Inferring Boolean Functions for Dynamical System Modeling

Pranay Ranjan

Diana Marculescu

In the analysis of non-linear dynamical systems, it is often of interest to determine a qualitative estimate of the behavior of the state variables. Modeling of the behavior of such dynamical systems as Boolean functions in conjunction with emulation on FPGA hardware can provide significant speed-ups of up to five orders of magnitude in runtime over conventional numerical methods of simulation. Moreover, the use of hardware allows tweaking the values of one or more of the state variables externally to observe how the system would respond to such exceptional or faulty scenarios. Previous attempts of inferring Boolean models were based on the use of decision trees, with time complexity being exponential in the number of features involved. A reduction of time complexity using methods like pruning leads to a compromise in terms of accuracy of the inferred models.

We propose a heuristic algorithm of polynomial time complexity, which infers Boolean functions for the next state of the variables characterizing the system as a function of the current state values. The algorithm is further extended to include a time-delayed sampling of variables under consideration which reflects the *refractory period* found in real world dynamical systems. This added flexibility of the algorithm to use the history of the behavior of the system provides enhanced precision. Moreover, the algorithm is structured to perform more efficiently for functions which have a *canalyzing* nature with respect to one or more of the features. Such an approach can find applications in the study of biological gene regulatory networks, modeling of continuous time circuit dynamics of non-ideal digital systems, or modeling of cyber-physical systems characterized by sigmoidal dynamic behavior. Fig. 1 shows the pseudo code for the inference algorithm with a linear complexity in the data series size. When applying this algorithm for generic or canalyzing random Boolean functions, a cross validation error of less than 10% could be achieved with a significantly smaller training set for a synthetic dataset with 15 features (Fig 2).

```
Do While (all negative examples covered){
    • Evaluate neg[] for all the variables.
    Do while (all positive examples covered){
        • Evaluate pos[] and indicator[] for all the variables
        // If (neg[i] == 0) indicator[] = 10000
        // (some arbitrarily high value)
        • Select the variable with the maximum indicator[].
        • Flag the positive examples which evaluate to 1
          when the selected variable is 1.
    }
    • Flag the unflagged negative esamples for which the
      currently formed clause evaluates to 0.
}
```

Fig. 1: Inference algorithm

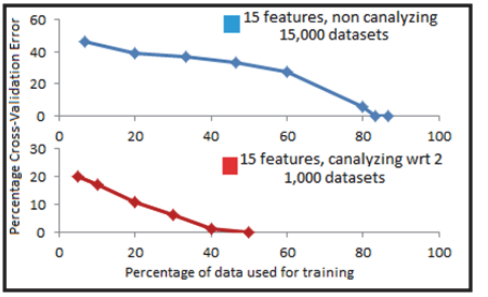

Fig. 2: Cross validation error for synthetic data