Boolean Function Evaluation Over a Sample

Lisa Hellerstein **Devorah Kletenik** **Patrick Lin**
Department of Computer Science and Engineering
NYU Polytechnic School of Engineering
Brooklyn, NY 11201
{lisa.hellerstein,dkletenik,patrick.lin}@nyu.edu

Abstract

We consider the problem of minimizing expected prediction cost, when there are costs associated with determining attribute values. Given a Boolean hypothesis function and a sample of this function, we seek to minimize the average prediction cost over the sample (Sample-BFE problem). We define and explore two different versions of this problem and show how existing work can be used to yield approximation algorithms for k-of-n and CDNF formulas.

1 Introduction

We consider the problem of minimizing expected prediction cost, given a (representation of a) Boolean hypothesis function that we want to use for prediction. We assume that there is a cost c_i associated with obtaining the value of the i-th bit of the function. In contrast to previous work on *Stochastic Boolean Function Evaluation* (Stochastic-BFE, also known as sequential testing of Boolean functions), here we do not assume independence of the attributes. Instead, we use the empirical distribution defined by a given sample of the function. That is, we seek to minimize the average prediction cost over the given sample. We call this problem *Sample-Based Boolean Function Evaluation* (Sample-BFE). We are interested in approximation algorithms solving the Sample-BFE problem for standard types of Boolean functions, such as linear threshold formulas.

In the Sample-BFE problem, we seek to find an evaluation procedure (we use the word strategy interchangeably) that correctly computes the given hypothesis function on all possible examples, even those that do not appear in the sample. We also discuss a related problem in which the evaluation procedure must correctly compute the given hypothesis function on all examples in the sample, but can produce incorrect outputs for examples that are not in the sample. We call this the Relaxed Sample-BFE, because of the less restrictive requirement on the correctness of the evaluation procedure.

Relaxed Sample-BFE is similar to the problem of producing a decision tree of minimum average cost given a sample of a Boolean function, where the Boolean function is not given as part of the input. That problem has an $O(\log m)$-approximation algorithm, where m is the size of the sample [10, 4]. This same approximation algorithm factor was recently achieved for the more general version of the problem where there is a given probability distribution over the elements in the sample, and the goal is to minimize expected cost with respect to the distribution (rather than just average cost over the sample) [7]. In Relaxed Sample-BFE, since we are given the function along with the sample, we can take advantage of properties of that function to achieve better approximation factors.

The Sample-BFE Problem and the Relaxed Sample-BFE problem can have very different solutions, given the same input. To illustrate this, suppose the function we want to evaluate is the OR function

of n variables. This function has a single negative example, the all 0's assignment. Suppose the input sample does not contain this example. In Relaxed Sample-BFE, the optimal evaluation procedure for this sample is just to output 1, without looking at any of the input variables. This procedure will output an incorrect prediction on the one negative assignment of the function, but since that assignment is not in the sample, it doesn't matter. In contrast, in the Sample-BFE problem, the evaluation procedure must specify the order in which to request the values of all n variables until either a variable is found to have the value 1, or all variables have been found to have the value 0. The evaluation procedure will thus correspond to a permutation of the variables that minimizes the average cost of finding a variable set to 1 in each of the examples in the sample. If the costs c_i are all equal, this is equivalent to the min-sum set cover problem, which has an approximation algorithm achieving a factor of 4 approximation; achieving a better approximation factor is NP-hard [9].

We consider two types of evaluation strategies in what follows, adaptive and non-adaptive. An adaptive strategy corresponds to a decision tree, where the decision as to which variable to observe next can depend on the results of previous observations. The decision tree may be represented implicitly. A non-adaptive strategy observes the variables in the order specified by a fixed permutation of the variables, until the function value can be determined. Non-adaptive strategies have a compact representation that can be pre-computed before prediction time, but may not be able to achieve the same costs achieved by adaptive strategies.

Both the Sample-BFE Problem and its relaxed variant require correct evaluation of the hypothesis on all examples in the sample. We leave for future work problems where perfectly accurate evaluation of the input hypothesis is not required. We note that if the input hypothesis has in fact been learned from a random sample, since an algorithm solving the Sample-BFE algorithm must produce an evaluation procedure that computes the hypothesis exactly, the prediction accuracy of the hypothesis is preserved. In contrast, an algorithm solving the relaxed Sample-BFE has no such restriction, and will be at risk of overfitting the sample.

2 Definitions

In some applications, such as medical diagnosis, determining the value of an attribute x_i during prediction requires the performance of a test. In what follows, we use the phrase "testing x_i" to mean the process of determining the value of x_i.

A partial assignment is a vector $b \in \{0,1,*\}^n$. We view b as an assignment to variables $x_1, \ldots, x_n$. We will use $b \in \{0,1,*\}^n$ to represent the outcomes of tests, where for $\ell \in \{0,1,*\}$, $b_i = \ell$ means that test i has outcome l, and $b_i = *$ means that the outcome of test i is not known.

For partial assignments $a, b \in \{0,1,*\}^n$, a is an *extension* of b, written $a \succ b$, if $a_i = b_i$ for all $b_i \neq *$. Given $f : \{0,1\}^n \to \{0,1\}$, a partial assignment $b \in \{0,1,*\}^n$ is a *0-certificate* (*1-certificate*) of f if $f(a) = 0$ ($f(a) = 1$) for all a such that $a \succ b$. Given cost vector $c = (c_1, \ldots, c_n)$, the cost of a certificate b is $\sum_{j:b_j \neq *} c_j$.

Let $N = \{1, \ldots, n\}$. In what follows, we assume that utility functions are integer-valued. In the context of standard work on submodularity, a *utility function* is a function $g : 2^N \to \mathbb{Z}_{\geq 0}$. Given $S \subseteq N$ and $j \in N$, $g_S(j)$ denotes the quantity $g(S \cup \{j\}) - g(S)$. We will also use the term *utility function* to refer to a function $g : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$ defined on partial assignments. Let $g : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$, be such a utility function, and let $b \in \{0,1,*\}^n$. We define $g(S,b) = g(b')$ where b' is the partial assignment such that $b'_i = b_i$ for $i \in S$, and $b'_i = *$ otherwise. For $j \in N$, we define $g_{S,b}(j) = g(S \cup \{j\}, b) - g(S,b)$. If $j \notin S$ and $b_j \neq *$, this denotes the increase in utility of adding j to S when j gets the value indicated by b. When b represents test outcomes, this is the increase in utility of testing j. For $\ell \in \{0,1,*\}$, the quantity $b_{x_i \leftarrow \ell}$ denotes the partial assignment that is identical to b except that $b_i = \ell$.

Utility function $g : \{0,1,*\}^n \to \mathbb{Z}_{\geq 0}$ is *monotone* if for $b \in \{0,1,*\}^n$, $i \in N$ such that $b_i = *$, and $\ell \in \{0,1\}$, $g(b_{x_i \leftarrow \ell}) - g(b) \geq 0$. Utility function g is *submodular* if for all $b, b' \in \{0,1,*\}^n$ and

$\ell \in \{0,1\}$, $g(b_{x_i \leftarrow \ell}) - g(b) \geq g(b'_{x_i \leftarrow \ell}) - g(b')$ whenever $b' \succ b$ and $b_i = b'_i = *$. In the testing context, if the n test outcomes are predetermined, submodularity means that the value of a given test (measured by the increase in utility) will not increase if we delay that test until later.

A linear threshold formula (LTF) with integer coefficients has the form $\sum_{i=1}^{n} a_i x_i \geq \theta$ where the a_i and θ are integers. It represents the function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that $f(x) = 1$ if $\sum_{i=1}^{n} a_i x_i \geq \theta$, and $f(x) = 0$ otherwise. A CNF formula is a conjunction (AND) of disjunctions (ORs). A DNF formula is a disjunction (OR) of conjunctions (ANDs). A CDNF formula for Boolean function f is a pair (ϕ_0, ϕ_1) where ϕ_0 and ϕ_1 are CNF and DNF formulas for f, respectively.

In the Stochastic Boolean Function Evaluation (Stochastic-BFE) problem, we are given as input a representation of a Boolean function $f(x_1, \ldots, x_n)$. For each x_i, we are given c_i and p_i, where $c_i > 0$ is the cost of testing x_i, $0 < p_i < 1$, and $p_i = \Pr[x_i = 1]$. We assume the x_i are independent. The problem is to find an adaptive evaluation strategy minimizing the expected cost of evaluating f.

We use ADAPT to denote the expected cost of the optimal adaptive strategy and NONADAPT to denote the expected cost of the optimal non-adaptive strategy.

3 Results for k-of-n formulas

A k-of-n formula is a Boolean function on n variables whose output is 1 if and only if at least k of its input variables are 1. These functions correspond to unweighted linear threshold functions, where the coefficients are all 1. When $k = 1$, the k-of-n formula is the OR function. There is a polynomial-time exact algorithm for the Stochastic-BFE problem for k-of-n formulas, which exploits the assumed independence of the attributes [11, 5, 12, 6].

The Sample-BFE problem for k-of-n formula is harder. As previously discussed, Sample-BFE problem for OR functions has a 4-approximation algorithm [9]. Here we present additional results on approximating the Sample-BFE problem for k-of-n formulas, based on existing results and techniques in the literature.

3.1 Non-adaptive approximation for Sample-BFE of k-of-n formulas with unit costs

We consider the Sample-BFE problem for k-of-n formulas with unit costs. In this section, we use algorithms for a problem called *generalized min-sum set cover*. The input to this problem is a universe U of n elements and a collection $S = \{S_1, \ldots, S_m\}$ subsets S_i of U. Each set S_i has a covering requirement $K(S_i)$. The goal is to find an ordering (permutation) of the elements that minimizes the average *covering time* of the sets. The covering time of a set S_i is the first time that $K(S_i)$ elements from the set have been chosen, assuming that for $j \in \{1, \ldots, n\}$ the j-th element in the permutation is chosen at time j.

The generalized min-sum set cover problem was introduced by Azar et al. [2]. The best approximation algorithm for this problem is due to Skutella and Williamson [13]. It achieves an approximation factor of 28 (improving on the factor of 485 approximation achieved by Bansal et al. [3]). Let `AlgSW` denote the algorithm of Skutella and Williamson.

Theorem 1. *There is an algorithm for the Sample-BFE problem for k-of-n formulas with unit costs that yields a constant-factor approximation with respect to NONADAPT.*

Proof. We can view the Sample-BFE problem for k-of-n formulas as two separate generalized min-sum set cover problems: one for the positive examples in the sample and one for the negative ones. The n variables of the function correspond to the elements of the set U. The examples in the sample correspond to the subsets S_i. For the positive (negative) examples, x_i *covers* a given example if $x_i = 1$ ($x_i = 0$) in that example. For the positive examples, each example must be covered at least k times (yielding a 1-certificate), and for the negative examples, each example must be covered at least $n - k + 1$ times (yielding a 0-certificate). Minimizing the average covering time of the

positive (negative) examples corresponds to finding the optimal non-adaptive strategy for evaluating the k-of-n formula on the sample consisting of only the positive (negative) examples.

We can use the `AlgSW` to solve each of these problems separately. Because `AlgSW` achieves a solution within a factor of 28 of optimal, the evaluation procedure produced for the positive (negative) examples will be within a factor of 28 of the optimal non-adaptive evaluation procedure minimizing the average cost of evaluation on the positive (negative) examples.

To evaluate a k-of-n formula on any example, we can simply run the two copies of `AlgSW` in round-robin fashion, one copy for the positive examples and one copy for the negative examples (with the appropriate covering requirements for each copy), until the first copy finds a 1-certificate, or the second copy finds a 0-certificate. (Once a variable has been tested by one of the two algorithms in the round-robin, we do not actually need to test it in the other algorithm, since we can just use the test result that was already determined.)

The optimal non-adaptive algorithm solving the Sample-BFE problem for the whole sample cannot have average cost that is lower than the optimal non-adaptive algorithm solving the Sample-BFE problem on the positive (negative) examples alone. The average cost incurred by the round-robin procedure on the positive (negative) examples in the sample is no more than twice the average cost incurred by running just the first copy of `AlgSW` on those examples. Thus the round-robin procedure achieves an approximation bound of at most 56 of NONADAPT. □

3.2 Adaptive Approximation for Sample-BFE of k-of-n formulas when k is constant

We describe an algorithm solving the Sample-BFE problem when k is constant and arbitrary costs are allowed. We use techniques from an algorithm due to Cicalese et al. [7] to get an approximation bound with respect to the optimal adaptive strategy.

Theorem 2. *There is an algorithm for the Sample-BFE problem for k-of-n formulas when k is constant that yields an $O(\log n)$ approximation with respect to ADAPT.*

Proof. Given a sample of a Boolean function f containing m examples, and costs for the variables, the algorithm of Cicalese et al. can be used to produce a decision tree consistent with the sample whose average evaluation cost, computed over the sample, is within a factor of $O(\log m)$ of ADAPT. The algorithm is based on the construction of an integer-valued monotone submodular utility function g (previously used by Golovin et al. [10]) which assigns a utility value to partial assignments of f. Associated with g is an integer Q, called the goal utility. The value of the utility function on a partial assignment is equal to Q if and only if all examples in the sample that are extensions of that partial assignment have the same label. Thus aquiring enough information from tests in order to classify an example from the sample is equivalent to achieving goal utility Q. Based on properties of their utility function, they prove that their algorithm achieves an approximation factor of $O(\log Q)$, which is $O(\log m)$, with respect to the minimum average cost achieved by any decision tree achieving goal utility Q on all examples in the sample.

To achieve the $O(\log Q)$ approximation factor, Cicalese et al. use the fact that their utility function is monotone and submodular. They also exploit the property that for each variable x_i, there is a setting of x_i such that discovering that $x_i = 1$ or that $x_i = 0$ (starting from no information at all) will yield an increase of utility equal to at least 1/2 of the goal utility. In addition, because their algorithm is recursive, they need this property to hold on any induced problem that is generated once a subset of the tests have already been performed (the induced problem is defined on the untested variables, with the goal value reset to be the remaining distance to original goal value Q). A small modification of their analysis shows that the precise factor of 1/2 of goal utility is not essential to the proof of their $O(\log Q)$ bound; their $O(\log m)$ bound can still be achieved if 1/2 is replaced by a smaller constant factor α.

Deshpande et al. [8] constructed an integer-valued monotone submodular utility function g for the evaluation of linear threshold formulas. If the LTF is a k-of-n formula, then g is as follows: for

$b \in \{0,1,*\}^n$, $g(b) = k(n-k+1) - (k - (\sum_{i:b_i=1} 1))((n-k+1) - (\sum_{i:b_i=0} 1))$. The goal utility for g is $k(n-k+1)$; achieving this goal utility is equivalent to having at least k 1's in b or having at least $n-k+1$ 0's in b. It is easy to see that discovering that a bit is equal to 1, starting from no information, yields utility that is at least $1/k$ of the goal utility. This property still holds if we consider the induced problem generated once a subset of the tests have been performed. For example, if the original value of k is 4 and $n = 9$, and we have discovered one 1 and three 0's, the induced problem is to evaluate a 3-of-5 function. Discovering a new bit equal to 1 on this induced problem will still yield utility that is at least 1/4 of the distance to the new goal utility value.

To solve the Sample-BFE problem for k-of-n formulas, we run the algorithm of Cicalese et al. using the above utility function g, with only small changes. Their algorithm uses a top-down decision tree construction procedure to produce a tree consistent with the sample. The algorithm defines a special setting of each variable x_i, used in constructing the "spine" of the tree. The special setting, either $x_i = 1$ or $x_i = 0$, is the one yielding the smaller increase in their utility function, beginning from no information. We replace their utility function by our g to get the special setting of each variable. A technical complication is that we may place a variable x_i in a node in our constructed tree and find that all examples in the subsample reaching that node have the same value on x_i, meaning that one child of the node will be reached by an empty subsample (this does not happen with the utility function used by Cicalese et al.). We do not try to continue building the tree at this child. Instead, we place a specially marked leaf indicating that the remaining untested variables should be evaluated in some fixed but arbitrary order (e.g., in increasing order of index).

When k is constant, the analysis of Cicalese et al. can then be followed, with only small changes, to prove an approximation factor of $O(\log n)$, since the goal utility of g is $O(k(n-k+1))$. □

4 Non-adaptive $O(\log Q)$ approximations for Sample-BFE with unit costs

Under the assumption of unit costs, we describe a generic approach for obtaining a non-adaptive algorithm solving the Sample-BFE problem that achieves an approximation factor of $O(\log Q)$ with respect to NONADAPT, where Q is the goal value of an *assignment-feasible* utility function $g(x_1, \ldots, x_n)$ for the function $f(x_1, \ldots, x_n)$ being evaluated. Deshpande et al. [8] defined g to be assignment-feasible for f with goal value Q if g is an integer-valued monotone-submodular integer valued utility function where $g(*, \ldots, *) = 0$, and for all $b \in \{0,1,*\}^n$, $g(b) = Q$ if and only if b is either a 0-certificate or a 1-certificate of f.

The approach uses the results of Azar and Gamzu [1] for the problem of *ranking with submodular valuations*. In this problem, the input is a ground set $M = \{1, \ldots, m\}$ of elements, a collection of n monotone submodular set functions $f^1, \ldots, f^n$, where each $f^i : 2^M \to \mathbb{R}_+$, a weight for each function, and a list $Q^1, \ldots, Q^n$ of goal values for the f^i. The problem is to find a permutation of the ground set that minimizes the weighted covering time of the f^i, where f^i is considered covered when it has reached its goal utility Q^i.

Given a Sample-BFE Problem for a function f, if we have an assignment-feasible utility function g for f, with goal value Q, we can reduce the problem to an instance of ranking with submodular valuations. We do this by associating a submodular set function f^j with each example E_j in the input sample, defined on subsets of the variables of f, such that the value of f^j on a subset S is equal to the to the value of g on the partial assignment setting the variables in S according to their values on E_j (and setting all other variables to $*$). The ground set corresponds to the variables of f, and the weight associated with each f^j is 1. The goal value Q^j for each f^j is the goal value Q of g. Using the algorithm of Azar and Gamzu, the approximation factor obtained by that algorithm yields an approximation factor of $O(\log Q)$ for the Sample-BFE problem.

Given a CDNF formula, Deshpande et al. [8] showed how to construct an associated assignment-feasible utility function having a goal value of kd, where k is the number of clauses in the CNF and d is the number of terms in the DNF. Thus we have a non-adaptive algorithm solving the Sample-

BFE problem for CDNF formulas with an approximation factor of $O(\log kd)$ with respect to NON-ADAPT.

5 Open Problems

There are a number of open problems suggested by the above work. One problem is whether we can obtain the same approximation bounds for the Sample-BFE problem as for the Stochastic-BFE problem. Deshpande et al. [8] give a 3-approximation for Stochastic-BFE for linear threshold functions. Can a constant-factor approximation also be obtained for Sample-BFE? A related open problem concerns the problem of deriving adaptivity gaps for the Sample-BFE problem. More generally, it is not clear how to best formulate crisp theoretical problems whose solutions will be useful in developing machine learning classifiers that address both accuracy and prediction cost.

Acknowledgments

The authors were partially supported by NSF Grant 1217968.

References

[1] Yossi Azar and Iftah Gamzu. Ranking with submodular valuations. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11. SIAM, 2011.

[2] Yossi Azar, Iftah Gamzu, and Xiaoxin Yin. Multiple intents re-ranking. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing*, pages 669–678. ACM, 2009.

[3] Nikhil Bansal, Anupam Gupta, and Ravishankar Krishnaswamy. A constant factor approximation algorithm for generalized min-sum set cover. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, 2010.

[4] G. Bellala, S. Bhavnani, and C. Scott. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Transactions on Information Theory*, 2012.

[5] Y. Ben-Dov. Optimal testing procedure for special structures of coherent systems. *Management Science*, 1981.

[6] M.-F. Chang, W. Shi, and W. K. Fuchs. Optimal diagnosis procedures for k-out-of-n structures. *IEEE Transactions on Computers*, 39(4):559–564, April 1990.

[7] Ferdinando Cicalese, Eduardo Laber, and Aline Medeiros Saettler. Decision trees for function evaluation - simultaneous optimization of worst and expected cost. *CoRR*, abs/1309.2796, 2014.

[8] Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. In *SODA 2014*, pages 1453–1466. SIAM, 2014.

[9] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min-sum set cover. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 94–107, 2002.

[10] D. Golovin, A. Krause, and D. Ray. Near-optimal Bayesian active learning with noisy observations. In *24th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 766–774, 2010.

[11] S. Salloum. *Optimal testing algorithms for symmetric coherent systems*. PhD thesis, University of Southern California, 1979.

[12] S. Salloum and M. Breuer. An optimum testing algorithm for some symmetric coherent systems. *Journal of Mathematical Analysis and Applications*, 101(1):170 – 194, 1984.

[13] Martin Skutella and David P. Williamson. A note on the generalized min-sum set cover problem. *Operations Research Letters*, 39(6):433 – 436, 2011.