Automatic Synthesis of Statistical Data Analysis Programs

— Position Statement —

Bernd Fischer
RIACS/NASA Ames Research Center
fisch@email.arc.nasa.gov

Statistical data analysis is a core activity in experimental sciences. It encompasses a wide variety of tasks, ranging from, e.g., a simple linear regression to fitting complex dynamical models. Developing statistical data analysis programs, however, is an arduous and error-prone process which requires profound expertise in different areas: statistics, numerics, software engineering, and of course the scientific application domain.

We believe that statistical data analysis is a very promising domain for the application program synthesis, despite—or even because—these difficulties. Statistics provides a unifying and concise domain-specific notation. Graphical models (Buntine 1994) provide a structuring mechanism which can be exploited during the synthesis process, e.g., to decompose a problem into independent subproblems. Statistical algorithms like EM (Dempster, Laird, & Rubin 1977) are often applicable to a wide range of problems; their generic formulations allow a "plug'n'play"-style algorithm combination. Recently developed sophisticated data structures as for example kd-trees (Pelleg & Moore 1999) offer orders-of-magnitude speed-up for certain problems but are rarely employed due to the increased programming complexity they cause. Finally, data analysis is characterized by an iterative development style: an initial model is hypothesized, implemented, evaluated on real data, and—if necessary—refined. However, each iteration typically involves substantial programming efforts as prototyping is often not sufficient to work with real data sets and even small model modifications may require radically dfferent algorithms. Program synthesis encapsulates many of the statistical, numerical, and software engineering aspects of each iteration and thus allows users to concentrate on their scientific application. Its fast turn-around times make model refinement and design-space exploration feasible.

We are currently developing AUTOBAYES, a program synthesis system for the generation of data analysis programs from statistical models. A statistical model specifies the properties for each problem variable (i.e., observation or parameter) and its dependencies in the form of a probability distribution. Figure 1 shows the model for a random walk (i.e., a simple noisy dynamical process) in AUTOBAYES's input notation. The last two lines are the core of the specification; the remaining lines just declare the model variables

```
model walk as 'Random Walk'.

const nat n_points.
  where 0 < n_points.

double rate  as 'drift rate / time slice'.
double error as 'drift error / time slice'.
  where 0 < error.

data double x(0..n_points-1).
x(I) ~ gauss(cond(I>0,x(I-1),0)+rate, error).

max pr(x|rate, error) for rate, error.
```

Figure 1: AUTOBAYES specification of random walk

and impose some additional constraints on them. The distribution statement

```
x(I) ~ gauss(cond(I>0,x(I-1),0)+rate, error)
```

characterizes the observed data x: each observation x_i is normal distributed (or Gaussian) with a gradually changing mean and constant but unknown error. Its mean value is expected to depend on the previous observation x_{i-1} and an unknown drift rate. Over time, the data points thus drift away from their origin, which is here modeled as zero. The optimization statement

```
max pr(x|{rate, error}) for {rate, error}
```

characterizes the data analysis task: find the values for the unknown *rate* and *error* parameters which explain best the known data x, i.e., maximize the data probability under the parameter values and model distributions.

From such models, AUTOBAYES generates optimized and thoroughly documented C/C++ code which can be linked dynamically into the Matlab and Octave environments. AUTOBAYES follows a schema-based synthesis approach rather than the direct proof-as-programs-based approach. However, schemas can equivalently be understood as theorems of the domain theory or as generic algorithms, and AUTOBAYES incorporates schemas with a distinctive "theorem flavor" as well as such with more of an "algorithm flavor." The schemas are organized into four different hierarchical layers. The top-most layer contains the most theorem-like schemas, which are direct formalizations of different decomposition theorems for graphical models. These schemas

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

contain almost no code skeletons. The subsequent layers then become more and more algorithm-like as their code skeletons grow more and more detailed. These layers contain formula decompostion schemas (e.g., an index decomposition for independently and identically distributed random variables), proper statistical algorithm schemas (e.g., EM), and numerical optimization schemas (e.g., the simplex method), respectively. This layering is not only conceptual but is also used as a heuristic to control the synthesis process. Each schema has a number of enabling conditions which are checked against the current statistical model; each schema application can modify the model and thus trigger the application of other schemas. This mechanism allows AUTOBAYES to generate code as a composition of different schemas, thus "re-inventing" data-analysis algorithms from simple building-blocks. AUTOBAYES augments the schema-guided approach by symbolic-algebraic computation and can thus derive closed-form solutions for many problems and sub-problems. More details on AUTOBAYES and the incorporated schemas can be found in (Fischer, Schumann, & Pressburger 2000) and (Fischer & Schumann 2001).

We have applied AUTOBAYES to a large number of textbook examples, machine learning benchmarks, and NASA applications. These encompass such diverse tasks as for example clustering of rock sample spectra, changepoint detection in γ-ray bursts, and software reliability estimation. Synthesis times were generally in the sub-minute range (on a 360MHz Sun workstation); small specifications are typically solved in a few seconds—the synthesis time for the random walk example from Figure 1 is 1.8 seconds. The generated programs were as long as 2000 lines of commented code. This yields leverage factors from specifcation to code between 1:10 and 1:30, depending on the existence (and AUTOBAYES's ability to find) closed-form solutions. For the random walk example, AUTOBAYES can find the existing closed-form solution; the generated 120 lines C-program thus consist mainly of boilerplate code and comments.

Acknowledgments

AUTOBAYES is a joint development effort with Wray Buntine and Johann Schumann.

References

W. L. Buntine 1994. "Operations for learning with graphical models". *JAIR*, **2**:159–225, 1994.

A. P. Dempster, N. M. Laird, and D. B. Rubin 1977. "Maximum likelihood from incomplete data via the EM algorithm (with discussion)". *J. of the Royal Statistical Society series B*, **39**:1–38, 1977.

B. Fischer, J. Schumann, and T. Pressburger 2000. "Generating Data Analysis Programs from Statistical Models (Position Paper)". In W. Taha, (ed.), *Proc. Intl. Workshop Semantics Applications, and Implementation of Program Generation*, *LNCS* **1924**, pp. 212–229, Montreal, Canada, September 2000. Springer.

B. Fischer and J. Schumann 2000. AutoBayes: A System for Generating Data Analysis Programs from Statistical Models, 2001. Submitted for publication. Available at `http://ase.arc.nasa.gov/people/fischer`.

D. Pelleg and A. Moore 1999. "Accelerating Exact k-means Algorithms with Geometric Reasoning". In S. Chaudhuri and D. Madigan, (eds.), *Proc. 5th KDD*, pp. 277–281, San Diego, CA, August 15–18 1999. ACM Press.