POLYNOMIALS FOR CLASSIFICATION TREES AND APPLICATIONS

IAN H. DINWOODIE

Technical Report #2008-07
June 25, 2008

This material was based upon work partially supported by the National Science Foundation under Grant DMS-0635449 to the Statistical and Applied Mathematical Sciences Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

POLYNOMIALS FOR CLASSIFICATION TREES AND APPLICATIONS

IAN H. DINWOODIE
DUKE UNIVERSITY

Abstract. This paper relates computational commutative algebra to tree classification with binary covariates. With a single classification variable, properties of uniqueness of a tree polynomial are established. In a binary multivariate context, it is shown how trees for many response variables can be made into a single ideal of polynomials for computations. Finally, a new sequential algorithm is proposed for uniform conditional sampling. The algorithm combines the lexicographic Groebner basis with importance sampling and it can be used for conditional comparisons of regulatory network maps. The binary state space leads to an explicit form for the design ideal, which leads to useful radical and extension properties that play a role in the algorithms.

1. Introduction

This paper is about representing classification trees as polynomials and applications in statistics. The polynomial representation is useful for computations, and also for determining equivalences and measuring differences between functions which may be obscured by a large tree representation.

Classification trees are valuable tools for approximating functions of several covariates, when traditional regression methods are too complex. Their value has been demonstrated in many applications (see Breiman, Friedman *et al.* (1984)).

The setup for the paper assumes binary covariates or binary predictors. Binary covariates lead to nice properties of uniqueness for polynomials from classification trees. Finite dimensional algebra for 0-dimensional ideals can be applied for binary covariates, both in foundations and in algorithms, which would be difficult in a more general setting. In particular, algebraic notions of radical ideal, standard monomials, and extension of solutions are used in Proposition 3.1, Theorem 3.1, and Theorem 4.2.

Coding a Boolean function (that is, a 0-1 valued function on the domain of binary covariates) as a polynomial from its unique disjunctive normal form is standard in

Date: June 2008.
Key words and phrases. Binary data, dynamical system, Groebner basis, hypercube, regulatory network, sequential importance sampling, syllabification.

computer science, and modern methods of algebra have been used to study these polynomials (see Clegg, Edmonds, and Impagliazzo (1996)). The polynomials that arise from classification trees in this paper have more general forms than disjunctive normal, but are used in the same spirit and are shown to be unique for practical trees.

Polynomials that we introduce in this paper from the fitted tree can be used for computing conditional distributions, simultaneously for all covariate values, as in the theory of generating functions. An example relating syllable stress to syllable weight in linguistics is presented in Section 2. In Section 3 we consider multiple binary responses. This leads to applications in binary dynamics and gene regulatory networks. Here we show how one can compare two maps on binary d-tuples near steady states by computing the dimension of a quotient space, and we also describe a sequential importance sampling algorithm for sampling on constrained states. The algebraic methods that we describe in Sections 3 and 4 are related to the work of Laubenbacher and Stigler (2004) and Laubenbacher and Sturmfels (2007) on modeling regulatory networks.

2. Tree Polynomials

A classification tree T on binary predictor variables $x_1, \ldots, x_d \in \{0,1\}$ is a rooted binary tree, with interior nodes and root node labelled with one of the covariate indices $i = 1, \ldots, d$ for the "splitting" variable, and terminal or leaf nodes $\mathcal{L}$ that have a probability distribution on the values $1, 2, \ldots, C$ of a class label.

Each edge in the tree is labelled with the expression $x_i = 1$ if it is a right branch from its parent node that splits variable i, or $x_i = 0$ if it is a left branch from its parent node that splits variable i.

A leaf node $\tau \in \mathcal{L}$ can be specified by a vector $\tau = (\tau_1, \tau_2, \ldots, \tau_d)$ where $\tau_i \in \{0, 1, \star\}$. If $\tau_i = 0$, the path back to the root node includes a split of the form $x_i = 0$; if $\tau_i = 1$, the path back to the root node includes a split of the form $x_i = 1$; if $\tau_i = \star$, the path back to the root node does not split on variable x_i.

Let L^τ be the variables that are split left in the path to τ, and R^τ those that are split right:

$$L^\tau = \{i : \tau_i = 0\} \tag{1}$$

$$R^\tau = \{i : \tau_i = 1\}. \tag{2}$$

We will use the indicator functions of these sets as d-tuples:

$$\mathbf{r}^\tau = I_{R^\tau} \tag{3}$$

$$\mathbf{l}^\tau = I_{L^\tau}. \tag{4}$$

Thus the support of $\mathbf{r}^\tau + \mathbf{l}^\tau$ is $R^\tau \cup L^\tau$, the set of indices in the branch to terminal node τ.

A *reduced* tree is one where in every branch from the root node to a leaf τ, a variable is used for a split at most once, so there are no vacuous combinations of splits like $x_i = 0$ and $x_i = 1$, and no repeated splits like $x_i = 1, x_i = 1$, on different edges in the same branch. Any tree may be taken to be reduced, without loss of generality, when the covariates $\mathbf{x}$ are binary as they are in our setting. Standard tree-fitting algorithms will always produce a reduced tree with binary covariates. One consequence of assuming reduced trees, as we will, is that $L^\tau \cap R^\tau = \emptyset$.

Let $H^d := \{0,1\}^d$, binary d-tuples of covariates or predictors. For each leaf node $\tau \in \mathfrak{L}$, define the set of $\mathbf{x}$ values in H^d consistent with the path to τ:

$$A^\tau = \{\mathbf{x} \in H^d : \mathbf{x}_{R^\tau} = \mathbf{1}, \mathbf{x}_{L^\tau} = \mathbf{0}\}$$

where $\mathbf{1}$ is a vector of all 1's and $\mathbf{0}$ is a vector of all 0's. Then $H^d = \cup_{\tau \in \mathfrak{L}} A^\tau$, a disjoint union.

Each leaf node τ is also labelled with a $C-$tuple of conditional probabilities for the C classes: $p_1^\tau, \ldots, p_C^\tau$ where

$$p_c^\tau = P(Y = c \mid \mathbf{x} \in A^\tau).$$

With a marginal probability distribution μ on H^d, there is a joint probability distribution on vectors $(\mathbf{x}, y) \in H^d \times \{1, 2, \ldots, C\}$ given by $\mu(\mathbf{x}) p^\tau(c),\ \mathbf{x} \in A^\tau$.

Define the *tree polynomial* in $\mathbb{C}[\mathbf{z}, \mathbf{s}] := \mathbb{C}[z_1, \ldots, z_C, s_1, \ldots, s_d]$ for the regression tree T by

$$p_T = \sum_{\tau \in \mathfrak{L}} \left((p_1^\tau z_1 + \cdots + p_C^\tau z_C) \mathbf{s}^{\mathbf{r}^\tau} (1 - \mathbf{s})^{\mathbf{l}^\tau} \right). \tag{5}$$

We are using the monomial notation $\mathbf{s}^{\mathbf{a}} := s_1^{a_1} \cdots s_d^{a_d}$ and $(1 - \mathbf{s})^{\mathbf{a}} = (1 - s_1)^{a_1} \cdots (1 - s_d)^{a_d}$.

This definition differs from a standard multivariate probability generating function in that the probabilities $\mu(\mathbf{x})$ on the covariates are not present, only conditional probabilities from the terminal nodes are used.

Define the ideal $I_{01} \subset \mathbb{C}[\mathbf{s}]$ as

$$I_{01} = \langle s_i^2 - s_i, i = 1, \ldots, d \rangle. \tag{6}$$

Note that the generators in the ideal I_{01} above are a Groebner basis for grevlex order. Observe that I_{01} is a 0-dimensional ideal, and it is radical by Seidenberg's Lemma (p. 250, Kreuzer and Robbiano (2000)). It is the design ideal of the hypercube H^d, in the terminology of Pistone, Riccomagno, and Wynn (2001). The normal form $NF_{I_{01}}(p)$ of a polynomial with respect to the ideal I_{01} is the remainder in the long division algorithm, and has the properties that $p - NF_{I_{01}}(p) \in I_{01}$ and no lead term in I_{01} divides any monomial of $NF_{I_{01}}(p)$ (p. 113, Kreuzer and Robbiano (2001)).

The following lemma is useful because it says that if one wants to apply a function f to the individual probability generating functions on the leaves of the tree, than one can get all the results by applying f once to the tree polynomial, then simplifying with the normal form.

Lemma 2.1. *For a univariate polynomial f,*

$$NF_{I_{01}}[f(p_T)] = \sum_{\tau \in \mathcal{L}} f(p_1^\tau z_1 + \cdots + p_C^\tau z_C)\mathbf{s}^{\mathbf{r}^\tau}(1-\mathbf{s})^{\mathbf{l}^\tau}.$$

Proof. Since $\mathrm{NF}_{I_{01}}(p+q) = \mathrm{NF}_{I_{01}}(p) + \mathrm{NF}_{I_{01}}(q)$, it is sufficient to prove the result for polynomials of the form $f(x) = x^k$. By direct calculation, one finds that $\mathrm{NF}_{I_{01}}(s_i(1-s_i)) = 0, \mathrm{NF}_{I_{01}}(s_i^2) = s_i, \mathrm{NF}_{I_{01}}((1-s_i)^2) = (1-s_i)$, and square-free monomials $\mathbf{s}^{\mathbf{a}}$ do not reduce. These properties show that

$$\begin{aligned} \mathrm{NF}_{I_{01}}[p_T^2] &= \mathrm{NF}_{I_{01}}\left[\sum_{\tau \in \mathcal{L}} (p_1^\tau z_1 + \cdots + p_C^\tau z_C)^2 \mathbf{s}^{2\mathbf{r}^\tau}(1-\mathbf{s})^{2\mathbf{l}^\tau}\right] \\ &= \sum_{\tau \in \mathcal{L}} (p_1^\tau z_1 + \cdots + p_C^\tau z_C)^2 \mathbf{s}^{\mathbf{r}^\tau}(1-\mathbf{s})^{\mathbf{l}^\tau} \end{aligned}$$

so the result holds for $f(x) = x^2$ and it follows by induction for $f(x) = x^k$. □

Proposition 2.1. *$Cov(I_{Y=1}, I_{Y=2} \mid \mathbf{x})$ is the coefficient on $z_1 z_2$ in $-\frac{1}{2}NF_{I_{01}}[{p_T}^2](\mathbf{x})$.*

Proof. Since $\mathrm{Cov}(I_{Y=1}, I_{Y=2} \mid \mathbf{x}) = -p_1^\tau p_2^\tau$ when $\mathbf{x} \in A^\tau$, we can use Lemma 2.1 with $f(x) = -x^2/2$. □

Example 2.1. Below is a simplified version of Table 7 from Duanmu, Kim, and Stiemon (2002), which is categorical data on syllable stress from the CMUDICT database, at `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`, the "CMU Pronouncing Dictionary."

The simplified version below records the weights of the last three syllables in variables w1, w2, w3, where 0 codes "heavy" and 1 codes everything else. The other variable S indicates which syllable has the main stress. Freq is the number of words of the type described in each row in the database.

```
Freq W1 W2 W3 S
1207 0 0 0 1
3004 0 0 0 2
277 0 0 0 3
3927 0 0 1 1
4214 0 0 1 2
138 0 0 1 3
3134 0 1 0 1
62 0 1 0 2
286 0 1 0 3
5102 0 1 1 1
1708 0 1 1 2
210 0 1 1 3
836 1 0 0 1
2220 1 0 0 2
135 1 0 0 3
1839 1 0 1 1
4077 1 0 1 2
337 1 0 1 3
```

A classification tree on the syllable of main stress can be fit in R (2007), where the covariates $\mathbf{x}$ in the above presentation are now $\mathbf{w}$, a mnemonic for weight:

```
1) root 33413 55920 1 ( 0.48020 0.47841 0.04139 )
  2) W2 < 0.5 22211 35470 2 ( 0.35158 0.60848 0.03994 )
    4) W3 < 0.5 7679 11850 2 ( 0.26605 0.68030 0.05365 ) *
    5) W3 > 0.5 14532 23220 2 ( 0.39678 0.57053 0.03269 ) *
  3) W2 > 0.5 11202 15630 1 ( 0.73523 0.22050 0.04428 ) *
```

The output gives a tree with three leaves marked with *. They correspond to the partition $\mathcal{P} = \{A^{\star,0,0} = \{\mathbf{w} : \text{w2=0, w3=0}\}, A^{\star,0,1} = \{\mathbf{w} : \text{w2=0, w3 =1}\}, A^{\star,1,\star} = \{\mathbf{w} : \text{w2=1}\}\}$.

From the three conditional distributions on the syllable of main stress at each terminal node, this rule follows: if the second syllable is heavy (w2=0), put the main stress on the second syllable (S=2). If the second syllable is not heavy, put the main stress on the first syllable. This rule is obtained by using the mode of the distribution on each leaf to estimate the correct value of S at each value of $\mathbf{w}$. Figure 1 shows the tree structure.

The misclassification rate for this rule is 11662/33413=35%, which is obtained by counting the number of times the rule predicts a different syllable for main stress than the data. (Duanmu *et al.* observe that existing rules have high misclassification rates on the CMUDICT data.) This does not necessarily mean that the rule

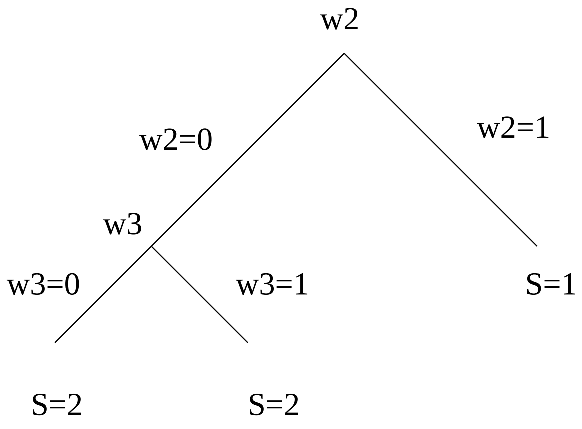

FIGURE 1. Graphical tree function for Example 2.1

```
> pt;
-13/100*z(1)*s(2)*s(3)+11/100*z(2)*s(2)*s(3)+1/50*z(3)*s(2)*s(3)
 +47/100*z(1)*s(2)-23/50*z(2)*s(2)-1/100*z(3)*s(2)
 +13/100*z(1)*s(3)-11/100*z(2)*s(3)-1/50*z(3)*s(3)
 +27/100*z(1)+17/25*z(2)+1/20*z(3)

> substitute(pt,s(1),0,s(2),0,s(3),0);
27/100*z(1)+17/25*z(2)+1/20*z(3)
```

FIGURE 2. Tree polynomial and operations for Example 2.1

will correctly place the main stress 65% of the time on actual text, since the relative frequencies in actual text are not uniform. Weights on the data that reflect relative frequencies may be introduced for more practical results.

Figure 2 below is edited from Singular (Greuel, Pfister, and Schöeneman (2007)), where pt is the tree polynomial with approximate coefficients, and it is evaluated at $\mathbf{x} = (0, 0, 0)$ to get the conditional distribution $(p^{(\star,0,0)})_j)_{j=1,2,3}$ on the leaf, corresponding to the partition with w2=0 and w3=0. .

3. Binary Maps and Algebra

We now specialize to the case of two classes ($C = 2$), which we will represent as $c = 0, 1$ rather than $c = 1, 2$. First we consider a single Boolean function f on H^d, and then we consider a $d-$tuple of Boolean functions on $H^d = \{0,1\}^d$. This leads to applications of binary regulatory networks.

A Boolean function $f : H^d \to \{0,1\}$ can be represented with a classification tree T_f, possibly in more than one way. The distributions at the terminal nodes will be degenerate ($p_0^\tau \in \{0,1\}, p_1^\tau = 1 - p_0^\tau$). The value of the function at a point $\mathbf{x} \in A^\tau$ is 0 if $p_0^\tau = 1$, or 1 if $p_1^\tau = 1$, or very simply $f(A^\tau) = p_1^\tau$.

With the tree polynomial p_{T_f} from the tree T_f from Section 2, set

$$q_{T_f}(\mathbf{s}) := p_{T_f}(0,1,\mathbf{s}) = \sum_{\tau \in \mathcal{L}} p_1^\tau \prod_{i \in R^\tau} s_i \prod_{j \in L^\tau} (1 - s_j) \in \mathbb{C}[\mathbf{s}]. \tag{7}$$

This will have square-free monomials for a reduced tree T, because the tree will not split on a single variable twice in one branch. We can call $q_{T_f} \in \mathbb{C}[\mathbf{s}]$ the *small tree polynomial* by contrast with $p_{T_f} \in \mathbb{C}[\mathbf{z}, \mathbf{s}]$.

Lemma 3.1. *The polynomial q_{T_f} evaluates like the Boolean function f on H^d for any tree that represents the Boolean function f:*

$$f(\mathbf{x}) = q_{T_f}(\mathbf{x}), \ \mathbf{x} \in H^d.$$

Proof. For a particular binary vector $\mathbf{x}$, let $\tau_\mathbf{x}$ be the terminal node whose partition set $A^{\tau_\mathbf{x}}$ contains $\mathbf{x}$.

For $\tau \neq \tau_\mathbf{x}$, there is an index k where either $\tau_k = 0, \tau_{\mathbf{x},k} = 1$, or $\tau_k = 1, \tau_{\mathbf{x},k} = 0$ where $\tau, \tau_\mathbf{x}$ are considered to be d-tuples in $\{0, 1, \star\}^d$ that uniquely determine the terminal node. (If this were not the case, then the vectors τ and $\tau_\mathbf{x}$ would differ only where one of them was $\star$, and then one of the sets $A^\tau, A^{\tau_\mathbf{x}}$ would be a subset of the other, not possible.)

In the first case, $\prod_{i \in R^\tau} x_i = 0$, in the second case, $\prod_{j \in L^\tau}(1 - x_j) = 0$. Thus for all $\tau \neq \tau_\mathbf{x}$:

$$p_1^\tau \prod_{i \in R^\tau} x_i \prod_{j \in L^\tau} (1 - x_j) = 0.$$

Finally, one monomial does not vanish: $p_1^{\tau_\mathbf{x}} \prod_{i \in R^\tau} x_i \prod_{j \in L^\tau}(1 - x_j) = p_1^{\tau_\mathbf{x}} = f(\mathbf{x})$. $\square$

Recall that the standard monomials for an ideal I are the monomials that are not contained in the ideal of lead terms of I, where the term ordering will be grevlex (or indeed any graded ordering). Let F_2 be the field of two elements $\{0, 1\}$ with addition mod 2.

Lemma 3.2. *For a reduced tree, the monomial terms in q_{T_f} are standard monomials for the ideal I_{01} in both $\mathbf{C}[\mathbf{s}]$ and $F_2[\mathbf{s}]$.*

Proof. The monomial terms in q_{T_f} are square-free if the tree is reduced. Furthermore, the defining generating set for I_{01} at (6) is a Groebner basis in grevlex order, by direct verification. Now no lead term from I_{01} can divide a square-free monomial term of q_{T_f}, so by the definition of Groebner basis it follows that the monomials in q_{T_f} are standard monomials. □

Proposition 3.1. *Let f and g be $0-1$-valued functions on H^d. Any two reduced trees T^1, T^2 representing f have equal tree polynomials $q_{T^1_f} = q_{T^2_f}$ in $\mathbb{C}[\mathbf{s}]$. Also, if $q_{T_f} = q_{T_g}$ for any reduced trees representing f and g, then $f = g$.*

Proof. If T^1 and T^2 are two trees representing f, then the polynomials $q_{T^1_f}$ and $q_{T^2_f}$ evaluate the same on H^d, by Lemma 3.1. So the two polynomials are equal as functions on H^d, and their difference $q_{T_f} - q_{T_g} \in I_{01}$. Since the trees are reduced, no lead term in the Groebner basis for I_{01} divides the monomials in $q_{T_f} - q_{T_g}$, so the difference must be 0 and the polynomials are equal in $\mathbb{C}[\mathbf{s}]$.

The second part follows from Lemma 3.1. □

In general, two different polynomials in $F_2[\mathbf{s}]$ may evaluate the same as functions on H^d (for example s_i and s_i^2), but the above result on uniqueness holds in $F_2[\mathbf{s}]$ as well.

Consider dynamics of the form $\mathbf{x}^0 \in H^d, \mathbf{x}^k = F(\mathbf{x}^{k-1}), k = 1, 2, \ldots, n$. Then there are binary maps $f_j : H^d \to \{0, 1\}, j = 1, \ldots, d$ for each coordinate, and

$$F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_d(\mathbf{x})). \tag{8}$$

Noiseless dynamics are

$$\mathbf{y}^k = F(\mathbf{x}^{k-1}), k = 1, 2, \ldots, n$$

In some examples, such as the regulatory network of Example 3.1 and 4.1, the system converges to a fixed point very quickly from almost any initial point. This makes reverse engineering from the noiseless dynamics practically impossible, as there can be only a small amount of high-dimensional data.

We will consider noisy data $\mathbf{y}^k$ starting at a point $\mathbf{x}^0$ in a specific form:

$$\mathbf{x}^k = \mathbf{y}^{k-1} + \mathbf{z}^k, k = 1, 2, \ldots, n \tag{9}$$

$$\mathbf{y}^k = F(\mathbf{x}^k) + \epsilon^k \tag{10}$$

where $\mathbf{z}^k, \epsilon^k$ are i.i.d. sequences in Z_2^d, vector addition is modulo 2. In particular, the independent components z_j^k of $\mathbf{z}^k$ take value 1 with probability $p > 0$, and the independent ϵ_j^k take the value 1 with probability $\alpha \geq 0$.

In this model, nonzero $\mathbf{z}^k$ help the estimation goal and may be viewed as "excitation" of the state, whereas the honest noise ϵ^k hinders estimation. Without the perturbation $\mathbf{z}^k$ of the state space, there is no hope in general to estimate F from system dynamics. The above model assumes that one observes the "excited" x-values and the noisy y-values in pairs $(\mathbf{x}^k, \mathbf{y}^k), k = 1, 2, \ldots, n$.

There are many methods for estimating or "reverse engineering" a map F – see the surveys Lattner, Kim *et al.* (2003) and Markowetz and Spang (2007). For example, one may try to recover the coordinate maps f_j of F using auto linear regression of y_j^k on $\mathbf{y}^{k-1},\ k = 1, 2, \ldots, n$. But the number of interaction terms in the linear model needed to fit arbitrary Boolean functions will be very large and the size of the design matrix for least-squares fitting will be impossibly huge. By contrast, one may fit a classification tree on the n pairs $(\mathbf{x}^k, y_j^k)$ for each coordinate j, and this procedure is very fast. It builds the interaction terms in the course of fitting the tree. One obtains a reduced binary classification tree T_j on each coordinate $j = 1, 2, \ldots, d$, and one may estimate each underlying Boolean function $f_j(\mathbf{x})$ by assigning to each terminal node the most likely value 0 or 1 in the distribution $(p_0^{\tau_{\mathbf{x}}}, p_1^{\tau_{\mathbf{x}}})$ on the node $\tau_{\mathbf{x}}$ whose partition $A^{\tau_{\mathbf{x}}}$ contains $\mathbf{x}$ for the tree T_j. The fitting procedure is computationally efficient, but it mixes up the "modeling" and the "fitting" procedures in such a way that the "dimension" of the model is unclear and it is possible to manipulate the fitting procedure to overfit data. So in general questions of model fit are difficult and computations such as those we discuss below for comparing maps are relevant.

Consistency results for fitted trees have been proven for i.i.d. data (Chapter 12, Breiman, Friedman *et al.*, 1993). But we believe that consistency has not been proven for the Markovian data $(\mathbf{x}^k, \mathbf{y}^k)$ in (9-10) that arise in regulatory networks. It is plausible that consistency should hold with $p > 0$ and $\alpha < 1/2$, with α sufficiently small possibly depending on p.

It will happen that two fitting procedures yield different maps F and G because of noise in the data, or because the fittings were done on two sequences that started at different initial points $\mathbf{x}_0$ and the orbits are not identical. It may be that one map F is theoretical and a second map G is an empirical estimate of the true map from tree classification or other statistical method. We consider now the problem of comparing two maps F and G.

Let π be the uniform distribution on the hypercube H^d. A first notion of comparison is the uniform measure of equality

$$\pi(\{\mathbf{x} \in H^d : F(\mathbf{x}) = G(\mathbf{x})\}).$$

This computation can be done quickly and easily with a Monte Carlo method by sampling uniformly exactly from H^d. So this calculation is very easy, but not interesting because only the fraction of the state space H^d that relates to biologically interesting states should be considered. Therefore conditional computations will be considered.

Thinking of F as a hypothesis, a conditional comparison on the preimage of a steady state $\mathbf{x}_0$ is

$$\pi(\{G = F \mid F(\mathbf{x}) = \mathbf{x}_0\}). \tag{11}$$

This type of conditional computation may be related to the notion of "relevant subsets" in Lehmann (p. 404, 1986) but we offer no formal theory or justification. We describe below how the computation can be done algebraically. In the next section we explain a Monte Carlo method for a slightly different conditional comparison.

Define the difference ideal

$$I_{F,G} := \langle p_{f_1} - p_{g_1}, p_{f_2} - p_{g_2}, \ldots, p_{f_d} - p_{g_d} \rangle. \tag{12}$$

In the case where we fix $G = \mathbf{a}$, where $\mathbf{a}$ is a point, then the notation $I_{F,\mathbf{a}}$ will mean the ideal of differences $\langle p_{f_1} - a_1, \ldots, p_{f_d} - a_d \rangle$.

Theorem 3.1. *The conditional probability that $G = F$ in (11) can be computed as*

$$\pi(\{G = F \mid F(\mathbf{x}) = \mathbf{x}_0\}) = \frac{\dim\left(\mathbb{C}[\mathbf{s}]/(I_{F,G} + I_{F,\mathbf{x}_0} + I_{01})\right)}{\dim\left(\mathbb{C}[\mathbf{s}]/(I_{F,\mathbf{x}_0} + I_{01})\right)}.$$

Proof. By definition,

$$\pi(\{G = F \mid F(\mathbf{x}) = \mathbf{x}_0\}) = \frac{|\{\mathbf{x} : F(\mathbf{x}) = G(\mathbf{x}), F(\mathbf{x}) = \mathbf{x}_0\}|}{|\{\mathbf{x} : F(\mathbf{x}) = \mathbf{x}_0\}|}.$$

Focusing on the numerator, $\{\mathbf{x} \in H^d : F(\mathbf{x}) = G(\mathbf{x}), F(\mathbf{x}) = \mathbf{x}_0\} = \mathbf{V}(I_{F,G} + I_{F,\mathbf{x}_0} + I_{01})$.

Now Theorem 2.10 of Cox, Little, O'Shea (1998) says that

$$|\mathbf{V}(I_{F,G} + I_{F,\mathbf{x}_0} + I_{01})| \leq \dim \mathbf{C}[\mathbf{s}]/(I_{F,G} + I_{F,\mathbf{x}_0} + I_{01}),$$

with equality if $I := I_{F,G} + I_{F,\mathbf{x}_0} + I_{01}$ is radical, which we now show.

Let p_i be the unique monic generator of the elimination ideal $I_{F,G} + I_{F,\mathbf{x}_0} + I_{01} \cap \mathbf{C}[s_i]$. Clearly $s_i^2 - s_i \in I \cap \mathbf{C}[s_i]$. Then p_i must be a factor of $s_i^2 - s_i$, so it could be any of $1, s_i, s_i - 1, s_i^2 - s_i$. Consider $p_{i,red} = p_i/\mathrm{GCD}(p_i, p_i')$. It would be 1 $(p_i' = 0), s_i$ $(p_i' = 1), s_i - 1$ $(p_i' = 1), s_i^2 - s_i$ $(p_i' = 2s_i - 1)$ in the four cases–that is, it is equal to p_i in all cases. Now by Proposition 2.7 of Cox, Little, O'Shea (1998), it follows that I is radical.

A similar argument works on the denominator.

□

To count the number of steady states or fixed points for F, we can set $G = id$, the identity map on H^d:

$$\{\mathbf{x} \in H^d : \mathbf{x} = F(\mathbf{x})\} = \mathbf{V}(I_{F,id}).$$

Then by an argument like the one in Theorem 3.1,

$$|\{\mathbf{x} \in H^d : \mathbf{x} = F(\mathbf{x})\}| = \dim \mathbb{C}[\mathbf{s}]/I_{F,id}. \tag{13}$$

Example 3.1. Let us consider a four cell case of the regulatory network of Albert and Othmer (2003). We will apply the result of Theorem 3.1 with F the hypothesized map, and $G = \hat{F}$ our estimate from tree classification.

In this example, each of four cells has 15 components with binary states, and the number of binary variables is then 60. The state space may be considered to be H^{60} in our notation. However certain boundary conditions are imposed on the first component (SIP) in each cell, and these give $x_1 = 1, x_{16} = 0, x_{31} = 0, x_{46} = 1$. So we may take the state space to be H^{56}.

We have a map F from Albert and Othmer (2003) on the 60 variables. Four of the variables are fixed, as above. The update on component wg (coordinate 2 in each cell) looks like

$$wg^{t+1} = (CIA^t \wedge SLP^t \wedge -CIR^t) \vee (wg^t \wedge [CIA^t \vee SLP^t] \vee -CIR^t)$$

in one cell. Some components are updated using variables in other cells, and two components update "instantly" (PH and SMO, coordinates 10 and 11) which affects the polynomial representation. See Appendix I and II for the polynomials.

The fixed points or steady states for the map number 10, and certain of them are of biological interest. In particular, the "wild type" **wt** from their Figure 4b (and Figure 6b) has the form:

```
cell   0   1   2   3
 [1]   1   0   0   1
 [2]   1   0   0   0
 [3]   1   0   0   0
 [4]   0   1   0   0
 [5]   0   1   0   0
 [6]   0   1   0   0
 [7]   0   1   0   0
 [8]   1   0   1   0
 [9]   1   0   1   1
[10]   1   0   1   0
[11]   1   1   1   0
[12]   1   0   1   1
[13]   1   0   1   1
[14]   1   0   1   0
[15]   0   0   0   1
```

Data was simulated with parameters $p = \alpha = .1$ and 5000 states of binary vectors in H^{60} were generated, starting at the wild type $\mathbf{wt}$. Doing tree classification on each coordinate of the data gives 60 coordinate maps in an estimate $\hat{F}$ of F. For example, the fitted tree on the second coordinate map (variable wg in cell 0) from `tree` in R is:

```
node), split, n, deviance, yval
      * denotes terminal node

1) root 5000 1192.00 0.60780
  2) V15 < 0.5 3912  733.00 0.75030
    4) V2 < 0.5 1615  402.40 0.52940
      8) V14 < 0.5 751   71.48 0.10650 *
      9) V14 > 0.5 864   79.83 0.89700 *
    5) V2 > 0.5 2297  196.50 0.90550 *
  3) V15 > 0.5 1088   94.06 0.09559 *
```

This corresponds to an empirical Boolean function

$$\hat{f}_2 = \texttt{(V2 and not V15) or (V14 and not V2 and not V15)}$$

and from (7) this gives a polynomial in $\mathbb{C}[s_1, \dots, s_{60}]$:

$$q_{\hat{f}_2} = s_2(1 - s_{15}) + s_{14}(1 - s_2)(1 - s_{15}).$$

By simple Monte Carlo sampling in H^{56}.

$$\pi(\{\mathbf{x} : \hat{F}(\mathbf{x}) = F(\mathbf{x})\}) = 0.00148.$$

With cells 0, 1, 2, 3 labelling each row and fifteen cell components across the top, the probabilities of a difference in $\hat{F}$ and F at each component are

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.23	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.06	0.17	0.14	0.00	0.00	0.06	0.06
2	0.00	0.13	0.50	0.00	0.50	0.25	0.50	0.12	0.44	0.42	0.17	0.50	0.50	0.44	0.06
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.17	0.14	0.00	0.00	0.06	0.06

This data (from simple Monte Carlo sampling) shows that the estimate $\hat{F}$ is working better in cells 0, 1, and 3, whereas the third cell (number 2) is the source of many errors.

Since most of the state space H^{56} is not biologically relevant, let us compute the conditional probabilities (11) with respect to the wild type steady state $\mathbf{wt}$ by counting standard monomials using Theorem 3.1:

$$\begin{aligned}\pi(\{\mathbf{x} : \hat{F}(\mathbf{x}) = F(\mathbf{x}) \mid F(\mathbf{x}) = \mathbf{wt}\}) &= \frac{|\{\mathbf{x} : \hat{F}(\mathbf{x}) = F(\mathbf{x}), F(\mathbf{x}) - \mathbf{wt} = 0\}|}{|\{\mathbf{x} : F(\mathbf{x}) - \mathbf{wt} = 0\}|} \\ &= 0/1451520.\end{aligned}$$

So clearly $\hat{F}$ does not well approximate F near this wild type.

On cells 0 and 1, the result is somewhat better:

$$\begin{aligned}\pi(\{\mathbf{x} : \hat{F}_{1:30}(\mathbf{x}) = F_{1:30}(\mathbf{x}) \mid F(\mathbf{x}) = \mathbf{wt}\}) &= \frac{|\{\mathbf{x} : \hat{F}_{1:30}(\mathbf{x}) = F_{1:30}(\mathbf{x}), F(\mathbf{x}) - \mathbf{wt} = 0\}|}{|\{\mathbf{x} : F(\mathbf{x}) - \mathbf{wt} = 0\}|} \\ &= 552960/1451520 = 38\%.\end{aligned}$$

The algebra calculation for the conditional probability is done almost immediately in Singular on a small laptop. If one were to attempt to sample directly in H^{56} and take only those $\mathbf{x}$ for which $F(\mathbf{x}) = \mathbf{wt}$, the efficiency would be quite low, since the proportion of states with $F(\mathbf{x}) = \mathbf{wt}$ is $1451520/2^{56} = 2.0 \times 10^{-11}$. Therefore the algebra is quite useful.

4. Sequential Sampling for Conditional Comparisons on Attractors

The full backward-orbit version of equation (11) for conditional comparisons of transition maps F and G is

$$\pi(\{G = F \mid F^{\infty} = \mathbf{x}_0\}) \tag{14}$$

where $\mathbf{x}_0$ is a steady state for F, and $\{F^{\infty} = \mathbf{x}_0\} = \cup_{k=1}^{\infty}\{\mathbf{x} : F^k(\mathbf{x}) = \mathbf{x}_0\}$ is the event that a point ever hits $\mathbf{x}_0$. G will be an estimate $\hat{F}$ for dynamics F from some statistical procedure like tree classification, and conditioning on a hypothesis F is analogous to classical conditional testing on margins.

For some steady states, it will happen that $\{\mathbf{x} : F^{\infty}(\mathbf{x}) = \mathbf{x}_0\}$ has very small probability in H^d, so uniform sampling from H^d to get points in $\{\mathbf{x} : F^{\infty}(\mathbf{x}) = \mathbf{x}_0\}$ may not be possible.

The method we propose for computing (14) combines Bayes' Rule, a lexicographic basis and sequential importance sampling (SIS).

Bayes' Rule rewrites (14) as

$$\pi(\{G = F \mid F^{\infty} = \mathbf{x}_0\}) = \frac{\pi(\{F^{\infty} = \mathbf{x}_0 \mid G = F\}) \cdot \pi(\{G = F\})}{\pi(\{F^{\infty} = \mathbf{x}_0\})}.$$

Then each of the three quantities in the expression can be evaluated: $\pi(\{F^{\infty} = \mathbf{x}_0 \mid G = F\})$ can done with the proposed SIS method, and both $\pi(\{G = F\})$ and $\pi(\{F^{\infty} = \mathbf{x}_0\})$ can be computed with Monte Carlo averages by direct sampling from H^d.

Now we describe the SIS procedure. The target distribution π will be uniform on the set

$$\Omega := \{\mathbf{x} \in H^d : G(\mathbf{x}) = F(\mathbf{x})\},$$

which can be challenging for expectations when its size is large. The set Ω may be considered a set of constrained 0-1 tables, with nonlinear polynomial constraints generalizing the linear constraints that arise in loglinear models (Chen, Dinwoodie, Sullivant (2006)). The proposal distribution p, from which we generate an i.i.d. sample $(X_i, i = 1, \ldots, n)$ in Ω, will be close to uniform.

The proposal distribution p will be expressed as a product of successive conditional distributions

$$p(\mathbf{x}) = p_d(x_d) \cdot p_{d-1}(x_{d-1}|x_d) \cdot p_{d-2}(x_{d-2}|x_d, x_{d-1}) \cdots p_1(x_1|x_d, \ldots, x_2)$$

just as a random point $(X_{i,1}, X_{i,2}, \ldots, X_{i,d}) \in \Omega$ will be generated sequentially backwards: $X_{i,d}, X_{i,d-1}, \ldots, X_{i,2}, X_{i,1}$.

The unnormalized weights w_i are defined by $w_i = 1/p(X_i)$. We also generate an i.i.d. sequence of Poisson integers with mean λ: $\tau_1, \tau_2, \ldots, \tau_n$. The SIS Monte Carlo estimate for $\mu^A := \pi(\{\mathbf{x} : F^\infty(\mathbf{x}) \in A \mid F(\mathbf{x}) = \hat{F}(\mathbf{x})\})$ is given by

$$\hat{\mu}^A_{n,\lambda} := \frac{1}{n} \sum_{i=1}^{n} I_A(F^{\tau_i}(X_i)) \frac{w_i}{\bar{w}}. \tag{15}$$

Proposition 4.1. *For a collection of fixed points $A \subset H^d$,*

$$\lim_{n\to\infty} \hat{\mu}^A_{n,\lambda} \leq \mu^A$$

with probability 1, and

$$\lim_{\lambda\to\infty} \lim_{n\to\infty} \hat{\mu}^A_{n,\lambda} = \mu^A.$$

Proof. With I_A the indicator function for the attractor of fixed points, we have

$$\hat{\mu}^A_{n,\lambda} := \frac{1}{\bar{w}} \frac{1}{n} \sum_{i=1}^{n} I_A(F^{\tau_i}(X_i)) w_i$$

and $\bar{w} \to E_p(w_i) = |\Omega|$ almost surely by the law of large numbers. Also, by the law of large numbers,

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^{n} I_A(F^{\tau_i}(X_i)) w_i &\to E_{p,\lambda}(I_A(F^{\tau_1}(X_1) w_1) \\ &\leq E_{p,\lambda}(I_A(F^{\infty}(X_1))) w_1 \\ &= \sum_{\mathbf{x}\in\Omega} I_A(F^{\infty}(\mathbf{x})) \end{aligned}$$

Together with the limit for $\bar{w}$, we have $\lim_{n\to\infty} \hat{\mu}^A_{n,\lambda} \leq \mu^A$ and

$$\lim_{n\to\infty} \hat{\mu}^A_{n,\lambda} = \sum_{t=0}^{\infty} \sum_{\mathbf{x}\in\Omega} \frac{I_A(F^t(\mathbf{x}))}{|\Omega|} \lambda^t e^{-\lambda}/t!$$

and the last sum converges to μ^A as $\lambda \to \infty$, which proves equality in the limit. □

A diagnostic measure of efficiency is the quantity cv^2 given by the sample variance of the normalized weights: $cv^2 = \mathrm{var}(\{w_i/\bar{w}\})$. It has been argued that $1/(1 + cv^2)$ can be used a measure of efficiency of the SIS estimator relative to i.i.d. sampling from π (p. 35, Liu (2001)).

To implement SIS for sampling on Ω, the procedure is the following.

Importance Sampling Algorithm on Ω:

(1) Compute a lexicographic Groebner basis for $I_\Omega := I_{F,\hat{F}} + I_{01}$ with variable order $s_1 > s_2 > \cdots > s_d$ in $\mathbb{C}[\mathbf{s}]$. If $I_\Omega = \langle 1 \rangle$, then Ω is empty and stop. Otherwise Ω is not empty and continue.

(2) For sample size n, let the index i run from 1 to n:

   (a) Using the polynomials from the lex basis that only involve s_d, determine which of $\{0,1\}$ solve the system and let $n_d \in \{1,2\}$ be the number of values in $\{0,1\}$ that solve the equations. Then uniformly sample $X_{i,d}$ from the set of roots, and let $p_d(X_{i,d}) = 1/n_d$.

   (b) Now consider the equations in the lex basis that involve s_{d-1} and s_d. Determine which of $\{0,1\}$ solve the system in x_{d-1} setting $s_d = X_{i,d}$, and let n_{d-1} be the number of roots. Choose $X_{i,d-1}$ uniformly from the roots, and set $p_{d-1}(X_{i,d-1}|X_{i,d}) = 1/n_{d-1}$.

   (c) Continue backwards counting the number of solutions n_{d-k} to the equations in the lex basis that involve variable $s_{d-k}, \ldots, s_d$, with $s_{d-k+1} = X_{d-k+1}, \ldots, s_d = X_d$. Choose $X_{i,d-k}$ uniformly from the n_{d-k} solutions, and set $p_{d-k}(X_{i,d-k}|X_{i,d-k+1}, \ldots, X_{i,d}) = 1/n_{d-k}$.

   (d) Complete $X_i = (X_{i,1}, \ldots, X_{i,d}) \in \Omega$ when $X_{i,1}$ is chosen and $p_d(X_{i,1}|X_{i,d}, \ldots, X_{i,2})$ is computed.

(3) Get $l_i = -\log(p_d(X_{i,d})) - \cdots - \log(p_1(X_{i,1}|X_{i,d}, \ldots, X_{i,2})$.

(4) Look at each element of the sample and the random times τ_i to compute $I_A(F^{\tau_i}(X_i))$ and $\hat{\mu}^A_{n,\lambda}$.

The following result justifies the method above and uses the explicit form of I_{01} from (6) to prove the sequential solution.

Proposition 4.2. *The Importance Sampling Algorithm on Ω above always produces an element $X_i \in \Omega$ if Ω is not empty, and the importance sampling weights w_i are*

$$w_i = e^{l_i}.$$

Proof. Assume Ω is not empty. By the Nullstellensatz and the Hilbert Basis Theorem, there is a finite set of polynomials that generate the ideal: $I_\Omega = \langle p_1, \ldots, p_g \rangle$. Now assume these polynomials are a lexicographic Groebner basis, which always exists. Suppose we have a partial solution $(a_{d-k+1}, \ldots, a_d)$– that is, a solution in $\mathbb{C}^k$ for some $k = 1, \ldots, d-1$ to the equations in $I_\Omega \cap \mathbb{C}[s_{d-k+1}, \ldots, s_d]$. This can always be extended another dimension to a solution $(a_{d-k}, a_{d-k+1}, \ldots, a_d) \in \mathbb{C}^{k+1}$, because the polynomial $s_{d-k}^2 - s_{d-k} \in I_\Omega \cap \mathbb{C}[s_{d-k}, \ldots, s_d]$ and the Extension Theorem (Theorem 3, p. 115, Cox, Little, O'Shea (1997)) says the extension is possible with the presence of such a univariate polynomial.

The weights w_i are defined by $w_i = 1/p(X_i)$ where p is the proposal distribution. The sequential construction of X_i shows that

$$p(X_i) = p_d(X_{i,d}) \cdot p_{d-1}(X_{i,d-1}|X_{i,d}) \cdots p_1(X_{i,1}|X_{i,d}, \ldots, X_{i,2}).$$

This gives weight

$$\begin{aligned} w_i &= e^{-\sum_{k=0}^{d-1} \log p_{d-k}(X_{i,k}|X_{i,d},\ldots,X_{i,d-k+1})} \\ &= e^{l_i}. \end{aligned}$$

□

Example 4.1. Let us continue with the example of Albert and Othmer (2003). A steady state, different than the wild type already discussed, is the "ectopic" pattern **e** from Figure 5b of their paper:

```
cell  0   1   2   3
 [1]  1   0   0   1
 [2]  0   0   0   0
 [3]  0   0   0   0
 [4]  0   0   0   0
 [5]  0   0   0   0
 [6]  0   0   0   0
 [7]  0   0   0   0
 [8]  0   0   0   0
 [9]  1   1   1   1
[10]  0   0   0   0
[11]  0   0   0   0
[12]  1   1   1   1
[13]  1   1   1   1
[14]  0   0   0   0
[15]  1   1   1   1
```

The set $\{\mathbf{x} : F^\infty(\mathbf{x}) = \mathbf{e}\}$ has probability approximately 0.01577, from direct Monte Carlo sampling in H^{56}. Bayes' Rule then gives

$$\pi(\{\hat{F} = F \mid F^\infty = \mathbf{e}\}) = \frac{\pi(\{F^\infty = \mathbf{e} \mid \hat{F} = F\} \cdot \pi(\{\hat{F} = F\})}{.01577}$$

and $\pi(\Omega = \{\hat{F} = F\})$ is approximately .00148, by sampling directly in H^{56} and observing differences. Now the SIS algorithm for sampling in Ω will get the final quantity $\pi(\{F^\infty = \mathbf{e} \mid \hat{F} = F\}$. With a sample size of $n = 10,000$ and $\lambda = 40$, we get of $\pi(\{F^\infty = \mathbf{e} \mid \hat{F} = F\} \approx 0.01068$, with a cv^2 value of 0.46. These numbers combine for

$$\pi(\{\hat{F} = F \mid F^\infty = \mathbf{e}\}) = \frac{.01068 \times .00148}{.01577} = .00100.$$

For the wild type **wt**, the conditional probability estimate $\pi(\{\hat{F} = F \mid F^\infty = \mathbf{wt}\})$ is 0.00000.

Error estimates are best done by repeating the calculations for independent estimates, then using simple methods for confidence intervals.

Finally, one can estimate the size of $\Omega := \{\mathbf{x} \in H^{56} : \hat{F} = F\}$ in two ways. First, relative to all of H^{56}, Ω has probability around .0015, giving a size estimate for $|\Omega| \approx 2^{56} \times .0015 = 1.1 \times 10^{14}$. This agrees with the estimate from SIS that uses the mean $\bar{w}$ of the unnormalized weights.

These calculations are valuable for comparing $\hat{F}$ and F, and give a mixed picture for how well tree classification worked in this particular instance. The fitted trees did not come out very well in cell 2, and conditioning on subsets related to recognizable steady states did not improve measures of fit.

5. Conclusions and Further Problems

This paper has shown how to connect computational tools from algebra to classification trees, and some of the computational advantages of this connection. We have looked at tree classification in examples where it works well in some ways, but also shows some performance weaknesses based on our calculations that used the algebra and sequential importance sampling methods of Sections 3 and 4.

We have treated the case of binary covariates, where the algebra for the variety of points $\{0, 1\}^d$ is both explicit and convenient. Extensions to more general sets of covariates would be interesting.

In the course of algebraic calculations in software, there can be errors from integer overflow on realistic problems with 60 binary variables. Cocoa (2007) can handle larger integers than Singular (2007) at this time, and this can be valuable in applications.

Acknowledgments. We thank Henry Wynn for discussions on this topic, in particular on perturbations of dynamical systems. We think David W. Dinwoodie for discussions on syllabification and Example 2.1. This material was based upon work partially supported by the National Science Foundation under Grant DMS-0635449 to the Statistical and Applied Mathematical Sciences Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Albert, R., and Othmer, H. G. (2003). The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster. Journal of Theoretical Biology,* **223**, 1-18.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1993). *Classification and Regression Trees.* Chapman and Hall, New York.

Clegg, M., Edmonds, J., and Impagliazzo, R. (1996). Using the Groebner basis algorithm to find proofs of unsatisfiability. In Proc. 28th ACM Symposium on Theory of Computing, pages 174–183.

Cover, T. M., and Thomas, J. A. (1991). *Elements of Information Theory.* Wiley, New York.

Chen, Y., Dinwoodie, I. H., and Sullivant, S. (2006). *Annals of Statistics*, **34**, 523-545.

CoCoATeam (2007). *CoCoA: a system for doing Computations in Commutative Algebra.* Available at http://cocoa.dima.unige.it.

Cox, D., Little, J., O'Shea, D. (1998). *Using Algebraic Geometry.* Springer, New York.

Cox, D., Little, J., O'Shea, D. (1997). *Ideal, Varieties, and Algorithms, Second Edition.* Springer, New York.

Duanmu, S., Kim, H.-Y., and Stiemon, N. (2002). Stress and Syllable Structure in English: Approaches to Phonological Variations. Manuscript.

Greuel, G.-M., Pfister, G., and Schönemann, H. (2007). SINGULAR 3.0.4. *A Computer Algebra System for Polynomial Computations.* Centre for Computer Algebra, University of Kaiserslautern `http://www.singular.uni-kl.de`.

Jain, J., Bitner, J., Fussell, D. S., and Abraham, J. A. (1992). Probabilistic verification of Boolean functions. *Formal Methods in System Design*, **1** 63-117.

Kreuzer, M., and Robbiano, L. (2000). *Computational Commutative Algebra I.* Springer, New York.

Lattner, A. D., Kim, S., Cervone, G., and Grefenstette, J. J. (2003). Experimental comparison of symbolic learning programs for the classification of gene network topology models. In *Annual Meeting of the GI Working Group "Machine Learning, Knowledge Discovery, Data Mining (FGML), Karlsruhe, Germany, 2003.* URL http://www.science.gmu.edu/ skime/papers/Lattner.

Laubenbacher, R., and Stigler, B. (2004). A computational algebra approach to the reverse engineering of gene regulatory networks. *Journal of Theoretical Biology*, **229** 523-537.

Laubenbacher, R., and Sturmfels, B. (2007) Computer algebra in systems biology. Manuscript

Lehmann, E. L. (2005). *Testing Statistical Hypotheses, Third Edition.* Springer, New York.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing.* Springer, New York.

Markowetz, F., and Spang, R. (2007). Inferring cellular networks – a review. *BMC Bioinformatics*, **8**(S6).

Pistone, G., Riccomagno, E., and Wynn, H. (2001). *Algebraic Statistics: Computational Commutative Algebra in Statistics.* Chapman and Hall, New York.

R Development Core Team (2007). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Appendix I: Polynomial maps in Singular for F, Example 3.1:

```
ring r=0, (w(1..15),x(1..15),y(1..15),z(1..15)),dp;
poly p01=1;
poly p02= w(14)*w(1)*(1-w(15)) + w(2)*(w(14)+w(1)-w(14)*w(1))*(1-w(15)) - w(14)*
w(1)*(1-w(15))*w(2)*(w(14)+w(1)-w(14)*w(1))*(1-w(15));
poly p03=w(2);
poly p04= (z(3) + x(3)-z(3)*x(3))*(1-w(1));
poly p05=w(4);
poly p06=w(5)*(1-w(15));
poly p07=w(6);
poly p08=w(14)*(1-w(5))*(1-w(15));
poly p09= w(8) + w(9)*(1-z(7))*(1-x(7)) - w(8)* w(9)*(1-z(7))*(1-x(7));
poly p012= 1-w(5);
poly p013=w(12);
poly p014=w(13)*(w(11)+z(6) + x(6) - w(11)*z(6) -w(11)*x(6) - x(6)*z(6) + w(11)*
x(6)*z(6));
poly p015= w(13)*(1-w(11))*(1-z(6))*(1-x(6));

poly p11=0;
poly p12= x(14)*x(1)*(1-x(15)) + x(2)*(x(14)+x(1)-x(14)*x(1))*(1-x(15)) - x(14)*
x(1)*(1-x(15))*x(2)*(x(14)+x(1)-x(14)*x(1))*(1-x(15));
poly p13=x(2);
poly p14= (w(3) + y(3)-w(3)*y(3))*(1-x(1));
poly p15=x(4);
poly p16=x(5)*(1-x(15));
poly p17=x(6);
poly p18=x(14)*(1-x(5))*(1-x(15));
poly p19= x(8) + x(9)*(1-w(7))*(1-y(7)) - x(8)* x(9)*(1-w(7))*(1-y(7));
poly p112= 1-x(5);
poly p113=x(12);
poly p114=x(13)*(x(11)+w(6) + y(6) - x(11)*w(6) -x(11)*y(6) - y(6)*w(6) + x(11)*
y(6)*w(6));
poly p115= x(13)*(1-x(11))*(1-w(6))*(1-y(6));

poly p21=0;
poly p22= y(14)*y(1)*(1-y(15)) + y(2)*(y(14)+y(1)-y(14)*y(1))*(1-y(15)) - y(14)*
y(1)*(1-y(15))*y(2)*(y(14)+y(1)-y(14)*y(1))*(1-y(15));
poly p23=y(2);
poly p24= (x(3) + z(3)-x(3)*z(3))*(1-y(1));
poly p25=y(4);
poly p26=y(5)*(1-y(15));
poly p27=y(6);
poly p28=y(14)*(1-y(5))*(1-y(15));
poly p29= y(8) + y(9)*(1-x(7))*(1-z(7)) - y(8)* y(9)*(1-x(7))*(1-z(7));
poly p212= 1-y(5);
poly p213=y(12);
poly p214=y(13)*(y(11)+x(6) + z(6) - y(11)*x(6) -y(11)*z(6) - z(6)*x(6) + y(11)*
z(6)*x(6));
poly p215= y(13)*(1-y(11))*(1-x(6))*(1-z(6));

poly p31=1;
poly p32= z(14)*z(1)*(1-z(15)) + z(2)*(z(14)+z(1)-z(14)*z(1))*(1-z(15)) - z(14)*
z(1)*(1-z(15))*z(2)*(z(14)+z(1)-z(14)*z(1))*(1-z(15));
poly p33=z(2);
poly p34= (y(3) + w(3)-y(3)*w(3))*(1-z(1));
poly p35=z(4);
poly p36=z(5)*(1-z(15));
poly p37=z(6);
poly p38=z(14)*(1-z(5))*(1-z(15));
poly p39= z(8) + z(9)*(1-y(7))*(1-w(7)) - z(8)* z(9)*(1-y(7))*(1-w(7));
poly p312= 1-z(5);
poly p313=z(12);
poly p314=z(13)*(z(11)+y(6) + w(6) - z(11)*y(6) -z(11)*w(6) - w(6)*y(6) + z(11)*
w(6)*y(6));
poly p315= z(13)*(1-z(11))*(1-y(6))*(1-w(6));

poly p010=p09*(p37+p17-p37*p17);
poly p011= (1-p09)+p37+p17 - (1-p09)*p37 - (1-p09)*p17 - p17*p37 + (1-p09)*p37*p
17;
poly p110=p19*(p07+p27-p07*p27);
poly p111= (1-p19)+p07+p27 - (1-p19)*p07 - (1-p19)*p27 - p27*p07 + (1-p19)*p07*p
27;
poly p210=p29*(p17+p37-p17*p37);
poly p211= (1-p29)+p17+p37 - (1-p29)*p17 - (1-p29)*p37 - p37*p17 + (1-p29)*p17*p
37;
poly p310=p39*(p27+p07-p27*p07);
poly p311= (1-p39)+p27+p07 - (1-p39)*p27 - (1-p39)*p07 - p07*p27 + (1-p39)*p27*p
07;
```

Appendix II: Polynomial maps for $\hat{F}$ from fitted trees, Example 3.1:

```
poly f01=1;
poly f02=w(14)*(1-w(2))*(1-w(15)) + w(2)*(1-w(15))   ;
poly f03=w(2);
poly f04= 0;
poly f05= w(4);
poly f06= (1-w(15))*w(5);
poly f07= w(6);
poly f08= (1-w(15))*(1-w(5))*w(14);
poly f09= w(8) + (1-z(7))*(1-x(7))*w(9)*(1-w(8));
poly f010=x(7)*w(8) + z(7)*(1-x(7))*w(8);
poly f011= x(7)+z(7)*(1-x(7)) + (1-w(8))*(1-w(9))*(1-z(7))*(1-x(7));
poly f012= 1-w(5);
poly f013= w(12);
poly f014= w(11)*w(13) + x(6)*(1-w(11))*w(13)+z(6)*(1-x(6))*(1-w(11))*w(13);
poly f015= (1-z(6))*w(13)*(1-x(6))*(1-w(11))   ;

poly f11=0;
poly f12= (1-x(15))*x(14)*x(2);
poly f13=x(2);
poly f14= w(3);
poly f15= x(4) ;
poly f16= (1-x(15))*x(5);
poly f17=x(6);
poly f18= (1-x(15))*x(14)*(1-x(5));
poly f19=x(8) ;
poly f110= w(6)*x(8);
poly f111= w(6)*x(8) + w(7)*x(9)*(1-x(8))+(1-w(7))*x(9)*(1-x(8))+(1-x(9))*(1-x(8
)) ;
poly f112= (1-x(5));
poly f113=x(12);
poly f114= x(11)*x(13) + w(6)*(1-x(11))*x(13);
poly f115= (1-w(6))*x(13)*(1-x(11));

poly f21=0;
poly f22 = 0;
poly f23= 0;
poly f24= (z(3)) + x(3)*(1-z(3));
poly f25= 1;
poly f26=  0;
poly f27= 0     ;
poly f28= 0;
poly f29 = (1-z(7))*(1-x(7));
poly f210= 0;
poly f211=  x(6)+z(7)*(1-x(6)) + x(7)*(1-z(7))*(1-x(6)) ;
poly f212=  0;
poly f213= 0;
poly f214= 0;
poly f215=0;

poly f31=1;
poly f32=z(2)*(1-z(15)) +z(14)*(1-z(2))*(1-z(15)) ;
poly f33= z(2);
poly f34= 0;
poly f35= z(4);
poly f36=(1-z(15))*(z(5));
poly f37= z(6);
poly f38=(1-z(15))*(1-z(5))*z(14);
poly f39= z(8)+(1-z(11))*(1-w(7))*z(9)*(1-z(8))  ;
poly f310=  w(6)*z(8);
poly f311= w(6)*z(8) + (1-z(8));
poly f312=  (1-z(5));
poly f313=  z(12);
poly f314= z(11)*z(13) + w(6)*(1-z(11))*z(13);
poly f315=0 ;
```