Predictive Distribution of the Mass and Speed Profile to Improve Aircraft Climb Prediction

Richard Alligier

ENAC

USA/Europe Air Traffic Management R&D Seminar, 2019
Problem Considered

- Ground-based: mass and speed intent not available
- Future trajectory of climbing aircraft (assuming no level-off)
Problem Considered

- Ground-based: mass and speed intent not available
- Future trajectory of climbing aircraft (assuming no level-off)
Problem Considered

- Ground-based: **mass** and **speed intent** not available
- Future trajectory of **climbing** aircraft (assuming no level-off)

![Graph showing flight trajectory](image)

Physical model (BADA) + Machine Learning models
Problem Considered

- Ground-based: 
  - mass and speed intent not available
- Future trajectory of climbing aircraft (assuming no level-off)

\[ \mathbb{P}(\text{future trajectory} | \text{FLIGHT} = \text{flight}) \]

Physical model (BADA) + Machine Learning models
What is Inside the “Physical Model + ML Models” Box?

**Ideas**

1. Compute trajectory distribution using mass and speed distributions
   - Quantification of Aircraft Trajectory Prediction Uncertainty Using Polynomial Chaos Expansions, Casado et al., DASC2017
   - Interdependent Uncertainty Handling in Trajectory Prediction, Zeh et al., Aerospace2019

2. ML models to predict distribution of mass and speed intent
   - Predicts distribution specific to the considered flight
   - Alligier Predictive Distribution of the Mass and Speed Profile to Improve Aircraft Climb Prediction ATM 2019
What is Inside the “Physical Model + ML Models” Box?

**Introduction**

What is Inside the “Physical Model + ML Models” Box?

**Physical model (BADA)**

- Probability distribution of future trajectory

---

**Ideas**

1. Compute trajectory **distribution** using mass and speed **distributions**

   - Quantification of Aircraft Trajectory Prediction Uncertainty Using Polynomial Chaos Expansions, Casado et al., DASC2017
   - Interdependent Uncertainty Handling in Trajectory Prediction, Zeh et al., Aerospace2019
What is Inside the “Physical Model + ML Models” Box?

- Physical model (BADA)
- Probability distribution of future trajectory

Ideas

1. Compute trajectory distribution using mass and speed distributions
   - Quantification of Aircraft Trajectory Prediction Uncertainty Using Polynomial Chaos Expansions, Casado et al., DASC2017
   - Interdependent Uncertainty Handling in Trajectory Prediction, Zeh et al., Aerospace2019
What is Inside the “Physical Model + ML Models” Box?

**Ideas**

1. **Compute trajectory distribution** using mass and speed distributions
   - Quantification of Aircraft Trajectory Prediction Uncertainty Using Polynomial Chaos Expansions, Casado et al., DASC2017
   - Interdependent Uncertainty Handling in Trajectory Prediction, Zeh et al., Aerospace2019

2. **ML models to predict distribution of mass and speed intent**
   - Predicts distribution specific to the considered flight
Focus of this Work

Predict Tailored Distribution of Mass and Speed Intent

- Information about the considered flight → Machine Learning model → Predictive distribution of mass and speed intent

\[
\text{mass} \sim \quad \text{speed intent} \sim
\]
Introduction

Previous Work Predicting a Mass Distribution

Previously Tested Methods and their Limits

- Particle Filtering
  - Only rely on the physical model
  - Do not take advantage of flight plan variables (e.g. “airline”, “airport”, etc)

  Aircraft Mass and Thrust Estimation Using Recursive Bayesian Method, Sun et al., ICRAT2018

- Gaussian Process Regression
  - Well suited for this task
  - Do not scale well to large historical database

  Statistical Modeling of Aircraft Takeoff Weight, Chati et al., ATM2017
In this Work

Method

- Predicted distributions are Gaussian $\mathcal{N}(\mu, \sigma)$
- Use Neural Networks to predict $\mu$ and $\sigma$

Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles, Lakshminarayanan et al., NeurIPS2017

$$(\mu, \sigma) = \text{NN}_{\text{weights}} \text{ (information about considered flight)}$$

- 10 past positions
- 10 past speeds
- weather data
- flight plan data
- Mode-S address ($\simeq$ tail number)

Data Used to Train/Validate

Large ADS-B database containing millions of flights
1 Data Preparation (No Machine Learning Used Yet)

2 Applying Machine Learning

3 Results
Data Preparation (No Machine Learning Used Yet)

Applying Machine Learning

Results
Data Preparation (No Machine Learning Used Yet)

Data Sources

Raw Data
- ADS-B data from The OpenSky Network (year 2017)
- Weather forecast grids from NOAA

Filtering/Sampling
- Keep only climbing segments with duration $\geq 750$ s
- Variables are smoothed (Kalman filter)
- One point each 15 seconds (linear interpolation)

Variables
- We have the altitude $H_p$, the true airspeed $V_a$, the temperature $T$, etc.
- We **DO NOT** have the mass nor the speed intent $(\text{cas}_1, \text{cas}_2, M)$
Extract Mass from One Trajectory [Alligier et al., 2014]

Max-climb thrust assumed

\[
\arg\min_{\text{mass}} \sum_{i=1}^{n} \left( \frac{\text{Power}(H_{p_i}, T_i, V_{a_i}, \text{mass})}{\text{mass}} - \frac{\Delta\text{energy}_i}{\Delta t_i} \right)^2
\]

Only the BADA physical model was used, no machine learning yet!
Extract \((c_{as_1}, c_{as_2}, M)\) from One Trajectory
[Alligier et al., 2015]

Speed schedule parametrized by \((c_{as_1}, c_{as_2}, M)\)

\[
\arg\min_{(c_{as_1}, c_{as_2}, M)} \sum_{i=1}^{n} \left( f(c_{as_1}, c_{as_2}, M; H_{p_i}, T_{i}) - T_{AS_i} \right)^2
\]

Only the atmosphere model was used, no machine learning yet.
Data Preparation (No Machine Learning Used Yet)

Extract \((c_{as1}, c_{as2}, M)\) from One Trajectory

[Alligier et al., 2015]

Speed schedule parametrized by \((c_{as1}, c_{as2}, M)\)

![Graph showing speed schedule parametrized by \((c_{as1}, c_{as2}, M)\).](image)

- \(c_{as1} = 220\) kt, \(c_{as2} = 270\) kt, \(M = 0.75\)
- \(c_{as1} = 240\) kt, \(c_{as2} = 285\) kt, \(M = 0.79\)

Only the atmosphere model was used, no machine learning yet!
Extract \((\text{cas}_1, \text{cas}_2, M)\) from One Trajectory
[Alligier et al., 2015]

Speed schedule parametrized by \((\text{cas}_1, \text{cas}_2, M)\)

Observed Speed \((\text{cas}_1, \text{cas}_2, M)\) Speed Profile

\[
\text{argmin}_{(\text{cas}_1, \text{cas}_2, M)} \sum_{i=1}^{n} (f(\text{cas}_1, \text{cas}_2, M; H_{pi}, T_i) - \text{TAS}_i)^2
\]

Only the atmosphere model was used, no machine learning yet!
Extract \((\text{cas}_1, \text{cas}_2, M)\) from One Trajectory
[Alligier et al., 2015]

Speed schedule parametrized by \((\text{cas}_1, \text{cas}_2, M)\)

\[
\arg\min_{(\text{cas}_1, \text{cas}_2, M)} \sum_{i=1}^{n} \left( f(\text{cas}_1, \text{cas}_2, M; H_{p_i}, T_i) - \text{TAS}_i \right)^2
\]

Only the atmosphere model was used, no machine learning yet!
At the End of the Data Preparation Phase

Obtained Climbing Trajectories Set

- One year of data (2017), covering 11 aircraft types
- 4.9 millions climbing trajectories decorated with mass + speed profile
- [https://opensky-network.org/datasets/publication-data](https://opensky-network.org/datasets/publication-data)
1. Data Preparation (No Machine Learning Used Yet)

2. Applying Machine Learning

3. Results
Predicted Distributions of mass, $\text{cas}_1$, $\text{cas}_2$ and $M$ are assumed:

- Gaussian
- Independent

For instance, for the mass:

$$\text{MASS}|X = \text{considered flight} \sim \mathcal{N}(\mu(\text{considered flight}), \sigma(\text{considered flight}))$$
Predicted Distributions of \textbf{mass}, \textit{cas}_1, \textit{cas}_2 \text{ and } \textit{M} \text{ are assumed:}

- Gaussian
- Independent

For instance, for the mass:

\[
\text{MASS}|X = \text{considered flight} \sim \mathcal{N}(\mu(\text{considered flight}), \sigma(\text{considered flight}))
\]

\textbf{Use of Neural Networks}

Learn \(\mu\) and \(\sigma\) functions
The dropout blocks are used to prevent over-fitting. At each iteration of the training process, only a randomly selected subset of the input features is used. This helps to reduce the risk of over-fitting by making the model less dependent on any single input feature.

Compared with mean values, the GBM method have typically reduced the RMSE by 56%, 49%, 39%, and 15%. Batch normalization is a technique that normalizes the inputs of each layer of a neural network. It hides the interaction between the weights and reduces the loss as a function of the weights. As a first-order approximation method improving the optimization process, the gradient descent method with adaptive learning rate and weight decay is used. The training phase consists in finding the weights minimizing the negative log-likelihood NLL. It is done using AdamW, a gradient descent method with adaptive learning rate and weight decay. The initial learning rate is found via the search method described in [41]. The remaining hyper-parameters are the learning rate decay, the weight decay, the number of hidden layers, the number of hidden units for each hidden layers, and the dropout rate. We tested 200 different sets of hyper-parameters. The tested hyper-parameters are randomly drawn.

The k-fold cross-validation usually provides a better assessment of the generalization error than a simple hold-out validation, nevertheless we chose the second approach here, using a cross-validation that might change through time if for instance new procedures for very specific reasons. The distribution of the trajectories is modeled using the GBM approach method. This latter has been tested in [28].

The network architecture is depicted in the figure. The input variables are embedded categorical input, numerical input, and encoded input. The manipulated vectors are in green whereas the operations are in black. The manipulated vectors are in green whereas the operations are in black. The input is at the top, the output is at the bottom. The manipulated vectors are in green whereas the operations are in black. The input is at the top, the output is at the bottom.

The predicted parameters are the predicted values of the mass and speed profile. These predicted parameters are then used to improve the aircraft climb prediction. All the code is implemented using the PyTorch library.
The dropout blocks are used to prevent over-fitting. At each iteration of the training process, only a randomly selected fraction of the input units are disabled, which helps to reduce the model's dependency on specific weights and thus prevents overfitting. This technique is particularly useful in deep learning models where the risk of overfitting is high due to the large number of parameters.

Compared with mean values, the GBM (Gradient Boosting Machines) method have typically reduced the RMSE by 56%, 49%, 39%, and 15% as reported in [28].

In the context of batch normalization, this technique normalizes the activations or responses at each layer, which helps to stabilize and speed up the training process. It is applied after each non-linear operation to ensure that the activations are scaled and shifted appropriately.

Network Architecture

![Network Architecture Diagram]

**Input variables**

The network architecture diagram above illustrates the input variables, including call signs, Mode-S addresses, and other relevant data. The input variables are processed through various layers, including embedding, sum, concatenate, linear, and LeakyReLU layers, which help in transforming and normalizing the data. The manipulated vectors are in green, whereas the operations are in red. The input is at the top, and the output is the predicted parameters at the bottom.

For instance, the number of hidden layers is drawn inside the network, and the returned vector are only controlled by two additional weights. The input is transformed through a series of operations, including sum and concatenate, before being fed into the first linear layer. This process is repeated for hidden layers 1 through n, and finally, the predictions are made using the linear output.
The dropout blocks are used to prevent over-fitting. At each iteration of the training process, only a randomly chosen sub-batch of data is used to compute the loss as a function of the weights. As a first-order approximation, this is done to improve the optimization process. The gradient descent technique relies on a first-order approximation of the loss function to update the weights of the different layers. To do so, all the weights are updated using a discrete uniform distribution. Such a random selection of data ensures that the model is not biased towards any particular part of the dataset.

In the context of the GBM approach, the model is trained on trajectories from January to August. The final model is selected using a cross-validation strategy. For each hyper-parameter, the neural network is trained on trajectories from January to August, and the hyper-parameters having the best result on these trajectories will be chosen. Then, using this selected network and the hyper-parameters, the model is tested on trajectories from September to October.

All the code is implemented using the PyTorch library. The initial learning rate is found via a grid search method, and the learning rate decay, the weight decay, the number of hidden layers, the number of hidden units for each hidden layer, and the dropout rate are tested. The best hyper-parameters are chosen using the cross-validation strategy. The initial learning rate is found using a grid search method, and the hyper-parameters are tested using a random search method.

All the results presented in this section have been computed using the PyTorch library. The network architecture consists of an input layer, several hidden layers, and an output layer. The input layer receives the categorical and numerical inputs, and the output layer computes the predicted values. The network architecture is designed to handle both categorical and numerical inputs, and the output layer uses a softmax function to compute the predicted values.

The network architecture consists of an input layer, several hidden layers, and an output layer. The input layer receives the categorical and numerical inputs, and the output layer computes the predicted values. The network architecture is designed to handle both categorical and numerical inputs, and the output layer uses a softmax function to compute the predicted values.
The dropout blocks are used to prevent over-fitting. At each iteration of the training process, only a randomly chosen subset of the weights is updated. This encourages the network to learn more robust features and helps to reduce over-fitting.

Compared with mean values, the Gradient Boosting Machines (GBM) approach has typically reduced the Root Mean Squared Error (RMSE) by 56%, 49%, 39%, and 15%.

---

**Network Architecture**

- **Input variables**
  - Callsign
  - Mode-S address
  - $H_p$
  - $V_a$

- **Sum of embeddings**

- **Concatenate**

- **Merged input**

- **Linear**

- **LeakyReLU**

- **Hidden layer $n$**

- **Linear**

- **Final output**
  - $\mu_{\text{mass}}$
  - $\mu_{\text{cas}_1}$
  - $\mu_{\text{cas}_2}$
  - $\mu_{\text{Mach}}$
  - $\sigma_{\text{mass}}$
  - $\sigma_{\text{cas}_1}$
  - $\sigma_{\text{cas}_2}$
  - $\sigma_{\text{Mach}}$

---

Applying Machine Learning
The dropout blocks are used to prevent over-fitting. At each iteration of the training process, only a randomly chosen sub-set is used. From [28], compared with mean values, the GBM method have typically reduced the RMSE by 56%, 49%, 39%, and 15%.

The batch loss as a function of the weights. As a first-order approximation method improving the optimization process. The gradient of the layers before one batch normalization the weights of the different layers. To do so, all the weights block aims to reduce the interaction between normalization data. The whole network is used to compute the prediction on new trajectories from January to October. The predicted parameters are the predicted values of mass, Mach number, and the hyper-parameters having the best result on these trajectories will be the chosen one. Then using this selected network is trained on trajectories from January to August. Then it is tested on the trajectories from September to October, and the hyper-parameters having the best result on these months of 2017 were used to build the predictive models. The ten first trajectories from November and December of the year 2017. The ten first validation, nevertheless we chose the second approach here, with the same distribution. It will mask the non-stationarity of the problem we are studying and the performance evaluation obtained will be too optimistic. For this reason, we chose a k-fold cross-validation usually provides a better assessment of the generalization error than a simple hold-out test set.

All the statistics in this section have been computed on the test set and the hyper-parameters having the best result on these trajectories from January to October and finally the validation set uses trajectories recorded from January to August, and the dropout rate. We tested 200 different sets of hyper-parameters. The tested hyper-parameters are randomly drawn: the initial learning rate is found via the search method weight decay ([41]), the training phase consists in finding the weights minimizing the negative log-likelihood NLL. It is done using AdamW, a gradient descent method with adaptive learning rate and no impact on the mean and variance of the vector returned by the block. The returned vector are only controlled by two additional weights inside the block.

All the code is implemented using the PyTorch library. Applying Machine Learning.
The dropout blocks are used to prevent over-fitting. At each iteration of the training process, only a randomly chosen subset of the weights is updated. This helps to prevent the model from becoming too sensitive to the specific data points used for training, which can lead to over-fitting. The batch normalization block is used to normalize the inputs to each hidden layer. This helps to improve the learning process by making the inputs more stable and easier for the model to learn from.

Compared with mean values, the GBM method has typically reduced the RMSE by 56%, 49%, 39%, and 15%.

The training phase consists in finding the weights minimizing the negative log-likelihood NLL. It is done using AdamW, a gradient descent method with adaptive learning rate and weight decay ([41]).

The initial learning rate is found via the search method ([43]). For each hyper-parameter, the neural network is trained on trajectories from January to August. After the training phase, the weights updates. Conceptually, an ensemble of networks is used to compute the prediction on new trajectories from January to October. It might change through time if for instance new procedures are applied at a specific airport. Using a cross-validation that is pessimistically biased, the final model is the one trained on data not used in the model building process. These results have been computed from all the trajectories in the months of 2017.

The statistics in this section have been computed on the test set. The predicted parameters are the predicted values compared with the “true” parameters extracted from the future trajectory. We will compare the neural network (NN) approach to the Gradient Boosting Machines (GBM) method.

The predicted values are compared with the “true” parameters extracted from the future trajectory. We will compare the neural network (NN) approach to the Gradient Boosting Machines (GBM) method.

For instance, the number of hidden layers is drawn inside the batch normalization block: the mean and variance of the vector returned by the batch normalization block will have no impact on the mean and variance of the vector returned by the next hidden layer. The manipulated vectors are in green whereas the operations applied to these vectors are in red. The input is at the top, the output is at the bottom. The manipulated vectors are in green whereas the operations applied to these vectors are in red. The input is at the top, the output is at the bottom.
How to Train it?!

Data set does not contain the true values of $\mu$ and $\sigma$!
How to Train it?!

Data set does not contain the true values of $\mu$ and $\sigma$!

**Maximum Likelihood Principle**

1. Assume a probability model parametrized by $w$ for $Y|X$:
   \[ p(Y = y|X = x) = f(y, x; w) \]
How to Train it?!

Data set does not contain the true values of $\mu$ and $\sigma$!

Maximum Likelihood Principle

1. Assume a probability model parametrized by $w$ for $Y|X$:
   \[ p(Y = y|X = x) = f(y, x; w) \]

2. Choose $w$ maximizing likelihood of the training set $T = (x_i, y_i)_{1 \leq i \leq n}$:
   \[ w \text{ maximizing } \prod_{i=1}^{n} p(Y = y_i|X = x_i) = \prod_{i=1}^{n} f(y_i, x_i; w) \]
Data set does not contain the true values of $\mu$ and $\sigma$!

**Maximum Likelihood Principle**

1. Assume a probability model parametrized by $w$ for $Y|X$:
   \[ p(Y = y | X = x) = f(y, x; w) \]

2. Choose $w$ maximizing likelihood of the training set $T = (x_i, y_i)_{1 \leq i \leq n}$:
   \[ w \text{ maximizing } \prod_{i=1}^{n} p(Y = y_i | X = x_i) = \prod_{i=1}^{n} f(y_i, x_i; w) \]

**Applying Maximum Likelihood Principle**

1. We assume $Y|X = x \sim \mathcal{N}(\mu(x; w), \sigma(x; w))$
How to Train it?!

Data set does not contain the true values of $\mu$ and $\sigma$!

Maximum Likelihood Principle

1. Assume a probability model parametrized by $w$ for $Y|X$:
   \[ p(Y = y | X = x) = f(y, x; w) \]

2. Choose $w$ maximizing likelihood of the training set $T = (x_i, y_i)_{1 \leq i \leq n}$:
   \[ w \text{ maximizing } \prod_{i=1}^{n} p(Y = y_i | X = x_i) = \prod_{i=1}^{n} f(y_i, x_i; w) \]

Applying Maximum Likelihood Principle

1. We assume $Y|X = x \sim \mathcal{N}(\mu(x; w), \sigma(x; w))$

2. Choose $w$ minimizing
   \[ \sum_{i=1}^{n} \frac{1}{2} \left( \frac{y_i - \mu(x_i; w)}{\sigma(x_i; w)} \right)^2 + \log \sigma(x_i; w) \]
1 Data Preparation (No Machine Learning Used Yet)

2 Applying Machine Learning

3 Results
### Experimental Setup

#### One Year of Data Split in Three

- **$S_{Train}$**: January, ..., August
- **$S_{Valid}$**: September, October
- **$S_{Test}$**: November, December

Select best hyper-parameter

- Train final model
- Evaluate final model

---

### In this Result Section

- All the statistics are computed on $S_{Test}$
- $x$ will refer to the explanatory variables
- $y$ will refer to the predicted variable
\[
\text{RMSE} = \sqrt{\frac{1}{|S_{\text{Test}}|} \sum_{(x,y) \in S_{\text{Test}}} (y - \text{predict}(x))^2}
\]

Two Methods:

- **NN** [this paper]: predicted value is \( \mu(x) \)
- **GBM** [Alligier and Gianazza, 2018]: Gradient Boosted Trees
Results

\[ \text{RMSE} = \sqrt{\frac{1}{|S_{\text{Test}}|} \sum_{(x,y) \in S_{\text{Test}}} (y - \text{predict}(x))^2} \]

Two Methods:

- **NN [this paper]:** predicted value is \( \mu(x) \)
- **GBM [Alligier and Gianazza, 2018]:** Gradient Boosted Trees

<table>
<thead>
<tr>
<th>factor</th>
<th>method</th>
<th>mass [kg]</th>
<th>cas_1 [kt]</th>
<th>cas_2 [kt]</th>
<th>Mach [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GBM</td>
<td>NN</td>
<td>GBM</td>
<td>NN</td>
<td>GBM</td>
</tr>
<tr>
<td>A320</td>
<td>1929</td>
<td>1953</td>
<td>10.10</td>
<td>10.58</td>
<td>8.85</td>
</tr>
<tr>
<td>B738</td>
<td>2508</td>
<td>2532</td>
<td>8.44</td>
<td>9.17</td>
<td>7.84</td>
</tr>
<tr>
<td>B77W</td>
<td>10742</td>
<td>10621</td>
<td>7.11</td>
<td>7.50</td>
<td>5.47</td>
</tr>
<tr>
<td>DH8D</td>
<td>738</td>
<td>720</td>
<td>6.95</td>
<td>7.04</td>
<td>11.53</td>
</tr>
</tbody>
</table>

**Conclusion:** NN slightly worse
Predict interval $I_{\gamma}(x)$ that should contain $Y$ with coverage probability $\gamma$:

$$\mathbb{P}(Y \in I_{\gamma}(x) | X = x) = \gamma$$
Build Prediction Interval

Predict interval $I_{\gamma}(x)$ that should contain $Y$ with coverage probability $\gamma$:

$$\mathbb{P}(Y \in I_{\gamma}(x) | X = x) = \gamma$$

- NN (this paper): $I_{\gamma}(x) = [\mu(x) - r_{\gamma}\sigma(x); \mu(x) + r_{\gamma}\sigma(x)]$
Build Prediction Interval

Predict interval $I_{\gamma}(x)$ that should contain $Y$ with coverage probability $\gamma$:

$$\mathbb{P}(Y \in I_{\gamma}(x) | X = x) = \gamma$$

- NN (this paper): $I_{\gamma}(x) = [\mu(x) - r_{\gamma}\sigma(x); \mu(x) + r_{\gamma}\sigma(x)]$
- GBM [Alligier and Gianazza, 2018]:
  $$I_{\gamma}(x) = [\text{predict}_{\text{GBM}}(x) - s/2; \text{predict}_{\text{GBM}}(x) + s/2]$$

$s$ such that GBM have same actual coverage probability as NN on $S_{\text{Test}}$.
Results

Build Prediction Interval

Predict interval $I_\gamma(x)$ that should contain $Y$ with coverage probability $\gamma$:

$$\mathbb{P}(Y \in I_\gamma(x) | X = x) = \gamma$$

- NN (this paper): $I_\gamma(x) = [\mu(x) - r_\gamma \sigma(x); \mu(x) + r_\gamma \sigma(x)]$

- GBM [Alligier and Gianazza, 2018]:
  
  $I_\gamma(x) = [\text{predict}_{\text{GBM}}(x) - s/2; \text{predict}_{\text{GBM}}(x) + s/2]$

$s$ such that GBM have same actual coverage probability as NN on $S_{\text{Test}}$

Mean Interval Size for $\gamma = 0.9$

<table>
<thead>
<tr>
<th>factor</th>
<th>mass [kg]</th>
<th>cas1 [kt]</th>
<th>cas2 [kt]</th>
<th>Mach [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>GBM</td>
<td>NN</td>
<td>GBM</td>
<td>NN</td>
</tr>
<tr>
<td>A320</td>
<td>6435</td>
<td>6290</td>
<td>40.29</td>
<td>26.61</td>
</tr>
<tr>
<td>B738</td>
<td>8202</td>
<td>7837</td>
<td>34.02</td>
<td>21.99</td>
</tr>
<tr>
<td>B77W</td>
<td>34892</td>
<td>32667</td>
<td>28.99</td>
<td>16.96</td>
</tr>
<tr>
<td>DH8D</td>
<td>2217</td>
<td>2129</td>
<td>26.06</td>
<td>18.11</td>
</tr>
</tbody>
</table>

Conclusion: Despite worse RMSE, NN intervals are smaller
Build Prediction Interval

Predict interval $I_\gamma(x)$ that should contain $Y$ with coverage probability $\gamma$:

$$\mathbb{P}(Y \in I_\gamma(x) \mid X = x) = \gamma$$

- **NN (this paper):** $I_\gamma(x) = [\mu(x) - r_\gamma \sigma(x); \mu(x) + r_\gamma \sigma(x)]$
- **GBM [Alligier and Gianazza, 2018]:**

$$I_\gamma(x) = [\text{predict}_{\text{GBM}}(x) - s/2; \text{predict}_{\text{GBM}}(x) + s/2]$$

$s$ such that GBM have same actual coverage probability as NN on $S_{\text{Test}}$

**Conclusion:** Despite worse RMSE, NN intervals are smaller
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{\text{Test}} \mid \sigma(x) \simeq \sigma\}$
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{\text{Test}} \mid \sigma(x) \simeq \sigma\}$

Conclusion: $\sigma(x)$ gives information on the expected error.
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{\text{Test}} \mid \sigma(x) \simeq \sigma\}$

$\text{RMSE}(S(\sigma)) = \sigma$
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{Test} \mid \sigma(x) \simeq \sigma\}$

$\text{RMSE}(S(\sigma)) = \sigma$
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{\text{Test}} \mid \sigma(x) \simeq \sigma\}$, $\text{RMSE}(S(\sigma)) = \sigma$

RMSE$(S(\sigma=8)) = \sigma$
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{\text{Test}} \mid \sigma(x) \simeq \sigma\}$, RMSE ($S(\sigma)$) = $\sigma$
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{\text{Test}} \mid \sigma(x) \simeq \sigma\}$, $\text{RMSE}(S(\sigma)) = \sigma$
Verifying that Observed RMSE Close to Predicted $\sigma(x)$

Let us consider $S(\sigma) = \{(x, y) \in S_{\text{Test}} \mid \sigma(x) \simeq \sigma\}$, $\text{RMSE}(S(\sigma)) = \sigma$

Conclusion: $\sigma(x)$ gives information on the expected error
Trajectories with Lowest/Highest $\sigma_{\text{mass}}(x)$

B738 flights with top-2 highest and bottom-2 lowest $\sigma_{\text{mass}}(x)$

Conclusion (qualitative analysis on a handful of examples):
- Low $\sigma(x)$: smooth typical energy rate profile
- High $\sigma(x)$: very low/high or non-smooth energy rate profile
Large Error Compared with the Predicted $\sigma_{\text{mass}}(x)$

$\text{MASS} \mid X = x \sim \mathcal{N}(\mu(x), \sigma(x))$

Very unlikely observation according to predicted $\mathcal{N}(\mu(x), \sigma(x))$

$\Leftrightarrow$ Very low $\mathbb{P}(\text{MASS} = \text{mass} \mid X = x)$
Large Error Compared with the Predicted \( \sigma_{\text{mass}}(x) \)

The B738 flight with the lowest \( \mathbb{P}(\text{MASS} = \text{mass}|X = x) \)
Large Error Compared with the Predicted $\sigma_{\text{mass}}(x)$

The B738 flight with the lowest $P(\text{MASS} = \text{mass}|X = x)$

Conclusion (qualitative analysis on a handful of examples):
Shift to constant ROC over-represented among erroneous predictions
Large Error Compared with the Predicted $\sigma_{\text{mass}}(x)$

The B738 flight with the lowest $\mathbb{P}(\text{MASS} = \text{mass}|X = x)$
Large Error Compared with the Predicted $\sigma_{\text{mass}}(x)$

The B738 flight with the lowest $\mathbb{P}(\text{MASS} = \text{mass} | X = x)$

![Graph showing energy rate vs. time](image-url)
Large Error Compared with the Predicted $\sigma_{\text{mass}}(x)$

The B738 flight with the lowest $P(\text{MASS} = \text{mass}|X = x)$

Conclusion (qualitative analysis on a handful of examples):
Shift to constant ROC over-represented among erroneous predictions
Conclusions

- NN RMSE slightly worse than GBM
- Predicted intervals smaller (on average) and tailored
- $\sigma(x)$ gives information on the expected error
- Not able to anticipate/predict a shift to a constant ROCD climb

Further Work

- $(\text{mass}, \text{cas}_1, \text{cas}_2, M)$ modeled as independent $\rightarrow$ model correlations
- Compute trajectory distribution

code + data: https://github.com/richardalligier/atm2019
Thank you, any questions?
[Alligier and Gianazza, 2018]
Learning aircraft operational factors to improve aircraft climb prediction: A large scale multi-airport study.

[Alligier et al., 2015]
Machine learning applied to airspeed prediction during climb.

[Alligier et al., 2014]
Comparison of Two Ground-based Mass Estimation Methods on Real Data (regular paper).

## Input Variables

<table>
<thead>
<tr>
<th>feature description</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>departure and arrival airports</td>
<td>2</td>
</tr>
<tr>
<td>aircraft type variant</td>
<td>1</td>
</tr>
<tr>
<td>airline operator</td>
<td>1</td>
</tr>
<tr>
<td>day of the week</td>
<td>1</td>
</tr>
<tr>
<td>callsign</td>
<td>1</td>
</tr>
<tr>
<td>ICAO 24 bit Mode-S address</td>
<td>1</td>
</tr>
<tr>
<td>distance between airports</td>
<td>1</td>
</tr>
<tr>
<td>temperature at $H_p = 0$</td>
<td>1</td>
</tr>
<tr>
<td>mass estimated on past points and error on past points</td>
<td>2</td>
</tr>
<tr>
<td>track angle at the current point</td>
<td>1</td>
</tr>
<tr>
<td>ground velocity at the current point</td>
<td>1</td>
</tr>
<tr>
<td>north and east wind components</td>
<td>2</td>
</tr>
<tr>
<td>longitude and latitude at the current point</td>
<td>2</td>
</tr>
<tr>
<td>vertical speed at the current and past points</td>
<td>10</td>
</tr>
<tr>
<td>altitude $H_p$ at the current and past points</td>
<td>10</td>
</tr>
<tr>
<td>airspeed $V_a$ at the current and past points</td>
<td>10</td>
</tr>
<tr>
<td>energy variation between the current and past points</td>
<td>9</td>
</tr>
<tr>
<td>temperature from current altitude $H_p$ to $H_p + 11,000$ m</td>
<td>12</td>
</tr>
</tbody>
</table>
Conditional Distribution

bivariate normal density

conditional distribution of $Y$ when $X = x$

source: http://demonstrations.wolfram.com/TheBivariateNormalAndConditionalDistributions
Extract Several Prediction Problems from one Trajectory

Trajectory Sample 1

Trajectory Sample 7

Alligier Predictive Distribution of the Mass and Speed Profile to Improve Aircraft Climb Prediction ATM 2019