighs them. We no longer have to define empirically a specific set of threshold values for each ATC sector (more than 500 sectors in France) and possibly for each complexity level (assuming these levels can be defined). We now have only one common threshold probability p_o allowing to determine if any given sector is overloaded or not. Let us consider a machine learning model which output $\widehat{\mathrm{y}}(\mathrm{x})$ is a triple of probabilities $(\mathbb{P}(\mathcal{C}_1|\mathrm{x}), \mathbb{P}(\mathcal{C}_2|\mathrm{x}), \mathbb{P}(\mathcal{C}_3|\mathrm{x}))$ where x is the input vector of complexity metrics and $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ are the *low*, *normal* and *high* workload categories. The usual decision functions such as *"if flow > capacity(sector,complexity) then overload"* are now replaced by the following decision function: *if* $\mathbb{P}(\mathcal{C}_3|\mathrm{x}) > p_o$ *then overload*

Having defined probability thresholds for "overload" and "underload", one can compute a prediction of the optimal sector openings, based on workload estimates. This is done by observing how the workload evolves in the prediction time window, in the ATC sectors of the current configuration. If the workload probabilities go beyond the "underload" or "overload" thresholds, a new optimal partition of the airspace is computed based on the workload model. This operation is repeated until a sequence of sector configurations is obtained, covering the time interval for the prediction (see [18], [19]).

Assuming the data used to train the workload model is representative of the variety of sectors and traffic situations and of the way sectors are operated, the machine learning approach proposed in this paper could improve the accuracy of workload predictions in many applications. A typical example is the elaboration of realistic predictions of ATC sector openings based on the estimated workload, as explained above. Another possible application is a *what-if* function allowing the operator to evaluate the impact of ATFCM measures. Assuming the operator detects a potential overload in an ATC sector, he or she could check if rerouting or delaying some flights could alleviate the workload in this sector, and see the impact on the other sectors. There is no doubt an improved workload prediction would highly benefit the ATM in terms of cost-efficiency and safety.

Conclusion

Let us now conclude this paper by summarizing our approach and our findings. We have compared several machine learning methods on the problem of learning workload prediction models from historical data. The examples used to learn our models were made of ATC complexity measurements computed from radar records and sector data, and workload measurements extracted from past ATC sector operations.

[3]ATFCM: Air Traffic Flow and Capacity Management

The three levels (low, normal, high) only give a rough indication of the workload, when compared with subjective ratings with 5 to 7 workload levels. However, the chosen target variable is probably less subject to a number of biases than subjective ratings or other workload measurements. Furthermore, subjective ratings are usually recorded during normal sector operations, and only for a few sectors. Little data is collected concerning overloads or underloads outside the usual domain of operation of the sector. The workload measurement we propose has the advantage of being easily available, in large quantities and for a great number of ATC sectors, because it can be directly extracted from historical records of past sector operations.

The results of the current study show that the choice of a neural network model in our previous works, which was not motivated by any comparative study at the time, is justified. Neural networks (Nnet) and gradient boosted trees (GBM) do perform better than linear models for classification (LDA) or quadratic discriminant analysis (QDA), with an overall rate of correct classifications of about $82\ \%$. The rates per class, for these two best methods, range from $75\ \%$ for the low workload class to $92\ \%$ for the high workload class.

In future works, we might want to examine more closely the generalization performance of our models on elementary airspace sectors that cannot be split into smaller sectors. For such sectors, our dataset contains only examples of the two classes "low" and "normal." It is well known in machine learning that a model can only be as good as the data that was used to produce it. So our models might not correctly detect overloads for such elementary airspace sectors. If so, a possible workaround might be to produce some artificial data samples of the "high" workload class for such sectors. This would force our model to correctly assess the boundary between normal and high workload for these specific sectors.

Other work might consider the seasonal variability in our data. It would be interesting to compare the performances of a same model tuned several times on data samples of different months.

REFERENCES

[1] R.H Mogford, J. A. Guttman, S. L. Morrow, and P. Kopardekar. The complexity construct in air traffic control: A review and synthesis of the literature. Technical report, FAA Technical Center: Atlantic City, 1995.

[2] P. Kopardekar. Dynamic density: A review of proposed variables. FAA WJHTC internal document. overall conclusions and recommendations, Federal Aviation Administration, 2000.

[3] B. Hilburn. Cognitive complexity in air traffic control, a litterature review. Technical report, Eurocontrol experimental centre, 2004.

[4] I. V. Laudeman, S. G. Shelden, R. Branstrom, and C. L. Brasil. Dynamic density: An air traffic management metric. Technical report, 1999.

[5] A. Majumdar, W. Y. Ochieng, G. McAuley, J.M. Lenzi, and C. Lepadetu. The factors affecting airspace capacity in europe: A framework methodology based on cross sectional time-series analysis using simulated controller workload data. In *Proceedings of the 6th USA/Europe Air Traffic Management R & D Seminar*, 2005.

[6] J.H. Crump. Review of stress in air traffic control: Its measurement and effects. *Aviation, Space and Environmental Medecice*, 1979.

[7] P. Averty, S. Athènes, C. Collet, and A. Dittmar. Evaluating a new index of mental workload in real ATC situation using psychological measures. Note CENA NR02-763, CENA, 2002.

[8] Caroline Martin, Julien Cegarra, and Philippe Averty. Analysis of mental workload during en-route air traffic control task execution based on eye-tracking technique. In *International Conference on Engineering Psychology and Cognitive Ergonomics*, pages 592–597. Springer, 2011.

[9] P. Kopardekar and S. Magyarits. Measurement and prediction of dynamic density. In *Proceedings of the 5th USA/Europe Air Traffic Management R & D Seminar*, 2003.

[10] G.B. Chatterji and B. Sridhar. Measures for air traffic controller workload prediction. In *Proceedings of the First AIAA Aircraft Technology, Integration, and Operations Forum*, 2001.

[11] C. Mannings, S. Mill, C. Fox, E. Pfleiderer, and H. Mogilka. The relationship between air traffic control events and measures of controller taskload and workload. In *Proceedings of the 4th Air Traffic Management Research & Developpment Seminar*, 2001.

[12] D. Gianazza and K. Guittet. Evaluation of air traffic complexity metrics using neural networks and sector status. In *Proceedings of the 2nd International Conference on Research in Air Transportation*. ICRAT, 2006.

[13] D. Gianazza and K. Guittet. Selection and evaluation of air traffic complexity metrics. In *Proceedings of the 25th Digital Avionics Systems Conference*. DASC, 2006.

[14] D. Gianazza. Smoothed traffic complexity metrics for airspace configuration schedules. In *Proceedings of the 3nd International Conference on Research in Air Transportation*. ICRAT, 2008.

[15] GM Flynn, A Benkouar, and R Christien. Adaptation of workload model by optimisation algorithms. Technical report, Eurocontrol, 2005.

[16] Jerry D Welch, John W Andrews, Brian D Martin, and Banavar Sridhar. Macroscopic workload model for estimating en route sector capacity. In *Proc. of 7th USA/Europe ATM Research and Development Seminar, Barcelona, Spain*, 2007.

[17] B. Sridhar, K. S. Sheth, and S. Grabbe. Airspace complexity and its application in air traffic management. In *Proceedings of the 2nd USA/Europe Air trafic Management R&D Seminar*.

[18] D. Gianazza, C. Allignol, and N. Saporito. An efficient airspace configuration forecast. In *Proceedings of the 8th USA/Europe Air Traffic Management R & D Seminar*, 2009.

[19] D. Gianazza. Forecasting workload and airspace configuration with neural networks and tree search methods. *Artificial Intelligence Journal, Elsevier*, 174(7-8):530–549, may 2010.

[20] N. Saporito, C. Hurter, D. Gianazza, and G. Beboux. A participatory design for the visualization of airspace configuration forecasts. In *Proceedings of the 4th International Conference on Research in Air Transportation*, 2010.

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 6. Springer, 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] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[24] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

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

[26] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[27] Greg Ridgeway. Generalized boosted models: A guide to the gbm package, 2007. *URL http://cran. open-source-solution. org/web/packages/gbm/vignettes/gbm. pdf*.

[28] D. Gianazza. Airspace configuration using air traffic complexity metrics. In *Proceedings of the 7^{th} USA/Europe Seminar on Air Traffic Management Research and Development*, 2007. best paper of "Dynamic Airspace Configuration" track.

BIOGRAPHY

David Gianazza received his two engineer degrees (1986, 1996) from the "École Nationale de l'Aviation Civile" (ENAC) and his M.Sc. (1996) and Ph.D. (2004) in Computer Science from the "Institut National Polytechnique de Toulouse" (INPT). He has obtained his HDR (a french post-doctoral degree similar to tenure) in 2016 from the 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.