# Integrating Feature Selection into Program Learning

Ben Goertzel[1,2], Nil Geisweiller[1], Cassio Pennachin[1], and Kaoru Ng[2]

[1] Novamente LLC
[2] Biomind LLC

**Abstract.** In typical practical applications of automated program learning, the scope of potential inputs to the programs being learned are narrowed down during a preliminary "feature selection" step. However, this approach will not work if one wishes to apply program learning as a key component of an AGI system, because there is no generally applicable feature selection heuristic, and in an AGI context one cannot assume a human cleverly tuning the feature selection heuristics to the problem at hand. A novel technique, LIFES (Learning-Integrated Feature Selection), is introduced here, to address this problem. In LIFES, one integrates feature selection into the learning process, rather than doing feature selection solely as a preliminary stage to learning. LIFES is applicable relatively broadly, and is especially appropriate for any learning problem possessing properties identified here as "data focusable" and "feature focusable. It is also applicable with a wide variety of learning algorithms, but for concreteness is presented here in the context of the MOSES automated program learning algorithm. To illustrate the general effectiveness of LIFES, example results are given from applying MOSES+LIFES to gene expression classification. Application of LIFES to virtual and robotic agent control is also discussed.

## 1 Introduction

Machine learning, for example automated program learning, is an example of an algorithmic approach that is very useful in a narrow AI context, and also shows great promise as a component of AGI systems. However, in order to make program learning and other ML algorithms useful for AGI, some changes must be made to the way they are used. Here we explore one of these changes: eliminating the separation between feature selection and learning.

In the typical workflow of applied machine learning, one begins with a large number of features, each applicable to some or all of the entities one wishes to learn about; then one applies some feature selection heuristics to whittle down the large set of features into a smaller one; then one applies a learning algorithm to the reduced set of features. The reason for this approach is that the more powerful among the existing machine learning algorithms tend to get confused when supplied with too many features. The problem with this approach is that sometimes one winds up throwing out potentially very useful information during the feature selection phase.

The human mind, as best we understand it, does things a bit differently. It does seem to perform operations analogous to feature selection, and operations analogous to the application of a machine learning algorithm to a reduced feature set – but then it also involves feedback from these "machine learning like" operations to the "feature selection like" operations, so that the intermediate results of learning can cause the introduction into the learning process of features additional to those initially selected, thus allowing the development of better learning results.

Compositional spatiotemporal deep learning (CSDLN) architectures [1] like HTM [2] or DeSTIN [3] incorporate this same sort of feedback. The lower levels of such an architecture, in effect, carry out "feature selection" for the upper levels – but then feedback from the upper to the lower levels also occurs, thus in effect modulating the "feature selection like" activity at the lower levels based on the more abstract learning activity on the upper levels. However, such CSDLN architectures are specifically biased toward recognition of certain sorts of patterns – an aspect that may be considered a bug or a feature of this class of learning architecture, depending on the context. For visual pattern recognition, it appears to be a feature, since the hierarchical structure of such algorithms roughly mimics the architecture of visual cortex. For automated learning of computer programs carrying out symbolic tasks, on the other hand, CSDLN architectures are awkward at best and probably generally inappropriate. For cases like language learning or abstract conceptual inference, the jury is out.

The question addressed here is: how to introduce an appropriate feedback between feature selection and learning in the case of machine learning algorithms with general scope and without explicit hierarchical structure. We introduce a specific technique enabling this, which we call LIFES, short for Learning-Incorporated Feature Selection. We argue that LIFES is particularly applicable to learning problems that possess the conjunction of two properties that we call data focusability and feature focusability. We illustrate LIFES in the context of the MOSES automated program learning algorithm [4], describing a specific incarnation of the LIFES technique that does feature selection repeatedly during the MOSES learning process, rather than just doing it initially prior to MOSES learning.

MOSES is a probabilistic evolutionary learning algorithm, that works via evolving a meta-population of "demes" (somewhat similar to "islands" in GA/GP), and within each deme evolving a population of programs in the local neighborhood of the deme's "exemplar" program. A successful deme spawns new demes with new exemplars, varying on the successful programs found in that deme. Each program is represented as a tree, and each node in the tree is considered as a "knob" that can be turned to different settings, thus producing a variant program with something different at that node. (For instance, a program tree node containing a Boolean operator would be considered as a knob that could be turned to other settings corresponding to other Boolean operators.) An unsuccessful deme is removed from the meta-population. What happens inside a deme is a combination of local search, crossover, and BOA-style probabilistic modeling. MOSES has been used for a variety of practical commercial applications , primarily in the area of data classification. It has also been used in a proto-AGI context, to enable the OpenCog integrative AGI architecture (which incorporates MOSES) to control a virtual dog learning new tricks based on imitation and reinforcement learning [5].

In the virtual dog application, MOSES learned virtual-agent control programs, based on a feature set including the perceptions and actions of the virtual dog in the virtual world. This was a fascinating but quite complex application. Here, to illustrate the effectiveness of LIFES in a MOSES context in a simple way, we give some example results from the application of MOSES-LIFES to supervised classification of genomic data.

We will not review MOSES in any detail here, as it has been covered extensively in previous publications, see [6] for an overview. For a discussion of earlier applications of MOSES to genomic data classification, see [7].

### 1.1 Machine Learning, Feature Selection and AGI

While the example results presented here are drawn from a "narrow AI" application, the key motivation for the development of LIFES is the application of automated program learning to general intelligence. The relation between feature selection and machine learning appears an excellent example of the way that, even when the same basic technique is useful in both narrow AI and AGI, the method of utilization is often quite different. In most applied machine learning tasks, the need to customize feature selection heuristics for each application domain (and in some cases, each particular problem) is not a major difficulty. This need does limit the practical utilization of machine learning algorithms, because it means that many ML applications require an expert user who understands something about machine learning, both to deal with feature selection issues and to interpret the results. But it doesn't stand in the way of ML's fundamental usability. On the other hand, in an AGI context, the situation is different, and the need for human-crafted, context-appropriate feature selection does stand in the way of the straightforward insertion of most ML algorithms into an integrative AGI systems.

For instance, in the OpenCog integrative AGI architecture that we have co-architected [8], the MOSES automated program learning algorithm plays a key role. It is OpenCog's main algorithm for acquiring procedural knowledge, and is used for generating some sorts of declarative knowledge as well. However, when MOSES tasks are launched automatically via the OpenCog scheduler based on an OpenCog agent's goals, there is no opportunity for the clever choice of feature selection heuristics based on the particular data involved. And crude feature selection heuristics based on elementary statistics, are often insufficiently effective, as they rule out too many valuable features (and sometimes rule out the most critical features). In this context, having a variant of MOSES that can sift through the scope of possible features in the course of its learning is very important.

An example from the virtual dog domain pursued in [5] would be as follows. Each procedure learned by the virtual dog combines a number of different actions, such as "step forward" "bark" "turn around" "look right" "lift left front leg,“ etc. In the virtual dog experiments done previously, the number of different actions permitted to the dog was less than 100, so that feature selection was not a major issue. However, this was an artifact of the relatively simplistic nature of the experiments conducted. For a real organism, or for a robot that learns its own behavioral procedures (say, via a deep learning algorithm) rather than using a pre-configured set of “animated“ behaviors, the number of possible behavioral procedures to potentially be combined using a MOSES-learned program may be very large. In this case, one must either use some crude feature selection heuristic, have a human select the features, or use something like the LIFES approach described here. LIFES addresses a key problem in moving from the relatively simple virtual dog work done before, to related work with virtual agents displaying greater general intelligence.

As an other example, suppose an OpenCog-controlled agent is using MOSES to learn procedures for navigating in a dynamic environment. The features that candidate navigation procedures will want to pay attention to, may be different in a well-lit environment than in a dark environment. However, if the MOSES learning process is being launched internally via OpenCog's goal system, there is no opportunity for a human to adjust the feature selection heuristics based on the amount of light in the environment. Instead, MOSES has got to figure out what features to pay attention to all by itself. LIFES is designed to allow MOSES (or other comparable learning algorithms) to do this.

So far we have tested LIFES in genomics and other narrow-AI application areas, as a way of initially exploring and validating the technique. As our OpenCog work proceeds, we will explore more AGI-oriented applications of MOSES-LIFES. This will be relatively straightforward on a software level as MOSES is fully integrated with OpenCog.

## 2 Data- and Feature- focusable Learning Problems

Learning-integrated feature selection as described here is applicable across multiple domain areas and types of learning problem – but it is not completely broadly applicable. Rather it is most appropriate for learning problems possessing two properties we call data focusability and feature focusability. While these properties can be defined with mathematical rigor, here we will not be proving any theorems about them, so we will content ourselves with semi-formal definitions, sufficient to guide practical work.

We consider a fitness function $\Phi$, defined on a space of programs $f$ whose inputs are features defined on elements of a reference dataset $S$, and whose outputs lie in the interval $[0, 1]$. The features are construed as functions mapping elements of $S$ into $[0, 1]$. Where $F(x) = (F^1(x). \dots , F^n(x))$ is the set of features evaluated on $x \in S$, we use $f(x)$ as a shorthand for $f(F(x))$.

We are specifically interested in $\Phi$ which are “data focusable” in the sense that, for a large number of highly fit programs, there is a non-trivially large set $S_f$that is subset of $S$and such that the fitness score of $f$ depends only on the restriction of $f$ to $S_f$. What it is exactly to say that f depends only on its restriction to $f$ might be defined in various sensible ways, for example saying that the fitness score of $f$ would be the same as the fitness score of the program $g$ that behaves just like $f$ in $S_f$ but always returns value zero outside of $S_f$.

One important case is where $\Phi$ is "property-based" in the sense that each element $x \in S$ has some Boolean or numeric property $p(x)$, and the fitness function $\Phi(f)$ rewards $f$ for predicting $p(x)$ given $x$ for $x \in S_f$, where $S_f$ is **some** non-trivial subset of $S$. For example, each element of $S$ might belong to some category, and the fitness function might represent the problem of placing elements of $S$ into the proper category – but with the twist that $f$ gets rewarded if it accurately places some subset $S_f$ of elements in $S$ into the proper category, even if it has nothing to say about all the elements in $S$ but not $S_f$.

For instance, consider the case where $S$ is a set of images. Suppose the function $p(x)$ indicates whether the image $x$ contains a picture of a cat or not. Then, a suitable fitness function $\Phi$ would be one measuring whether there is some non-trivially large set of images $S_f$ so that if $x \in S_f$, then $f$ can accurately predict whether $x$ contains a picture of a cat or not. A key point is that the fitness function doesn't care whether $f$ can accurately predict whether $x$ contains a picture of a cat or not, for $x$ outside $S_f$.

Or, consider the case where $S$ is a discrete series of time points, and $p(x)$ indicates the value of some quantity (say, a person's EEG) at a certain point in time. Then a suitable fitness function $\Phi$ might measure whether there is some non-trivially large set of time-points $S_f$ so that if $x \in S_f$, then $f$ can accurately predict whether $x$ will be above a certain level $L$ or not.

Finally, in addition to the property of data-focusability introduced above, we will concern ourselves with the complementary property of "feature-focusability." This means that, while the elements of $S$ are each characterized by a potentially large set of features, there are many highly fit programs $f$ that utilize only a small subset of this large set of features. The case of most interest here is where there are various highly fit programs $f$, each utilizing a different small subset of the overall large set of features. In this case one has (loosely speaking) a pattern recognition problem, with approximate solutions comprising various patterns that combine various different features in various different ways. For example, this would be the case if there were many different programs for recognizing pictures containing cats, each one utilizing different features of cats and hence applying to different subsets of the overall database of images.

There may, of course, be many important learning problems that are neither data nor feature focusable. However, the LIFES technique presented here for integrating feature selection into learning is specifically applicable to objective functions that are both data and feature focusable. In this sense, the conjunction of data and feature focusability appears to be a kind of "tractability" that allows one to bypass the troublesome separation of feature selection and learning, and straightforwardly combine the two into a single integrated process. Being property-based in the sense described above does not seem to be necessary for the application of LIFES, though most practical problems do seem to be property-based.

## 3 Integrating Feature Selection into Learning

The essential idea proposed here is a simple one. Suppose one has a learning problem involving a fitness function that is both data and feature focusable. And suppose that, in the course of learning according to some learning algorithm, one has a candidate program $f$, which is reasonably fit but merits improvement. Suppose that $f$ uses a subset $F_f$ of the total set $F$ of possible input features. Then, one may do a special feature selection step, customized just for $f$. Namely, one may look at the total set $F$ of possible features, and ask **which features or small feature-sets display desirable properties on the set $S_f$**. This will lead to a new set of features potentially worthy of exploration; let's call it $F'_f$. We can then attempt to improve $f$ by creating variants of $f$ introducing some of the features in $F'_f$ – either replacing features in $F_f$ or augmenting them. The process of creating and refining these variants will then lead to new candidate programs $g$, potentially concentrated on sets $S_g$ different from $S_f$, in which case the process may be repeated. This is what we call LIFES – Learning-Integrated Feature Selection.

As described above, the LIFES process is quite general, and applies to a variety of learning algorithms – basically any learning algorithm that includes the capability to refine a candidate solution via the introduction of novel features. The nature of the "desirable properties" used to evaluate candidate features or feature-sets on $S_f$ needs to be specified, but a variety of standard techniques may be used here (along with more advanced ideas) – for instance, in the case where the fitness function is defined in terms of some property mapping $p$ as describe, above, then given a feature $F^i$, one can calculate the mutual information of $F^i$ with $p$ over $S_f$. Other measures than mutual information may be used here as well.

The LIFES process doesn't necessarily obviate the need for up-front feature selection. What it does, is prevent up-front feature selection from limiting the ultimate feature usage of the learning algorithm. It allows the initially selected features to be used as a rough initial guide to learning – and for the candidates learned using these initial features, to then be refined and improved using additional features chosen opportunistically along the learning path. In some cases, the best programs ultimately learned via this approach might not end up involving any of the initially selected features.

To make these ideas more concrete, one must pursue them in the context of some specific learning algorithm; in the remainder of this paper we shall do so using the MOSES algorithm.

## 4 Integrating Feature Selection into MOSES Learning

The application of the LIFES process in the MOSES context is relatively straightforward. Quite simply, given a reasonably fit program $f$ produced within a deme, one then isolates the set $S_f$ on which $f$ is concentrated, and identifies a set $F'_f$ of features within $F$ that displays desirable properties relative to $S_f$. One then creates a new deme $f^*$, with exemplar $f$, and with a set of potential input features consisting of $F_f \cup F'_f$.

What does it mean to create a deme $f^*$with a certain set of “potential input features“ $F_f \cup F'_f$? Abstractly, it means that $F_{f^*} = F_f \cup F'_f$. Concretely, it means that the knobs in the deme’s exemplar $f^*$ must be supplied with settings corresponding to the elements of $F_f \cup F'_f$. The right way to do this will depend on the semantics of the features.

For instance, it may be that the overall feature space $F$ is naturally divided into groups of features. In that case, each new feature $F^i$ in $F'_f$ would be added, as a potential knob setting, to any knob in $f$ corresponding to a feature in the same group as $F^i$.

On the other hand, if there is no knob in $f$ corresponding to features in $F^i$’s knob group, then one has a different situation, and it is necessary to “mutate“ $f$ by adding a new node with a new kind of knob corresponding to $F^i$, or replacing an existing node with a new one corresponding to $F^i$.

## 5 Application to Genomic Data Classification

To illustrate the effectiveness of LIFES in a MOSES context, we now briefly describe an example application, in the genomics domain. The application of MOSES to gene expression data is described in more detail in [7], and is only very briefly summarized here. To obtain the results summarized here, we have used MOSES, with and without LIFES, to analyze two different genomics datasets: an Alzheimers SNP (single nucleotide polymorphism) dataset [9] previously analyzed using ensemble genetic programming [10] . The dataset is of the form “Case vs. Control” where the Case category consists of data from individuals with Alzheimers and Control consists of matched controls. MOSES was used to learn Boolean program trees embodying predictive models that take in a subset of the genes in an individual, and output a Boolean combination of their discretized expression values, that is interpreted as a prediction of whether the individual is in the Case or Control category. Prior to feeding them into MOSES, expression values were first Q-normalized, and then discretized via comparison to the median expression measured across all genes on a per-individual basis (1 for greater than the median, 0 for less than). Fitness was taken as precision, with a penalty factor restriction attention to program trees with recall above a specified minimum level.

These study was carried out, not merely for testing MOSES and LIFES, but as part of a practical investigation into which genes and gene combinations may be the best drug targets for Alzheimers Disease. The overall methodology for the biological investigation, as described in [11], is to find a (hopefully diverse) ensemble of accurate classification models, and then statistically observe which genes tend to occur most often in this ensemble, and which combinations of genes tend to co-occur most often in the models in the ensemble. These most frequent genes and combinations are taken as potential therapeutic targets for the Case category of the underlying classification problem (which in this case denotes inflammation). This methodology has been biologically validated by follow-up lab work in a number of cases; see e.g. [12] where this approach resulted in the first evidence of a genetic basis for Chronic Fatigue Syndrome. A significant body of unpublished commercial work along these lines has been done by Biomind LLC [http://biomind.com] for its various customers.

Comparing MOSES-LIFES to MOSES with conventional feature selection, we find that the former finds model ensembles combining greater diversity with greater precision, and equivalent recall. This is because conventional feature selection eliminates numerous genes that actually have predictive value for the phenotype of inflammation, so that MOSES never gets to see them. LIFES exposes MOSES to a much greater number of genes, some of which MOSES finds useful. And LIFES enables MOSES to explore this larger space of genes without getting bollixed by the potential combinatorial explosion of possibilities.

**Table 1.** Impact of LIFES on MOSES classification of Alzheimers Disease SNP data. Fitness function sought to maximize precision consistent with a constraint of precision being at least 0.5. Precision and recall figures are average figures over 10 folds, using 10-fold cross-validation. The results shown here are drawn from a larger set of runs, and are selected according to two criteria: best training precision (the fair way to do it) and best test precision (just for comparison). We see that use of LIFES increases precision by around 3% in these tests, which is highly statistically significant according to permutation analysis.

| Algorithm | Train. Precision | Train. Recall | Test Precision | Test Recall | |
|---|---|---|---|---|---|
| MOSES | .81 | .51 | .65 | .42 | best training precision |
| MOSES | .80 | .52 | .69 | .43 | best test precision |
| MOSES-LIFES | .84 | .51 | .68 | .38 | best training precision |
| MOSES-LIFES | .82 | .51 | .72 | .48 | best test precision |

## 6 Conclusion

We have described LIFES, a novel approach to overcoming the separation between feature selection and learning in a fairly general context. We have described in detail the application of LIFES to enhance the MOSES probabilistic evolutionary program learning algorithm, and given an example of MOSES-LIFES integration in the domain of genomics data classification.

The genomics example shows that LIFES makes sense and works in the context of MOSES, broadly speaking. It seems very plausible that LIFES will also work effectively with MOSES in an integrative AGI context, for instance in OpenCog deployments where MOSES is used to drive procedure learning, with fitness functions supplied by other OpenCog components. However, the empirical validation of this plausible conjecture remains for future work.

**Acknowledgement.** We would like to thank an anonymous referee for fixing our prior definition of "data-focused" fitness functions.

## References

1. Goertzel, B.: Perception processing for general intelligence: Bridging the symbolic/Subsymbolic gap. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 79–88. Springer, Heidelberg (2012)
2. Hawkins, J., Blakeslee, S.: On Intelligence. Brown Walker (2006)
3. Arel, I., Rose, D., Coop, R.: Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. In: Proc. AAAI Workshop on Biologically Inspired Cognitive Architectures (2009)
4. Looks, M.: Competent Program Evolution. PhD Thesis, Computer Science Department, Washington University (2006)
5. Goertzel, B., Pennachin, C., et al.: An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life. In: Proc. of the First Conf. on AGI. IOS Press (2008)
6. Looks, M., Goertzel, B.: Program representation for general intelligence. In: Proc. of AGI 2009 (2009)
7. Looks, M.: Scalable estimation-of-distribution program evolution. In: Genetic and Evolutionary Computation Conference (2007)
8. Goertzel, B., et al.: The cogprime architecture for embodied artificial general intelligence. In: Proceedings of IEEE Symposium on Human-Level AI, Singapore (2013)
9. Reiman, E.M., et al.: Gab2 alleles modify alzheimer's risk in apoe e4 carriers. Neuron 54(5) (2007)
10. Coelho, L., Goertzel, B., Pennachin, C., Heward, C.: Classifier ensemble based analysis of a genome-wide snp dataset concerning late-onset alzheimer disease. In: Proceedings of 8th IEEE International Conference on Cognitive Informatics (2009)
11. Goertzel, B., Coelho, L., Pennachin, C., Mudada, M.: Identifying Complex Biological Interactions based on Categorical Gene Expression Data. In: Proceedings of Conference on Evolutionary Computing, Vancouver, CA (2006)
12. Goertzel, B., et al.: Combinations of single nucleotide polymorphisms in neuroendocrine effector and receptor genes predict chronic fatigue syndrome. Pharmacogenomics (2005)