Linear Combinations of Features as Leaf Nodes in Symbolic Regression*

Jan Žegklitz
Czech Technical University in Prague, CIIRC
Jugoslávských partyzánů 1580/3
Prague 6, Czech Republic 160 00
zegkljan@fel.cvut.cz

Petr Pošík
Czech Technical University in Prague, FEE
Technická 1902/2
Prague 6, Czech Republic 166 27
petr.posik@fel.cvut.cz

ABSTRACT

We propose a new type of leaf node for use in Symbolic Regression (SR) that performs linear combinations of feature variables (LCF). LCF's weights are tuned using a gradient method based on back-propagation algorithm known from neural networks. Multi-Gene Genetic Programming (MGGP) was chosen as a baseline model. As a sanity check, we experimentally show that LCFs improve the performance of the baseline on a rotated toy SR problem. We then perform a thorougher experimental study on a number of artificial and real-world SR benchmarks. The usage of LCFs in MGGP statically improved the results in 5 cases out of 9, while it worsen them in only a single case.

CCS CONCEPTS

•Computing methodologies → Supervised learning by regression; Genetic programming;

KEYWORDS

genetic programming, symbolic regression

ACM Reference format:
Jan Žegklitz and Petr Pošík. 2017. Linear Combinations of Features as Leaf Nodes in Symbolic Regression. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3067695.3076009

1 INTRODUCTION

Symbolic regression (SR) is an inductive learning task with the goal to find a model in the form of a symbolic mathematical expression that fits the available training data. SR task is usually solved by Genetic Programming (GP) [5]. Recently, several methods emerged [1, 2, 6, 8] that explicitly evolve models in a form of (possibly regularized) "top-level" linear combinations of evolved complex features. Such models can be learned much faster since the evolution does not have to deal with the linear parts.

In some SR tasks, the underlying function could be modeled more easily if we had access to a suitable rotation of the feature space, or to suitable projections of the features. Such linear combinations of original features can be evolved in virtually any SR system that allows for numeric constants, but they must be tuned by mutation, and the linear combinations must be constructed via structural manipulation operators.

In this article we explore the possibility of using explicit linear combinations of features at the bottom of the evolved expression trees. These are added to the original features and can then be non-linearly combined by evolution. We have chosen Multi-Gene Genetic Programming (MGGP) [3, 8] as the base algorithm for the research as it is very close to regular GP but uses top-level linear combinations to speed up the search.

2 LINEAR COMBINATIONS OF FEATURES

We introduce a new type of leaf node – a Linear Combination of Features (LCF). This type of node is similar to a leaf node representing a variable, or feature. However, while an ordinary feature-node evaluates simply to the value of that feature, a LCF node evaluates to a linear combination of all the features present in the solved problem, i.e. $\mathrm{LCF}(\mathbf{x}) = w_0 + w_1 x_1 + \ldots w_n x_n$.

The LCFs effectively perform affine transformations of the feature space and we argue that they can provide more effective tools to deal with e.g. rotated functions and, in general, provide more flexibility to the GP algorithm.

The initialization of the weights is based on the idea that, at the start, there is no feature space transformation happening: only a single multiplicative weight of the LCF (corresponding to a single input variable) is set to 1, the rest is set to 0.

In order to be of any use, the weights of LCF nodes must be modified during the evolution. We use a *gradient-based tuning* approach. Since the structure of the expressions as well as the cost function is known, it is possible to compute the gradient of the expressions w.r.t. the weights. We use an approach fundamentally identical to the one used in neural networks – error backpropagation technique[1] [7]. When the individual partial derivatives are known, any first-order update method can be used to modify the weights to produce a more fit expression. We use the iRprop$^-$ update mechanism [4].

Other configurations. We also tested other configurations with different LCF initialization, weights tuning by mutation, and some restrictions imposed on the LCF nodes (see [9]). However, the

*A detailed treatment of the topic can be found at [9].

GECCO '17 Companion, Berlin, Germany
 978-1-4503-4939-0/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3067695.3076009

[1] When we use the term "(error) backpropagation", we mean only the procedure of determining the values of the partial derivatives and not the update mechanism.

initialization described above, the gradient-based tuning, and unrestricted LCF nodes outperformed the other configurations.

3 EXPERIMENTAL EVALUATION

To assess the benefit of our proposal, we evaluated it on two series of benchmarks. The first series consists of a toy problem – a simple sigmoid function applied along only a single dimension in multi-dimensional space – and its rotated version. The purpose of this test is to determine whether and how much are LCFs beneficial in the rotated environment. The expected result is that the baseline MGGP algorithm will be less successful on the rotated version while the LCF-enabled algorithm will take advantage of its ability to learn feature space transformations and hence be more successful.

The second series of tests involves 9 more complex artificial and real-world benchmarks and it aims to provide performance comparison of our proposals. Detailed description of the datasets can be found in the supplementary material.

The results on the toy problems, as can be seen in Table 1, have clearly shown that LCFs enable the algorithm to deal with the rotated feature space, while in the non-rotated case there was no difference (across a range of dimensions). This shows that the LCFs provide additional flexibility while not harming the algorithm when they are not needed.

Table 1: Test-set R^2 (1 means an ideal fit) and the statistical significance of the result on the toy problems. The statistically significantly better result has asterisk (*).

problem	baseline		LCFs		statistically better
	median	$\frac{\max}{\min}$	median	$\frac{\max}{\min}$	
non-rotated	1	$\frac{1}{1}$	1	$\frac{1}{1}$	-
rotated	0.991	$\frac{0.996}{0.912}$	1	$\frac{1}{1}$	LCFs

The results on the artificial and real-world benchmarks, as can be seen in Figure 1, have shown that while the LCFs are not always beneficial, overall they provide an improvement over the baseline.

Full results for both sets of problems can be found in the supplementary material and in [9].

4 CONCLUSIONS

In this article we presented a new type of leaf node for use in SR – linear combination of feature variables – used in the baseline algorithm of MGGP. We tested the proposal on two sets of benchmarks: toy problems specifically designed to test the ability of LCFs to provide feature space transformations, and a set of artificial and real-world benchmarks to provide a view on the overall performance.

The toy problems showed that LCFs are capable of handling existing feature space transformation and enable the algorithm to achieve better results. The artificial and real-world problems have shown that, overall, LCFs are beneficial.

ACKNOWLEDGMENTS

This work was supported by the Czech Science Foundation project Nr. 15-22731S with a student support from the Grant Agency of the Czech Technical University in Prague, grant No. SGS17/093/OHK3/1T/13. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

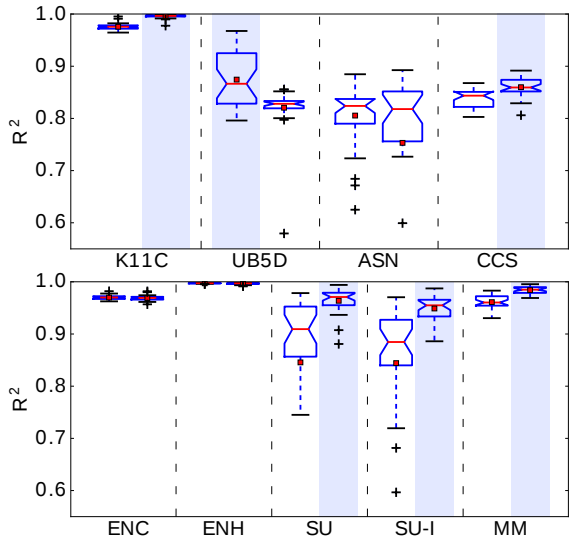

Figure 1: Test-set R^2 (1 is ideal fit) on all datasets. The left boxplots are the baseline, the right ones are for LCFs. The highlighted boxplots signify that the corresponding algorithm is statistically significantly better than the other one from the pair. Note that outliers at -0.719 for LCFs on ASN, at -0.664 for baseline on SU and at 0.175 for baseline on MM are not shown.

REFERENCES

[1] Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. 2014. Multiple Regression Genetic Programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14)*. ACM, New York, NY, USA, 879–886. DOI: http://dx.doi.org/10.1145/2576768.2598291

[2] Ignacio Arnaldo, Una-May O'Reilly, and Kalyan Veeramachaneni. 2015. Building Predictive Models via Feature Synthesis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*. ACM, New York, NY, USA, 983–990. DOI: http://dx.doi.org/10.1145/2739480.2754693

[3] Mark Hinchliffe, Hugo Hiden, Ben McKay, Mark Willis, Ming Tham, and Geoffery Barton. 1996. Modelling Chemical Process Systems Using a Multi-Gene Genetic Programming Algorithm. In *Late Breaking Paper, GP'96*. Stanford, USA, 56–65.

[4] Christian Igel and Michael Hüsken. 2000. Improving the Rprop learning algorithm. In *Proceedings of the second international ICSC symposium on neural computation (NC 2000)*, Vol. 2000. Citeseer, 115–121.

[5] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA. http://mitpress.mit.edu/books/genetic-programming

[6] Trent McConaghy. 2011. FFX: Fast, Scalable, Deterministic Symbolic Regression Technology. In *Genetic Programming Theory and Practice IX*, Rick Riolo, Ekaterina Vladislavleva, and Jason H. Moore (Eds.). Springer New York, 235–260. DOI: http://dx.doi.org/10.1007/978-1-4614-1770-5_13

[7] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (09 Oct 1986), 533–536. DOI: http://dx.doi.org/10.1038/323533a0

[8] Dominic P. Searson. 2015. *GPTIPS 2: An Open-Source Software Platform for Symbolic Data Mining*. Springer International Publishing, Cham, 551–573. DOI: http://dx.doi.org/10.1007/978-3-319-20883-1_22

[9] Jan Žegklitz and Petr Pošík. 2017. Learning Linear Feature Space Transformations in Symbolic Regression. (2017). arXiv:1704.05134