Analysis of Air Traffic Track Data with the AutoBayes Synthesis System

Johann Schumann[1], Karen Cate[2], and Alan Lee[3]

[1] SGT, Inc./ NASA Ames, Moffett Field, CA 94035, Johann.M.Schumann@nasa.gov
[2] NASA Ames, Moffett Field, CA 94035-0001, karen.tung@nasa.gov
[3] NASA Ames, Moffett Field, CA 94035-0001, alan.g.lee@nasa.gov

Abstract. The Next Generation Air Traffic System (NGATS) is aiming to provide substantial computer support for the air traffic controllers. Algorithms for the accurate prediction of aircraft movements are of central importance for such software systems but trajectory prediction has to work reliably in the presence of unknown parameters and uncertainties. We are using the AutoBayes program synthesis system to generate customized data analysis algorithms that process large sets of aircraft radar track data in order to estimate parameters and uncertainties. In this paper, we present, how the tasks of finding structure in track data, estimation of important parameters in climb trajectories, and the detection of continuous descent approaches can be accomplished with compact task-specific AutoBayes specifications. We present an overview of the AutoBayes architecture and describe, how its schema-based approach generates customized analysis algorithms, documented C/C++ code, and detailed mathematical derivations. Results of experiments with actual air traffic control data are discussed.

1 Introduction

Commercial Air Traffic Control (ATC) has coped with the strongly increasing volume of commercial passenger and freight air traffic that causes more and more crowded airways, delays, and canceled flights. The current ATC systems rely on radar data to measure the position of the aircraft; the communication between air traffic controllers and the pilots is handled via voice communication. The Next Generation Air Traffic System (NGATS) is aiming to provide substantial computer support for the air traffic controllers in the form of alert and advisory systems as well pave the way for computerized data links between ATC and the aircraft itself.

The NASA Ames Research Center has developed the software system CTAS (Center Tracon Advisory System) [1,2], which is used as a research and development platform for many projects, e.g., the automatic recognition/prediction of separation violations, or the En-route Descent Advisory system (EDA) [3].

A central software component of the CTAS system is the Trajectory Synthesizer (TS) [4]. Based upon the current state of the aircraft (position, altitude, heading, speed), the current atmospheric data like wind speed and direction,

and the intent (e.g., flight plan), the TS will calculate a prediction of the aircraft's future movements (trajectory). The TS utilizes aerodynamic equations of motion, weather forecasts, aircraft performance models, as well the intent of the current flight segment (e.g., "climb with constant speed to 30,000ft") to generate these predictions. It is obvious that the predictions of the trajectory synthesizer should be as accurate as possible, because many other ATC algorithms use the TS and base their decision upon the predicted trajectories. On the other hand, the accuracy of a trajectory prediction depends on many factors of uncertainty: weather forecast, surveillance data (radar, ADS-B), intent (controller instruction, pilot procedures), pilot inputs (delays in reaction and erroneous pilot actions), navigation, as well as aircraft performance modeling, all of which are often unknown.

In order to obtain realistic estimates for unknown parameters, data mining and analysis can be performed on large data sets of actual aircraft trajectories. All movements of commercial aircraft are registered by land-based radar systems and recorded approximately every 12 seconds. Thus huge data sets become available, even for a restricted region or even a single airport. Such trajectory data can also be used for testing the TS software: how close is the predicted trajectory to the actual one, compared after the fact. Such tests can reveal strength and weaknesses of the TS algorithm. However, there is a catch: almost all of the recorded trajectories comprise nominal flights (e.g., a straight overflight or a standard approach). Thus a high test coverage can only be obtained when an infeasible number of trajectories are exercised. For testing purposes, therefore, data sets with a well-balanced mixture of nominal and off-nominal trajectories are desirable. Extracting such a set from a huge set of actual air traffic data again is a data mining task.

In this paper, we describe, how the AutoBayes program synthesis system can be used for the estimation of unknown parameters and the extraction of a wide variety of flight scenarios. The AutoBayes system [5,6] is a schema-based synthesis system, which takes a high-level statistical model as its input. From that, AutoBayes generates a customized data analysis algorithm and produces C or C++ code, which can be directly invoked from Matlab or Octave to actually process the data. The AutoBayes system is well suited for mining of ATC data, because each different mining task requires a different statistical model, which would require the implementation of specific algorithms or existing code and libraries must be adapted accordingly—usually a very time-consuming task. The ability of AutoBayes to quickly synthesize customized algorithms from compact declarative specifications facilitates a quick exploration of different models.

AutoBayes has been used to support data analysis tasks on sets of actual aircraft trajectories. We will describe AutoBayes models for the unsupervised clustering of track data based upon sets of features. Such structuring of a large set of aircraft movements can be used to isolate specific trajectories of interest (e.g., some unusual flight profile), or to find representative trajectories in large sets of very similar trajectories. We will also report on the use of AutoBayes

models for time series analysis to estimate elements of the flight profile such as the CAS-mach speed transition, as well as for the detection of continuous descent approaches (CDA). For each of the applications, a short, fully declarative statistical specification of the problem was given to the AutoBayes program synthesis system, which in turn generated customized C/C++ algorithms, specifically tailored toward the analysis problem at hand.

There are several approaches in the literature for the automated analysis of air traffic control data using different statistical approaches. For example, [7] uses clustering algorithms to separate out nominal trajectories from those, which do not follow a usual profile and thus could be a potential source for problems. [8] performs statistical time series analysis on air traffic data.

For our analysis, we could have directly used one of the available EM-implementations, like Autoclass [9], EMMIX [10], MCLUST [11], or WEKA [12]. However, such tools are usually designed for Gaussian distributed data only and only allow little customization. Refining the statistical model (e.g., by incorporating other probability distributions for certain variables or to introduce domain knowledge), the analysis algorithms need to be modified substantially for each problem variant, making experimentation a time-consuming and error-prone undertaking with such tools.

The rest of this paper is structured as follows: In Section 2, we give an overview of the AutoBayes program synthesis system and its architecture. Section 3 discusses how AutoBayes has been used to generate clustering algorithms for the analysis of aircraft track data. We discuss the entire AutoBayes specification, describe, how AutoBayes automatically derives solutions for clustering of non-Gaussian distributions and present experimental results. In Section 4, we describe, how change-point time series models, have been used to estimate parameters or detect profiles. Section 5 summarizes our approach and concludes.

2 The AutoBayes Program Synthesis System

AutoBayes [5,6] is a fully automatic program synthesis system that generates efficient and documented C/C++ code from abstract statistical model specifications. Developed at NASA Ames, AutoBayes is implemented in approximately 90,000 lines of documented SWI Prolog (http://www.swi-prolog.org) code. From the outside, AutoBayes looks similar to a compiler for a very high-level programming language (Figure 1): it takes an abstract problem specification in the form of a (Bayesian) statistical model and translates it into executable C/C++ code that can be called from Matlab or Octave.

Once, a specification (for examples see Sections 3 and 4) has been converted into an internal representation by the input parser, a Bayesian network [13], representing the essential information of the specification is created. The synthesis kernel uses a schema-based approach to generate a customized algorithm that solves the statistical task at hand in an imperative intermediate language. This

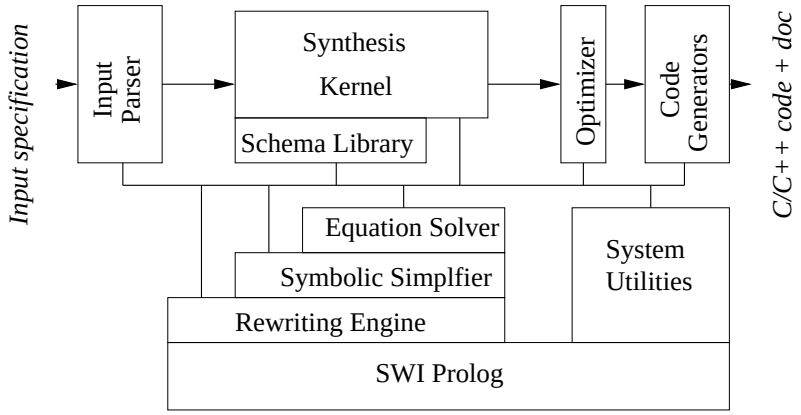

Fig. 1. The AutoBayes system architecture

program is then optimized and the code generator finally produces the desired code (currently C++ for Octave, C for Matlab, or stand-alone C).

AutoBayes uses a schema-based approach, because of the large and complex synthesis tasks, for which pure deductive synthesis (e.g., [14,15,16]) would not scale up. Each schema inside the schema library consists of applicability constraints and a template (parameterized code fragment with holes, which are Prolog variables). When a schema is applicable for the task at hand, AutoBayes tries to instantiate these parameters by direct calculations or by recursive calls to other schemas, which in turn solve the occurring subtasks. Using this mechanism, we not only generate code, but also comments and mathematical derivations, which can be automatically typeset in LaTeX (for an example see the appendix). All schemas are implemented as Prolog clauses; for guiding the search we use Prolog's backtracking mechanism and the constraints in the schemas. The constraints are formulated as conditions over the Bayesian network or as properties about the underlying model.

The schema library is organized in layers: top-level schemas comprise Bayesian network decomposition schemas, which are based upon independence theorems for Bayesian networks [13] as well as other statistical transformations. On the next layer contains schemas with statistical algorithm skeletons, like, for example the EM algorithm (expectation maximization) [17] or k-means for nearest-neighbor clustering. The bottom layer contains calls to symbolic solvers and standard numeric optimization methods, for example a Nelder-Mead simplex algorithm [18].

The schema-based synthesis system heavily relies on a powerful symbolic system. Based upon a small rewriting engine implemented in PROLOG, a large set of symbolic simplification rules, support for symbolic differentiation, as well as a number of symbolic equation solvers has been implemented. For details on the symbolic system see, e.g., [19].

3 Multivariate Clustering of Aircraft Track Data

The aircraft track data set used for this analysis is derived from radar track data. Every 12 seconds, position, altitude, heading, speeds, as well as numerous other data are recorded for each aircraft flying though a specific sector in the air space. Thus, a recording of an air traffic control center for one day usually contains the track data of thousands of aircraft; the data for each aircraft are stored as time series data. Additionally, data about wind speed and wind direction at different altitudes have been provided.

We are interested into automatically finding structure in these large data sets. Obvious categories for flight scenarios are departures and ascends, descents, or high altitude overflights. However, there is a multitude of other categories, which cannot be described as easily. We have been using multivariate clustering with AutoBayes to find such categories. As we did not want to perform a detailed time series analysis, we described each trajectory by a set of features, which were calculated from the track data (see Section 3.2) and used the feature vector as the input for multivariate clustering.

3.1 AutoBayes Clustering Models

Clustering problems can be specified as statistical mixture problems. Here, the statistical distribution or probability density function (PDF) for each feature must be known. In our case, several variables have discrete values (e.g., booleans or enumeration types for the different aircraft models), which have a discrete distribution; continuous variables include distances, times, and angles. Most clustering algorithms and tools make the assumption that all (continuous and discrete) variables are Gaussian (normal) distributed. For variables, which have a different PDF, Gaussian noise is added to the data in order to obtain a Gaussian-like distribution. However, such a model can be statistically very inaccurate, in particular, when dealing with angles. A noisy measurement of an angle close to 0° would, when considered normal distributed, yield (incorrectly) two classes with means around 0° and 360°. If arbitrary values for angles can occur, an angular rotation does not help either.

With AutoBayes, the user can directly specify customized mixture models, which correctly deal with variables having different probability distributions. Listing 1.1 shows a complete AutoBayes specification for such a clustering task. For space reasons, we focus on just three variables: heading (an angle), altitude (Gaussian distributed), and the type of the aircraft, which has discrete values. The actual string values (e.g., "B747") are converted into an enumeration type during preprocessing.

The first lines of Listing 1.1 define the model name and the global declarations for this problem. The constants N and C are instantiated when the generated code is executed, but they are assumed to be constant within the algorithm. Constraints are given after the **where** keyword. Lines 6–12 declare all distribution parameters for the statistical variables, θ_0, m for the von Mises distribution for the heading hd, μ, and σ for the altitude and ρ for the aircraft

type (e.g., $\rho_0 = \mathbf{P}(x = "B777")$). Line 12 states that each data point actually belongs to one of the specified types of aircraft. Since different classes will have different parameters, these variables are vectors over the classes. Note that all indexing is 0-based like in C. Φ is the unknown class frequency (Line 14), and c is the most likely class assignment for each aircraft track, which will be calculated and returned by the algorithm (Line 15). Statistically speaking, c is discrete distributed with probabilities Φ as reflected Line 17. Line 15 states that Φ is indeed a probability vector and has to add up to 1.

```
1  model ac_track as 'AC TRACK ANALYSIS'.
2
3  const nat N as 'NR. OF DATA POINTS'. where 0 < N.
4  const nat C as 'NR. OF CLASSES'. where 0 < C. where C << N.
5
6  double theta0_hd(0..C-1).    % THETA_0 FOR ANGLE
7  double m_hd(0..C-1).         % M FOR ANGLE
8  double mu_alt(0..C-1).       % MEAN, SIGMA FOR ALT
9  double sigma_alt(0..C-1). where 0 < sigma_alt(_).
10      % AC_TYPE: {"B777", "B737", "B755", "OTHER"}
11 double rho_AC_type(0..3, 0..C-1).
12     where 0 = sum(_i := 0..3, rho_AC_type(_i,_))-1.
13
14 double phi(0..C-1) as 'CLASS FREQUENCY'.
15          where 0 = sum(_i := 0 .. C-1, phi(_i))-1.
16 output double c(0..N-1) as 'CLASS ASSIGNMENT VECTOR'.
17     c(_) ~ discrete(vector(_i := 0 .. C-1, phi(_i))).
18
19 data double hd(0..N-1).
20     hd(_i) ~ vonmises1(theta0_hd(c(_i)),m_hd(c(_i))).
21 data double alt(0..N-1).
22     alt(_i) ~ gauss(mu_alt(c(_i)), sigma_alt(c(_i))).
23 data nat AC_type(0..N-1).
24     AC_type(_i) ~ discrete(vector(_j := 0..3,
25                                   rho_AC_type(_j,c(_i)))).
26
27 max pr({hd, alt, AC_type} |
28    {phi,theta0_hd,m_hd,mu_alt,sigma_alt,rho_AC_type})
29  for {phi,theta0_hd,m_hd,mu_alt,sigma_alt,rho_AC_type}.
```

Listing 1.1. AUTOBAYES specification for aircraft track clustering analysis

Lines 19–25 declare the data vectors and their respective distributions using the class-indexed parameters defined above. Finally, the goal statement (Lines 27–29) triggers the AUTOBAYES synthesis: maximize the probability of the data, given the distribution parameters for all distribution parameters. The solution to this task is a set of estimated parameters $(\phi, \theta_0, m, \mu, \sigma, \rho)$, which most likely explain the data. Note that the entire specification is purely declarative.

From this specification, AutoBayes generates 1032 lines of documented C code with a Matlab Mex interface in less than 5 seconds on a Mac book. In a first step, AutoBayes generates a Bayesian network, describing the specification is generated. Figure 2 shows the autogenerated graph. Shaded nodes are known variables (declared as **data**). Boxes around groups of nodes indicate that these variables are identically indexed (iid) over the number of classes C or the number of tracks N. As usual, directed arcs show the interdependency between the nodes.

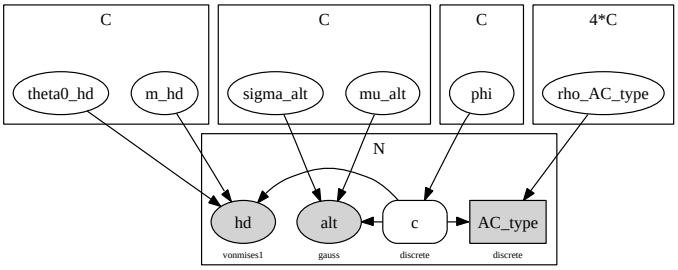

Fig. 2. Autogenerated Bayesian network for clustering specification (Listing 1.1)

Then, schemas are invoked, which detect that the problem can be solved using an iterative clustering algorithm like expectation maximisation (EM, [17]), or k-means [20]. By default, EM is selected. This algorithm schema breaks down the model and generates new subproblems, which must be solved by the recursive application of schemas. The code, which is produced by the schemas is used to assemble all parts of the EM algorithm. For our example, 4 different subproblems are produced: $\max \mathbf{P}(c \mid \phi)$ for ϕ, $\max \mathbf{P}(AC\_type \mid c, \rho_{AC})$ for ρ_{AC}, $\max \mathbf{P}(alt \mid c, \mu_{alt}, \sigma_{alt})$ for μ_{alt}, σ_{alt}, and $\max \mathbf{P}(hd \mid c, \mu_{hd}, \theta^0_{hd})$ for μ_{hd}, θ^0_{hd}.

The first subproblem can be solved symbolically using a Lagrange multiplier (for details see [5]). Each of the remaining subproblems can be solved independently. The optimization of the probability terms first triggers schemas, which replace the conditional probability by products and then attempt to optimize the atomic probability expressions. This is done in the text-book way by setting the partial derivatives to zero and solving for the unknown parameters. The appendix shows the detailed (autogenerated) derivation for the angle variable hd. For a detailed derivation for the Gaussian distributed variable see [5]; for the discrete variable see [6].

After all symbolic derivations have been performed, the top-level schema pulls together the individual pieces of the algorithm and, after several optimization and transformation steps, C code is generated. Please note that all schemas are independent of the specification and are not specifically tailored toward a given probability density. For each PDF, only symbolic expressions for the classical defining properties have to be provided. They can be taken directly out of a textbook.

3.2 Experiments and Results

For our clustering experiments, we used data sets containing flights around one large airport (DFW) over a period of between one and seven days. These data sets contained between a few hundred to more than ten thousand trajectories. In a first step, we extracted features from each aircraft track, so that each aircraft track was represented by a vector of features. Features included beginning or end coordinates for a track, overall changes in altitude, changes in headings, and much more. The features were kept as generic as possible but were selected to describe meaningful trajectories. For example, a low value of overall altitude changes (combined with a high altitude) can indicate a high altitude overflight. A large amount of altitude change can, when combined with other features (e.g., location of the trajectory start or end), describe take-off or landing scenarios.

A large number of customized features can be defined. They can include the straight-forward combination of information extracted from the time series, but can also include information extracted through additional preprocessing. For example, specific kinds of trajectories can (e.g., climb, holding, landing, overflight). Also distances from reference trajectories or lists of way-points can be used. The set of selected features spans a multi-dimensional hybrid space, as different features can have different probability distributions. Then, a task-specific AutoBayes model is set up and customized code is generated.

Figure 3 shows results that have been obtained using simple sets of features. In the first two panels, features included altitude, mean calibrated air speed (CAS), as well as the overall change in altitude. Trajectories belonging to different classes are shown in shades of grey. The two selected classes show low-speed approaches close to the airport (black) as compared to high altitude passages shown in grey. Panel (B) show climb approaches (grey), which characterized by increasing altitude and constant or increasing speed. The main directions of the departing aircraft, depending on the orientation of the runways can be seen clearly. Panels (C) and (D) shows members of a class of trajectories, which include many turns. Typically, navigation among way-points and holding patterns belong to this group (C). Panel (D) shows one selected trajectory, which was put into a separate class due to its many changes in heading. That flight most probably was a surveillance aircraft or a helicopter. The dots represent the individual radar measurements.

4 Change-point Detection Models

4.1 The CAS/mach Transition

Most climb procedures for commercial aircraft utilize a climb Calibrated Air Speed (CAS) and a climb mach speed. At low altitudes, the aircraft flies with the constant climb CAS measured in knots. As the aircraft ascends it reaches a particular altitude where the climb CAS is equivalent to the climb mach. This is the CAS-mach transition altitude. At this point the speed to be maintained

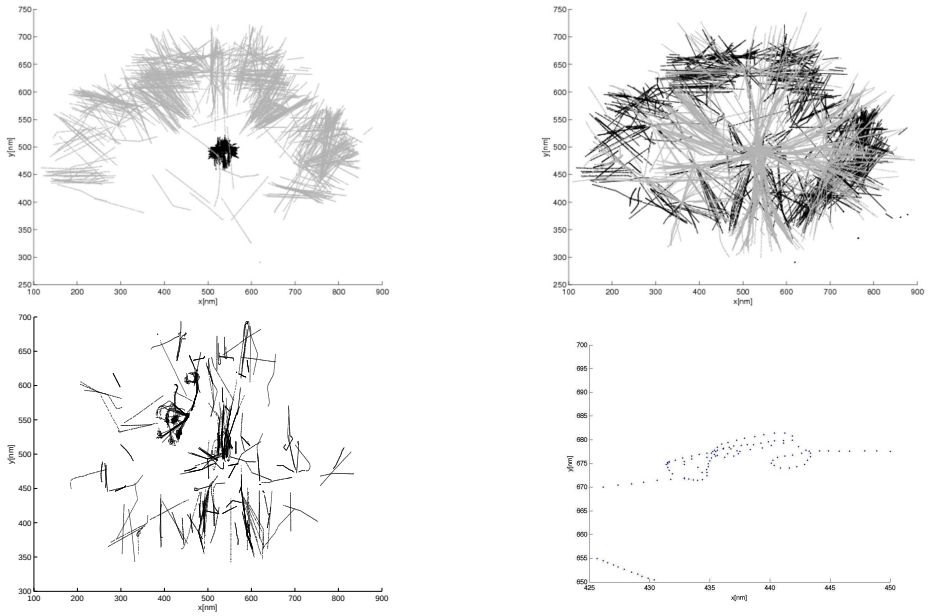

Fig. 3. Clustering results of aircraft track data

transitions from the climb CAS to the climb mach. Once the aircraft has reached its final altitude, it will continue to measure its speed in mach.

Such climb profiles are usually executed by the flight management system (FMS). The pilot has to enter the final altitude and speed (in mach) and the FMS will execute the appropriate profile. The altitude, at which the FMS performs the transition is usually kept constant. However, this parameter is not published and can vary between different air carriers. For an accurate prediction of trajectories as done by the CTAS TS software, however, good knowledge about these transition points is important.

4.2 Change-point Detection with AutoBayes

The calibrated air speed v_{CAS} and the mach number v_{mach} for such a climb scenario can be written as

$$v_{CAS} = \begin{cases} v^0_{CAS} & t \leq t_T \\ v^0_{CAS} - \Delta_C * (t - t_T) & t > t_T \end{cases}$$

$$v_{mach} = \begin{cases} v^0_{mach} + \Delta_m * (t_T - t) & t \leq t_T \\ v^0_{mach} & t > t_T \end{cases}$$

where t_T is the time when the flight regime changes (change-point). v^0_{mach}, v^0_{CAS} denote the constant mach number and CAS speed, respectively, and Δ_C, Δ_m

describe the slope of the speed changes. Prior to the transition, the CAS is kept constant, whereas the mach number increases linearly. This is due to the decreasing air pressure at high altitudes, which lowers the speed of sound. After this transition point, the mach number is kept constant for the remainder of the climb. Again, due to decreasing air pressure, the CAS is now decreasing.

Of course, all data are highly noisy, because turbulence, radar inaccuracies, and pilot maneuvers can perturb the data. Therefore, a statistical approach for finding this choice-point is needed. In our case, we simply assume that all measurements are Gaussian distributed, for example, $v_{CAS} = N(\mu_{vCAS}, \sigma_{vCAS})$. Then the problem of finding the best change point can be easily formulated as a statistical problem of change detection (e.g., [21,22]). Statistical text books usually provide examples and algorithms for just one variable. So, a manual development of the change-point detection algorithm and its implementation would be non-trivial and time-consuming.

In AutoBayes, we directly can specify our problem. Listing 1.2 contains the entire AutoBayes specification for this problem; AutoBayes then generates 684 lines of commented C code within a few seconds.

```
1  model climb_transition.
2  const nat n.  where 0 < n.
3  nat t_T as 'Transition time'. where t_T in 3..n-3.
4  double v_mach_0. double d_mach.
5  double v_CAS_0. double d_CAS.
6  const double sigma_sq.  where 0 < sigma_sq.
7  data double cas(0..n-1) as 'time series data: air speed'.
8  data double mach(0..n-1) as 'time series data: mach'.
9
10 cas(T) ~  gauss( cond(T < t_T,
11                       v_CAS_0,
12                       v_CAS_0 - (I-t_T)*d_CAS
13                  ), sqrt(sigma_sq)).
14 mach(T) ~ gauss( cond(T < t_T,
15                       v_mach_0 - (T-t_T)*d_mach,
16                       v_mach_0
17                  ), sqrt(sigma_sq)).
18 max pr({mach, cas}|{d_mach, v_mach_0, v_CAS_0, d_CAS, t_T})
19          for {v_mach_0, d_mach, v_CAS_0, d_CAS, t_T}.
```

Listing 1.2. AutoBayes specification for CAS-mach transition

Lines 1–6 declare model name and all constants and unknown parameters. Here, we assume that the uncertainty σ^2 is known, but all other parameters must be estimated. Lines 8–9 declare the known data vectors, which are time-series data here. Lines 10–17 are the core of the specification, where the probability distributions of each of the statistical variables are declared. This formulation is directly derived from the formulas shown above and uses C-style conditional expressions (**cond**(E,T,F) corresponds to E?T:F). Finally, lines 18–19 contains the goal statement.

Figure 4 shows results for three different climb scenarios: the mach number (top panel), the calibrated air speed (CAS, middle panel), and the altitude over time. The thin curves show actual aircraft data. The lines indicate the linear approximations. The location of the change point and the other unknown parameters have been estimated closely.

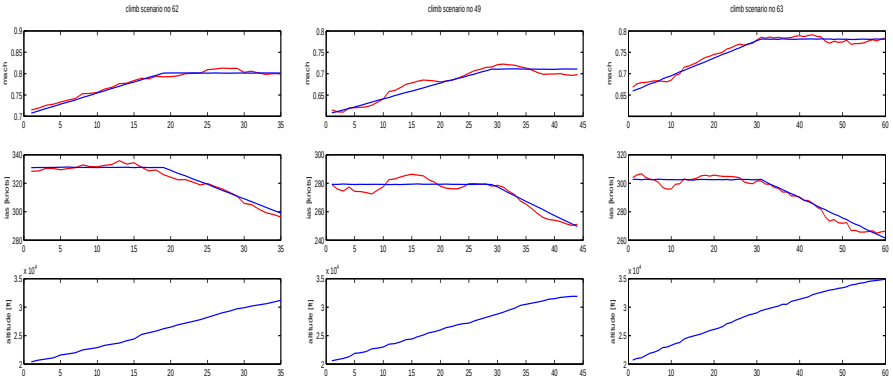

Fig. 4. Results of analysis with AutoBayes for three climb scenarios.

In a further analysis step, we analyzed more than 10,000 climb scenarios and recorded the altitude and speed at which the estimated CAS-mach transition took place. Figure 5 (left) shows the likelihood of the transition in an altitude over mach coordinate system as a set of contour lines (darker areas indicate larger likelihood). Two major transition altitudes could be detected, one at approximately 26,000ft, the other at around 31,000ft.

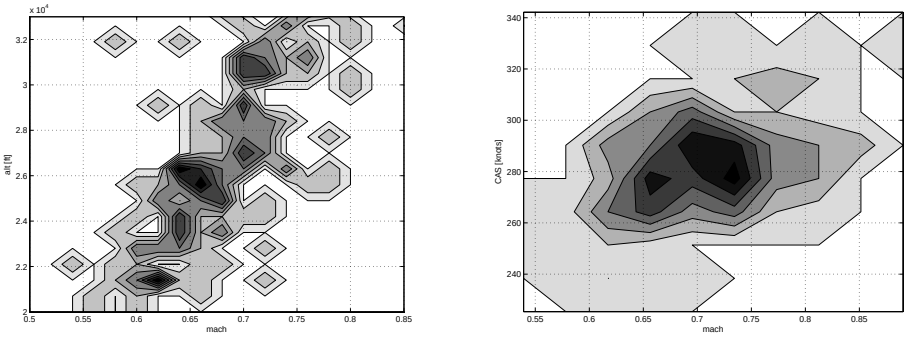

Fig. 5. Likelihood of transition point in an altitude over mach coordinate system (left) and in a CAS over mach coordinate system (right).

Figure 5 (right) shows the results of the analysis displayed as the likelihood of the transition in a CAS over mach coordinate system. This figure indicates that in most trajectories the climbs are initially performed with a CAS of 275 knots. Then the climb is continued most likely with mach 0.78 or mach 0.7. These two panels only show data for the B737 aircraft. Results for other types of aircraft show a similar behavior, but with different parameters.

4.3 CDA Detection

A continuous descent approach (CDA) is a flight profile where the aircraft continuously descends without any level-off segments. The benefits of this approach over a standard approach are a reduction in noise, emissions, fuel consumption, and flight time. For the detection of CDA-like scenarios, another AutoBayes change-point model has been used. One of the main characteristics of a CDA descent is a long stretch of almost constant vertical speed. In the data we analyzed [23], that speed was around -2,000ft/min. Figure 6 shows the horizontal profile in x-y coordinates and the altitude during the descent. Note that the long linear stretch during the descent (from appr. 35,000ft to appr. 12,000ft) indicates a constant descent speed.

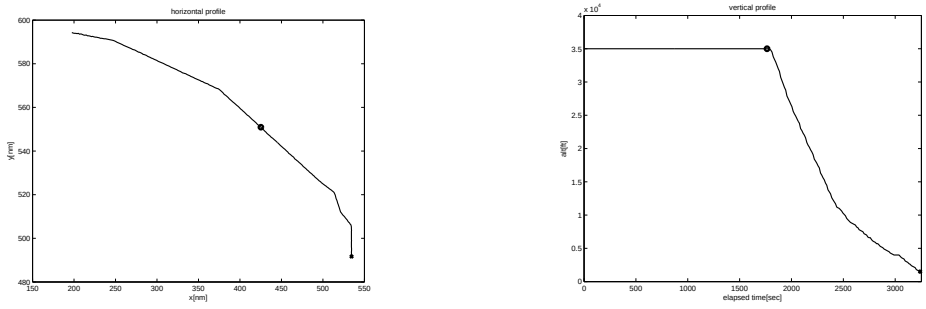

Fig. 6. Horizontal and vertical profile of a CDA approach. Start of descent and meter fix at 4,000ft are marked by a dot.

The AutoBayes specification of a CDA model entirely focuses on the time-series representing the vertical speed v_{vert} and has three distinct phases. In Phase 1, the descent has not yet started. The values of v_{vert} are around zero, but very indeterminate. Phase 2 comprises the CDA phase. Here, we assume that the vertical speed is Gaussian distributed around $\bar{v}_{cda}$ (in our case -2,000ft/min) with a reasonably σ^2 to accommodate for the noise. In Phase 3, again, only very vague assumptions about the vertical speed can be made. The time-points t_{12} and t_{23} represent the transitions. Formally, we have

$$v_{vert} = \begin{cases} N(\bar{v}, \sigma^2_{high}) & \text{for } t < t_{12} \\ N(\bar{v}_{cda}, \sigma^2_{low}) & \text{for } t_{12} \leq t < t_{23} \\ N(\bar{v}, \sigma^2_{high}) & \text{for } t_{23} \leq t \end{cases}$$

Figure 6 shows the vertical speed of the aircraft for one CDA approach. The different phases of the descent are marked. The thick horizontal lines show mean speeds for each segment ($\bar{v}, \bar{v}_{cda}$); the dashed lines show $\pm\sigma^2$ values around the means. Note that the actual vertical speed stays within these boundaries. It is obvious that this model is very primitive, yet it can capture the major vertical speed characteristics of a CDA approach. Listing 1.3 shows an excerpt of the AutoBayes specification containing the distribution of the time-series data and the goal statement. The rest of the specification is very similar to Listing 1.2.

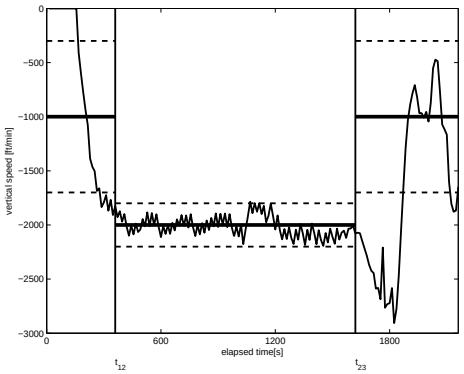

Fig. 7. Vertical speed over time for a CDA scenario.

```
1 v_vert(I) ~ gauss( cond(I < t_12  , v_bar,
2                         cond((I ≥ t_12) and (I< t_23),
3                            v_bar_cda, v_bar)),
4                    cond(I < t_12  , sigma_sq_high,
5                         cond((I ≥ t_12) and (I< t_23),
6                            sigma_sq_low, sigma_sq_high))).
7 max pr({v_vert}|{t_12, t_23}) for {t_12, t_23}.
```

Listing 1.3. AutoBayes specification for detection of CDA (excerpts)

In our analysis, we tested descent scenarios over the period of one week to find out scenarios, which were likely flown as actual CDA approaches. In a preprocessing phase, we eliminated all tracks that had longer horizontal stretches flown at low altitude. Then, we used our AutoBayes model to find the longest stretches with constant vertical speed. The length of these stretches related to the overall duration of the descent was taken as metric on how CDA-like an approach was. In the given data set, we were able to identify a number of likely CDA approaches. Many of them involved cargo or business jets and most were flown between during the night. Since standard procedures do not offer CDA approaches, this finding is consistent with the fact that approach procedures are handled more flexible during times with low traffic density.

5 Conclusions

In this paper, we discussed how the program synthesis tool AutoBayes can be used for the statistical analysis of aircraft track data. We presented experiments on multivariate clustering on trajectories, which have been represented as vectors of features, and on change-point models to estimate unknown parameters like the CAS-mach transition and the detection of constant descent approaches (CDA). For each of these tasks, we presented the AutoBayes specification, which comprises a compact, fully declarative statistical model. From that, AutoBayes generates a customized algorithm implemented in C or C++ within a few seconds run time. The schema-based synthesis approach used in AutoBayes together with its powerful symbolic system with enable the system to fully automatically synthesize complicated, but highly documented code of up to several thousand lines of C/C++.

The schema library of AutoBayes can be extended to handle different kinds of probability density functions and to incorporate different algorithm skeletons [24]. However, such extensions usually require a substantial understanding of the internal workings of AutoBayes and detailed knowledge of Prolog.

By using AutoBayes to generate customized algorithms for each individual task, a lot of otherwise necessary implementation effort can be saved, in particular as AutoBayes is capable of handling statistical models with many non-Gaussian distributions as well as incorporating domain knowledge and additional constraints in the form of (Bayesian) priors. With the generated code interfacing to Octave and Matlab, AutoBayes thus can be used to quickly and effectively explore a variety of statistical models for advanced data analysis tasks as are necessary for the analysis of Air Traffic data.

References

1. Erzberger, H.: CTAS: Computer intelligence for air traffic control in the terminal area. Technical Report NASA TM-103959, NASA (1992)
2. Denery, D.G., Erzberger, H.: The center-tracon automation system: Simulation and field testing. In: Proceedings of the Advanced Workshop on ATM (ATM 95) sponsored by the National Research Council of Italy, NASA TM-110366 (1995)
3. Coppenberger, R.A., Lanier, R., Sweet, D., Dorsky, S.: Design and development of the en route descent advisor (EDA) for conflict-free arrival metering. In: AIAA Guidance, Navigation, and Control Conference. AIAA-2004-4875. (2004)
4. Slattery, R., Zhao, Y.: Trajectory synthesis for air traffic automation. Journal of Guidance, Control and Dynamics **20** (1997) 232–238
5. Fischer, B., Schumann, J.: AutoBayes: A system for generating data analysis programs from statistical models. J. Functional Programming **13** (2003) 483–508
6. Schumann, J., Jafari, H., Pressburger, T., Denney, E., Buntine, W., Fischer, B.: AutoBayes program synthesis system users manual. Technical Report NASA/TM-2008-215366, NASA (2008)
7. Gariel, M., Srivastava, A., Feron, E.: Trajectory clustering and an application to airspace monitoring. Conf. on Intelligent Data Understanding (CIDU). (2009)

8. Meyerhoff, N., Wang, G.: Statistical analysis and time series modeling of air traffic operations data from flight service stations and terminal radar approach control facilities: Two case studies. Technical Report ADA109873, DTIC (1981)
9. Cheeseman, P., Stutz, J.: Bayesian classification (AutoClass): Theory and results. In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press (1996) 153–180
10. McLachlan, G., Peel, D., Basford, K.E., Adams, P.: The EMMIX software for the fitting of mixtures of normal and t-components. J. Statistical Software **4** (1999)
11. Fraley, C., Raftery, A.E.: MCLUST: Software for model-based clustering, density estimation, and discriminant analysis. Technical Report 415, Department of Statistics, University of Washington (2002)
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explorations **11** (2009)
13. Buntine, W.L.: Operations for learning with graphical models. J. AI Research **2** (1994) 159–225
14. Whittle, J., Van Baalen, J., Schumann, J., Robinson, P., Pressburger, T., Penix, J., Oh, P., Lowry, M., Brat, G.: Amphion/NAV: Deductive synthesis of state estimation software. In: Proc. 16th Intl. Conf. Automated Software Engineering, IEEE Comp. Soc. Press (2001) 395–399
15. Ayari, A., Basin, D.: A higher-order interpretation of deductive tableau. J. Symbolic Computation **31** (2001) 487–520
16. Manna, Z., Waldinger, R.J.: Fundamentals of deductive program synthesis. IEEE Trans. Software Engineering **18** (1992) 674–704
17. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm (with discussion). J. of the Royal Statistical Society series B **39** (1977) 1–38
18. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. 2nd. edn. Cambridge University Press (1992)
19. Fischer, B., Hajian, A., Knuth, K., Schumann, J.: Automatic derivation of statistical data analysis algorithms: Planetary nebulae and beyond. In: Proc. 23rd Intl. Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, American Institute of Physics (2003) 276–291
20. Hartigan, J.: Clustering Algorithms. Wiley (1975)
21. Brodsky, B.E., Darkhovsky, B.S.: Nonparametric Methods in Change-Point Problems. Kluwer (1993)
22. Basseville, M., Nikiforov, I.V.: Detection of Abrupt Changes: Theory and Application. Prentice-Hall (1993)
23. Roach, K.: Analysis of American Airlines continuous descent approaches at Dellas/Fort Worth airport. Technical Report NTX_20080522_CDA_Meeting, North Texas Research Station (NTX) (2008)
24. Gray, A.G., Fischer, B., Schumann, J., Buntine, W.: Automatic derivation of statistical algorithms: The EM family and beyond. In: Advances in Neural Information Processing Systems 15, MIT Press (2003) 689–696
25. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer (2006)

Appendix

This appendix shows how AutoBayes solves $\max \mathbf{P}(hd \mid c, \mu_{hd}, \theta^0_{hd})$ for μ_{hd}, θ^0_{hd}, from Listing 1.1. AutoBayes automatically generated the detailed derivation typeset in LaTeX. We only added a few line-breaks and abbreviated theta0_hd by θ and m_hd by m. The solution uses a function *solve_B* to solve the Bessel function. As no closed-form solution exists, we chose to approximate this function by $solve\_B(x) \approx \tan(0.5\pi x)$ (see [25]). If necessary, an AutoBayes schema could be developed to instantiate an iterative numerical algorithm.

The conditional probability $\mathbf{P}(hd \mid c, m, \theta)$ is under the dependencies given in the model equivalent to

$$\prod_{i=0}^{-1+N} \mathbf{P}(hd_i | c_i, m, \theta)$$

The probability occurring here is atomic and can thus be replaced by the respective probability density function given in the model. Summing out the expected variable c_{pv16} yields the log-likelihood function

$$\sum\nolimits_{j \,\in\, \mathrm{dom}\ c_i \sim q(i,j)}^{i=0\ldots-1+N} \log \prod_{k=0}^{-1+N} \exp\left(cos(hd_k - \theta_{c_k})\, m_{c_k}\right) \frac{1}{2\,\pi\, B_i(0, m_{c_k})}$$

which can be simplified to

$$(-1\ N\ \log 2) + (-1\ N\ \log \pi) + (-1 \sum_{i=0}^{-1+C} \log B_i(0, m_i) \sum_{j=0}^{-1+N} q(j,i)) + \sum_{i=0}^{-1+C} m_i \sum_{j=0}^{-1+N} cos((-1\,\theta_i) + hd_j)\, q(j,i)$$

This function is then optimized w.r.t. the goal variables m and θ.

...

The function

$$(-1\ \log B\_i(0, m_l) \sum_{i=0}^{-1+N} q(i,l)) + (m_l \sum_{i=0}^{-1+N} cos((-1\,\theta_l) + hd_i)\, q(i,l))$$

is then symbolically maximized w.r.t. the goal variables m_l and θ_l. The partial differentials

$$\frac{\partial f}{\partial m_l} = (-1\, B\_i(0, m_l)^{-1}\, B\_i(1, m_l) \sum_{i=0}^{-1+N} q(i,l)) + \sum_{i=0}^{-1+N} cos((-1\,\theta_l) + hd_i)\, q(i,l)$$

$$\frac{\partial f}{\partial \theta_l} = (-1\, m_l\, sin(\theta_l) \sum_{i=0}^{-1+N} cos(hd_i)\, q(i,l)) + (cos(\theta_l)\, m_l \sum_{i=0}^{-1+N} sin(hd_i)\, q(i,l))$$

are set to zero; these equations yield the solutions

$$m_l = solve\_B(\sum_{i=0}^{-1+N} q(i,l)^{-1} \sum_{i=0}^{-1+N} cos((-1\,\theta_l) + hd_i)\, q(i,l))$$

$$\theta_l = atan2(\sum_{i=0}^{-1+N} sin(hd_i)\, q(i,l),\ \sum_{i=0}^{-1+N} cos(hd_i)\, q(i,l))$$