Predicting Variable-Length Functional Outputs for Emulation of a NASA Flight Simulator

Yuning He
Advisor: Professor Herbert Lee

University of California Santa Cruz
Problem Background

- Use statistical modeling to improve NASA Intelligent Flight Control System (IFCS) as part of Next Generation Air System

- IFCS: adaptive control using a neural network to changing and sometimes extreme conditions to keep aircraft stable

- Important for Safety: always want control system to adapt correctly to unforeseen events and avoid a crash
Problem Background

- Problem of Interest: Validation & Verification of complex non-linear systems where outputs vary with time.

- The overall complexity and the number of possible states often makes system-level correctness guarantees intractable.

- Use statistical emulation to quantify the uncertainties in models and make reliable predictions of complex phenomena.
Problem Background

- NASA Flight Simulator: software simulation of the airplane and control system instead of working directly with the control software and live airplanes

- Mixed Integer Inputs $\mathbf{x}$: 10 continuous, 1 binary

- High-dimensional output curves: $T = 1901$ time steps

- 12 Output Curves $\mathbf{y}^{(k)} \in \mathbb{R}^T$ – e.g. a pose angle over time

- Wide variety of frequencies within and across output vars

- Neural network: measured—expected $\rightarrow 0 \implies$ oscillations
11 Input Variables in x

▶ Mixed Integer Inputs x: 1 binary, 10 continuous

▶ Histograms of training inputs shown below
Problem Background: Variable-Length Outputs

- NASA Flight Simulator may fail to control the flight for some \( x \) and terminate early

\[ x_1 \rightarrow \]

\[ x_2 \rightarrow \]

- 12 High-dim, Variable-Length output curves: \( T = T(x) \)
The Problem

- Statistical Emulation of the NASA Flight Simulator
  - treat the simulator as a black box and learn the mapping from inputs to outputs.
  - predict both output curve points and output curve length

- Identify success and failure regions of the input space
  - to avoid failure regions, improve IFCS to succeed where failing
- Sensitivity analysis: which input variables are most important
- This talk focuses on single output variable $y(t) = y^{(m)}(t)$
Why Emulate the NASA Flight Simulator for IFCS?

- Emulated success and failure curves are useful to NASA scientists for understanding and improving the IFCS.
- The NASA Flight Simulator takes \( \approx 1.2 \) seconds per output, but still slow to obtain just thousands of outputs.
- Emulator can do joint prediction simultaneously.
  - Once you have fit the emulator, the prediction is fast.
- NASA scientists are interested in failure regions.
  - In 11-dimensional input space, it is difficult to get a dense grid of runs that could be interpolated to give failure regions.
  - Instead, use emulator to predict failure regions.
- Doing sensitivity analysis is easier with an emulator.
General appearance and frequency content of success and failure curves are typically quite different.

Even the failure curves seem to cluster in terms of appearance. Clusters correlate well with failure curve length $T$

$T = 1900, 250, 140, 350$  
$T \approx 250$  
$T = 350..500$
Hypothesis: Using different bases and fitting distinct models for different classes of output curves will improve prediction of output curves. Our hypothesis is proven to be true shown in later slides.

Classes considered:

<table>
<thead>
<tr>
<th>Two Class Problem</th>
<th>Four Class Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>failure 0 &lt; T ≤ 600</td>
<td>failure1 0 &lt; T ≤ 180</td>
</tr>
<tr>
<td>success T = 1900</td>
<td>failure2 180 &lt; T ≤ 280</td>
</tr>
<tr>
<td></td>
<td>failure3 280 &lt; T ≤ 600</td>
</tr>
<tr>
<td></td>
<td>success T = 1900</td>
</tr>
</tbody>
</table>
Use previously described basis model for a given class:

\[ y(x) | \text{class} = \sum_{i=1}^{D} c_{i}^{\text{class}}(x) b_{i}^{\text{class}} + \epsilon \]

Note we allow different bases for different classes

For each class: Fit Treed Gaussian Process (TGP) model to learn the coefficient mappings \( x \rightarrow c_{i}^{\text{class}}(x), \ i = 1, \ldots, D \)
Gaussian Process Models

- A Gaussian Process (GP) is a collection of random variables $Z_1(x_1), \ldots, Z_m(x_m)$ indexed by input $x$ having jointly Gaussian distribution for any finite set of indices.

- The correlation matrix $K$ is the heart of the GP. The correlation $\rho(x, x')$ is based on the distance between $x$ and $x'$, e.g., $\rho(x, x') = \exp \left\{ -\frac{||x-x'||^2}{d} \right\}$

- Treed Gaussian Process (TGP) divides the input space into partitions, and models the mapping $x \rightarrow c_i(x)$ as a stationary GP within each partition.

- TGP gives the flexibility of non-stationary modeling.

- TGP handles mixed integer inputs.
Model class using classification TGP model (CTGP)

CTGP is an extension of TGP that handles categorical outputs. For each GP, introduce latent continuous variable \( \{ Z^r_m \}_{m=1}^M \), for each of the \( M \) possible classes.

Again the correlation within regions of the categorical outputs at \( \mathbf{x} \) and \( \mathbf{x}' \) based on \( ||\mathbf{x} - \mathbf{x}'|| \) is the heart of the model, similar to TGP model shown earlier.

\[
P(\text{class}(\mathbf{x}) = m | \text{region } r) = \frac{\exp(-Z_m(\mathbf{x}))}{\sum_{m'=1}^M \exp(-Z_{m'}(\mathbf{x}))}
\]
Comparing Different Classification Models

- We have tried many models for classification, and CTGP gives the lowest classification error of 11.8%

<table>
<thead>
<tr>
<th>Method</th>
<th>Classification Error</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest Neighbor</td>
<td>43.9%</td>
<td>No Training</td>
</tr>
<tr>
<td>GLM with binomial link</td>
<td>39.7%</td>
<td>few minutes</td>
</tr>
<tr>
<td>TGP_Flight_Length</td>
<td>27.3%</td>
<td>3 hours</td>
</tr>
<tr>
<td>R Tree</td>
<td>26.9%</td>
<td>few minutes</td>
</tr>
<tr>
<td>TGP_01</td>
<td>26.7%</td>
<td>3 hours</td>
</tr>
<tr>
<td>PL Tree</td>
<td>26.7%</td>
<td>3 minutes</td>
</tr>
<tr>
<td>CTGP</td>
<td>11.8%</td>
<td>10+ hours</td>
</tr>
<tr>
<td>SVM</td>
<td>19.4%</td>
<td>&lt; 1 second</td>
</tr>
</tbody>
</table>
Our Hierarchical Prediction Model In Detail - Model Fitting

- Divide training data \( \{(x_k^{tr}, y_k^{tr})\} \) into training data for different classes according to their length \( T \)
- Fit CTGP model for mapping from \( x \rightarrow \text{class}(x) \)
- For each class: Represent class training data output curves in given basis
  - Solve for coefficients

\[
y(x)|\text{class} = \sum_{i=1}^{D} c_i^{\text{class}}(x)b_i^{\text{class}} + \epsilon
\]

- Ridge regression \( y(x)|B, c, \Sigma_y \sim N(Bc, \Sigma_y) \) with prior: \( c \sim N(0, \Sigma_c) \)
- For each class: Fit TGP model to learn the coefficient mappings \( x \rightarrow c_i^{\text{class}}(x), i = 1, \ldots, D \)
Our Hierarchical Prediction Model In Detail - Prediction

- Model Fitting:
  - Fit CTGP model for mapping from $x \rightarrow \text{class}(x)$
  - For each class: Fit TGP model to learn the coefficient mappings $x \rightarrow c_i^{\text{class}}(x)$, $i = 1, \ldots, D$

- Prediction: Given a new input $x^{\text{new}}$
  - Use fitted CTGP model from above to predict $\text{class}(x^{\text{new}})$
  - Use fitted TGP model for $\text{class}(x^{\text{new}})$ to predict coefficients $c_i^{\text{class}}(x^{\text{new}})$
  - Predict output curve as: $y(x^{\text{new}}) = B^{\text{class}} c^{\text{class}}(x^{\text{new}})$
Bases

We considered several different bases in our work, including PCA, Fourier and Wavelet. The key idea is that we represent the functional output curve using a set of orthogonal basis functions. This allows us to deal with the fact that not all curves are of the same length. The bases that provided us with the best result is Fourier Bases.

Fourier Basis: For a given dimension $D = 2K + 1$, it is an orthonormal basis over the interval $[0, 1]$ containing sines and cosines with frequencies $1, \ldots, K$ as well as a constant function. That is, the continuous Fourier basis contains the $2K + 1$ basis functions:

\[
\begin{align*}
    b_0(t) &= 1 & t \in [0, 1] \\
    b_{2f-1}(t) &= \sin(2\pi ft) & f = 1, \ldots, K, \ t \in [0, 1] \\
    b_{2f}(t) &= \cos(2\pi ft) & f = 1, \ldots, K, \ t \in [0, 1].
\end{align*}
\]
In this work, we face a fundamental challenge of different length output curves. For example, for the failure2 class $T_{\text{max}} = 280$. The basis $B \in \mathbb{R}^{T_{\text{max}} \times D}$ for a class will have vectors of length $T_{\text{max}}$ to accommodate the longest curves in the class. How should we determine the coefficients $c \in \mathbb{R}^D$ for a training curve $y$ of length $T < T_{\text{max}}$?

The solution we came up with is to truncate the basis vectors but then place a zero-mean Normal prior on the basis coefficients to shrink the coefficients toward zero. This strategy avoids the need to extend curves to have the same length as the basis vectors, and results in good performance.
Test Configuration

- $D=25$ basis vectors for Fourier bases
- Classification Strategies: 4 class, 2 class, 1 class (all output curves)
- Cross Validation with 867 training runs (507 failures, 360 successes), 100 test runs. Among failures, failure1: 228, failure2: 213, failure3: 66
Results

- First we show results assuming perfect classification to test the proposed statistical model for a single class
- Later we show results using imperfect CTGP classification to test the full statistical model
- Error measure: The error $e_{v,c}$ for output variable $v$ and class $c$ with the set of test output curves $S_{v,c}$, $e_{v,c}$ is standarized given by

$$e_{v,c} = \frac{\sum_{i \in S_{v,c}} ||y_i^{\text{pred}} - y_i||_1}{\sum_{i \in S_{v,c}} T(x_i) \sigma_i}.$$

The true and predicted output curves $y_i$ and $y_i^{\text{pred}}$ are for the given output variable $v$, but we leave out the dependence on $v$ to simplify the error

- Conclusion: 4 class outperforms 2 class, both outperform 1 class
Prediction errors for all output variables using Fourier Series bases and class configuration

<table>
<thead>
<tr>
<th>outvar</th>
<th>failure1</th>
<th>failure2</th>
<th>failure3</th>
<th>success</th>
<th>failure1</th>
<th>failure2</th>
<th>failure3</th>
<th>success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.581</td>
<td>0.554</td>
<td>1.238</td>
<td>0.349</td>
<td>0.840</td>
<td>0.349</td>
<td>0.681</td>
<td>0.306</td>
</tr>
<tr>
<td>2</td>
<td>0.778</td>
<td>0.457</td>
<td>0.841</td>
<td>0.306</td>
<td>0.681</td>
<td>0.306</td>
<td>0.354</td>
<td>0.470</td>
</tr>
<tr>
<td>3</td>
<td>0.193</td>
<td>0.315</td>
<td>0.581</td>
<td>0.470</td>
<td>0.354</td>
<td>0.470</td>
<td>0.640</td>
<td>0.329</td>
</tr>
<tr>
<td>4</td>
<td>0.453</td>
<td>0.340</td>
<td>0.941</td>
<td>0.329</td>
<td>0.640</td>
<td>0.329</td>
<td>0.661</td>
<td>0.493</td>
</tr>
<tr>
<td>5</td>
<td>0.552</td>
<td>0.566</td>
<td>1.171</td>
<td>0.493</td>
<td>0.661</td>
<td>0.493</td>
<td>1.157</td>
<td>0.359</td>
</tr>
<tr>
<td>6</td>
<td>1.102</td>
<td>0.982</td>
<td>0.479</td>
<td>0.359</td>
<td>1.157</td>
<td>0.359</td>
<td>0.162</td>
<td>0.644</td>
</tr>
<tr>
<td>7</td>
<td>0.043</td>
<td>0.039</td>
<td>0.080</td>
<td>0.644</td>
<td>0.162</td>
<td>0.644</td>
<td>0.368</td>
<td>0.113</td>
</tr>
<tr>
<td>8</td>
<td>0.144</td>
<td>0.130</td>
<td>0.649</td>
<td>0.113</td>
<td>0.368</td>
<td>0.113</td>
<td>0.896</td>
<td>0.240</td>
</tr>
<tr>
<td>9</td>
<td>0.722</td>
<td>0.942</td>
<td>1.661</td>
<td>0.240</td>
<td>0.896</td>
<td>0.240</td>
<td>0.709</td>
<td>0.235</td>
</tr>
<tr>
<td>10</td>
<td>0.569</td>
<td>0.615</td>
<td>0.945</td>
<td>0.235</td>
<td>0.709</td>
<td>0.235</td>
<td>0.557</td>
<td>0.157</td>
</tr>
<tr>
<td>11</td>
<td>0.436</td>
<td>0.250</td>
<td>0.891</td>
<td>0.157</td>
<td>0.557</td>
<td>0.157</td>
<td>1.158</td>
<td>0.358</td>
</tr>
<tr>
<td>12</td>
<td>1.109</td>
<td>0.981</td>
<td>0.478</td>
<td>0.358</td>
<td>1.158</td>
<td>0.358</td>
<td>1.158</td>
<td>0.358</td>
</tr>
</tbody>
</table>
Let’s see some predicted curves for different output variables, and different classes (black: true output curve, red: predicted output curve)

More examples (show more figures in here and explain)
Predictions - Alpha: failure1 (left), failure2 (right)
Predictions - Error in roll: success

Run #519: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #568: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #596: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #612: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #86: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #136: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #156: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #201: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360

Run #201: output #1: y_error_1 (black), Predicted (red) [dim=25], nTrain=360
Predictions - Yaw rate: failure2 (left), Yaw rate: success (right)
For curve prediction, randomly split the 967 runs into 867 training examples and 100 test examples.

### 2-class

<table>
<thead>
<tr>
<th></th>
<th>failure</th>
<th>success</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>507</td>
<td>360</td>
<td>867</td>
</tr>
<tr>
<td>test</td>
<td>62</td>
<td>38</td>
<td>100</td>
</tr>
<tr>
<td># errors</td>
<td>3</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>error rate</td>
<td>4.5%</td>
<td>21.1%</td>
<td>11%</td>
</tr>
</tbody>
</table>

### 4-class

<table>
<thead>
<tr>
<th></th>
<th>failure1</th>
<th>failure2</th>
<th>failure3</th>
<th>success</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>228</td>
<td>213</td>
<td>66</td>
<td>360</td>
<td>867</td>
</tr>
<tr>
<td>test</td>
<td>29</td>
<td>28</td>
<td>5</td>
<td>38</td>
<td>100</td>
</tr>
<tr>
<td># errors</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>error rate</td>
<td>17.2%</td>
<td>32.1%</td>
<td>100%</td>
<td>5.3%</td>
<td>21%</td>
</tr>
</tbody>
</table>
Output Curve Prediction using CTGP Classification

- **CTGP usually correct (89% for 2-class, 79% for 4-class)**
  
  ![Correct Predicted Class](image1)
  
  ![Correct Predicted Class](image2)

- **When CTGP incorrect, predicted output curve from wrong model typically quite good near start of flight**

- **As expected, Predicted output curve using correct class usually better than Predicted curve using incorrect class**

  ![Incorrect Predicted Class](image3)
  
  ![Incorrect Predicted Class](image4)

  ![Incorrect Predicted Class](image5)

  ![Incorrect Predicted Class](image6)
Some tests for which predictions are very similar using the correct and incorrect class models

Even some tests in which predicted output curve is slightly better using incorrect class

Visual results and error rates (next) show our curve prediction method has some robustness wrt classification results
Predicted curves when CTGP incorrect are surprisingly good.

Error rates: black = correct class, (blue = incorrect class)

Error rates can only be measured where predictions made.

Need to account for errors in predicted output length $T$

<table>
<thead>
<tr>
<th>outvar</th>
<th>failure2</th>
<th>success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.554 (0.687)</td>
<td>0.349 (0.459)</td>
</tr>
<tr>
<td>2</td>
<td>0.457 (0.476)</td>
<td>0.306 (0.497)</td>
</tr>
<tr>
<td>3</td>
<td>0.315 (0.285)</td>
<td>0.470 (0.577)</td>
</tr>
<tr>
<td>4</td>
<td>0.340 (0.456)</td>
<td>0.329 (0.502)</td>
</tr>
<tr>
<td>5</td>
<td>0.566 (0.559)</td>
<td>0.492 (0.538)</td>
</tr>
<tr>
<td>6</td>
<td>0.982 (0.956)</td>
<td>0.359 (0.617)</td>
</tr>
<tr>
<td>7</td>
<td>0.039 (0.060)</td>
<td>0.644 (0.402)</td>
</tr>
<tr>
<td>8</td>
<td>0.130 (0.162)</td>
<td>0.113 (0.216)</td>
</tr>
<tr>
<td>9</td>
<td>0.942 (0.888)</td>
<td>0.240 (0.286)</td>
</tr>
<tr>
<td>10</td>
<td>0.615 (0.624)</td>
<td>0.235 (0.365)</td>
</tr>
</tbody>
</table>
Future Work

- Simultaneous predictions for multiple output variables
  - avoid modeling correlations between output variables by doing an orthogonal decomposition
- Model for defining the stability boundaries of a multiple input, multiple timeseries output system
  - investigate Bayesian *wombling* technique of Banerjee and Gelfand
- Model sensitivity with mixed-integer inputs

Thank you.