Generalized Schemata, Estimation of Distributions, and Sampling

Moshe Looks *moshe@metacog.org*

April 17, 2008

Abstract

In this paper I present some work towards estimation-of-distribution algorithms (EDAs) with expressive probabilistic models, based on an extended schema formalism. I first show how the probabilistic models built by current EDAs, such as the hierarchical Bayesian Optimization Algorithm (hBOA), may be understood to be processing schemata, and relationships amongst schemata, defined according to Holland's original formalism [2]. I then describe a more general schema formalism, and a new family of EDAs that may be based on it.

1 Holland Schemata and EDAs

The most fundamental formalism of evolutionary computation is John Holland's schema: a string of length n drawn from the alphabet $\{?, 0, 1\}$, in the context of an optimization problem over the domain $\{0, 1\}^n$. For $n = 4$, for example, the schema 0?1? *matches* the four solutions 0010, 0011, 0110, and 0111. A number of generalizations of such schemata have been made, most notably to schemata defined over trees and variable-length strings, by Poli and Langdon [4], along with some wonderfully (terribly?) complex math (schema theorems).

The Bayesian Optimization Algorithm (BOA)čiteBOA represents problem decompositions for n-bit optimization problems as Bayesian Networks with one variable per bit. What does the network structure actually *mean*, in terms of the individual bits in the optimization problem? Lets first consider a network with no edges - the ith node in the network (x_i, where $0 \leq i < n$) may be understood to model the *competition* between the schemata $?^{i-1}0?^{n-i}$ and $?^{i-1}1?^{n-i}$. These schemata are considered to be in competition because the are mutually exclusive - one and only one of them must appear in every complete solution. An empty Bayesian Network corresponds to assuming complete independence amongst our variables, just as only considering first-order schemata (schemata with exactly one 0 or 1) assumes that the optimal value for any bit in our optimization problem may be found independently of all others.

What if we add the edge (x_i, x_j)? We are now considering statistics for all (four) second-order schemata involving these two variables. Assuming a three-bit optimization problem with $i = 0$ and $j = 1$, where x_2 remains independent, the full set of schemata we are tracking will be:

$$\{0??, 1??, 00?, 01?, 10?, 11?, ??0, ??1\}. \tag{1}$$

Note that the directionality of the edge is significant; because the edge is from x_0 to x_1, we still track the first-order statistics for x_0, while those for x_1 are discarded in favor of the second-order schemata listed above. If we add the edge (x_2, x_0), the schemata considered will be:

$$\{0?0, 0?1, 1?0, 1?1, 00?, 01?, 10?, 11?, ??0, ??1\}. \tag{2}$$

If we instead add the edge (x_2, x_1), the schemata considered will be:

$$\{0??, 1??, 000, 001, 010, 011, 100, 101, 110, 111, ??0, ??1\}. \quad (3)$$

Since we are now tracking all of the third-order schemata in a three-bit optimization problem, the network is not doing a very good job of modeling (i.e. compressing) the data - the Bayesian Network representation is of course most effective when the maximal order of the schemata we need to track (k) is much lower than the size of the problem (n), but non-zero.

So one way to look at the probabilistic models learned by the BOA is as specifying a particular set of low-order schemata that are (hopefully) sufficient to for quickly finding a solution to our problem. If our set of schemata is larger than strictly necessary, we will need to consider more solutions (intuitively the number solutions that must be considered is proportionate to the number of schemata in our set). If our set of schemata is too small, however (i.e. considers schema of too low an order), we will only ever reach a local optimum in our search, unless we are willing to examine on the order of n^k solutions, where k is the *actual* order of the largest schema in the problem.

Yet adding edges is expensive - if x_j has k incoming edges, adding a new edge (x_i, x_j) will increase the number of schemata we must consider (i.e. the size of our conditional probabilities table) by 2^k.[1] To address primarily this issue, the hierarchical BOA (hBOA) [8] was developed, which uses a finer-grained representation than a Bayesian Network - a Bayesian Network with local structure (i.e. decision graphs or trees). The hBOA allows us to take smaller steps in model-building, and add accordingly fewer schemata at a time to be maintained.

Consider the three-node network above, with the single edge (x_0, x_1). With the hBOA, we can say that x_1 is also dependent on x_2, but only when $x_0 = 0$. This will give us the schemata:

$$\{0??, 1??, 000, 001, 010, 011, 10?, 11?, ??0, ??1\}. \quad (4)$$

Even though the order of the model has increased from two to three, we have only increased the number of schema we must consider by two, not four. For complex models, the savings from applying the hBOA vs. the BOA can be huge. With decision graphs we can even apply *merge* operations that decrease the number of schemata maintained. If we decide that if $\{x_0 = 0, x_2 = 1\}$ should be merged with $x_0 = 1$, we get the schemata:

$$\{0??, 1??, 000, 010, ?0?, ?1?, ??0, ??1\}.[2] \quad (5)$$

At this point one might ask, why bother with any kind of model structure at all? If, at the end of the day, we are just manipulating sets of schemata, why not just maintain such a set in memory, allowing for the finest-grained manipulation possible? I can see two primary reasons for using a probabilistic model:

1. There are well-developed and tractable heuristics for learning probabilistic models from data.

2. Sampling from the probability distribution defined by such a model (plus marginals) is simple and fast.

One can in fact design an EDA based on directly learning a set of "useful distinguishing schemata" and using them to generate new instances (this is a case of the Learnable Evolution Model approach [7]). But let us return our attention to the schemata themselves.

[1] For many problems of interest, $n \cdot 2^k \ll n^k$. This is, roughly speaking, the set of nearly decomposable problems.

[2] While merge operations aren't especially useful in practice, they are a nice illustration of increased expressiveness.

2 The Difficulty with non-Holland Schemata

Lets consider a very simple optimization problem where the goal is to discover a string of length n that contains k consecutive ones. Unfortunately, our schema formalism is not very useful in modeling this state of affairs - to describe the optima, we will need $n - k + 1$ schemata, one per optimum. It seems quite natural to introduce, in addition to ? for a single "don't-care" value, the symbol $*$ for "any number of don't-care values". The optima for this problem may now be described with a single schema, $*1^k*$.

But can we introduce a corresponding EDA, together with model-building and sampling procedures, for these new schemata? Unfortunately, this appears to be rather difficult (I give it a shot in [6]). What makes modeling and sampling so easy for EDAs based on classical schemata is the way that they naturally carve up the search space into disjoint subsets:

1. First-order Holland schemata are closed under complement.
2. Holland schemata that specify values for disjoint variables may be specified independently of each other (e.g. 1? and ?0 never conflict).
3. A set of Holland schemata may be easily sampled that fully specify a solution without overspecifying it.

In contrast:

1. The complement of a first-order extended schemata will not usually be an extended schemata.
2. Extended schemata with $*$s in them will often conflict with each other.
3. The same set of extended schemata can both overspecify and underspecify solutions, depending on how they are applied (consider $*11*$ and $*00*$ in a three-bit optimization problem).

3 Schema-based Sampling

What I propose as a solution to this difficulty is a new sampling scheme that may operate generically on any set of schemata, Holland or otherwise. How these schemata are acquired is not specified; it may be via the BOA or hBOA, Ben Goertzel's proposed PLEASURE ("Program Learning via Ensemble Analysis and SUbstitution of Repeated Entities.") algorithm [1], the feature selection scheme from the feature-based BOA [6], etc., or some combination thereof. The way this new procedure overcomes the difficulties of over- and under-specification is by applying the model in a local rather than a global context; rather than generating entire solutions *de novo* according to our model, we will use it to propose transformations to existing solutions.

procedure IMPROVE(Solution X)
  $SSets \leftarrow 2^{SelectSchemata(X)}$
  Remove conflicting sets of schemata from $SSets$
  while $SSets$ is non-empty **do**
    $S \leftarrow$ the set in $SSets$ with highest expected utility
    $X' \leftarrow X$ transformed to express the schema in S
    score X' and update expected utilities accordingly
    if X' improves on X or $Noise(X, X')$ **then** $Improve(X')$
    end if
  end while
end procedure

By "conflicting sets" I mean sets of schemata that are, taken in conjunction, not present in any single solutions. By "highest expected utility" I mean the set that, based on the system's internal model, has the highest likelihood, when used to transform X, of improving its score on the objective function.

This procedure bears a strong resemblance to and is inspired by the Building Block Hillclimber (BBHC) [3]. What this procedure does not specify is how schemata are proposed as worth investigating ($SelectSchemata$), how the existing solution is transformed to express the new set of schemata, the noise function ($Noise$), how expected utilities are calculated and updated, and termination criterion. The procedure has a number of desirable properties:

1. It is a generalization simulated annealing.
2. It is a generalization of iterative deepening search.
3. It is a generalization of best-first search.
4. It can operate on any type of schemata (or combination of types) - all that we need is a predicate indicating whether a given schema is present or absent in a solution, a predicate indicating whether a set of schemata may be applied in conjunction to a given solution, and a procedure for transforming a solution to contain a set of schemata that are not already present.
5. Under-specification is not a problem because we are basing our transformations on a solution that is already fully specified.
6. Over-specification is not a problem because we explicitly remove from consideration those combinations of schemata than conflict.
7. If we consider schemata that are negations of features, we can also learn from the commonalities in *bad* solutions that have been observed (as proposed in PLEASURE, for instance).
8. It supports soft chunking by allowing conflicting schemata at different scales to be considered (e.g., 1?001?? together with ??00???).
9. Relatedly, it facilitates further movement in the direction of fine-grained search - just as hBOA allows us to take smaller steps than BOA, the new approach allows us to make even smaller modifications to our model.
10. It introduces an important *level of indirection* - search does not act directly on solution space, but views and transforms solutions only via schemata that are deemed useful, based potentially on both prior knowledge and on learning.

There are a number of ways to consider augmenting this search procedure. Two that immediately spring to mind are backtracking (arguably a well-tuned $Noise$ function obviates this) and diversity management (explicitly maintaining multiple promising solution in parallel to search from, as in MOSES [5], based on extensional and/or intensional similarity comparisons amongst solutions).

References

[1] B. Goertzel. The pleasure algorithm. *http://groups.google.com/group/opencog/files*, 2008.

[2] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[3] David Iclanzan and Dan Dumitrescu. Overcoming hierarchical difficulty by hill-climbing the building block structure. In *Genetic and evolutionary computation conference*, 2007.

[4] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[5] M. Looks. *Competent Program Evolution*. PhD thesis, Washington University in St. Louis, 2006.

[6] M. Looks. Levels of abstraction in modeling and sampling: The feature-based Bayesian optimization algorithm. In *Genetic and Evolutionary Computation Conference*, 2006.

[7] R. S. Michalski. LEARNABLE EVOLUTION MODEL: Evolutionary processes guided by machine learning. *Machine Learning*, 2000.

[8] M. Pelikan and D. E. Goldberg. A hierarchy machine: Learning to optimize from nature and humans. *Complexity*, 2003.