Preventing Overfitting in GP with Canary Functions

Nate Foreman
Altarum Institute
3520 Green Court
Ann Arbor, Michigan, USA 48105
nathan.foreman@altarum.org

Matthew Evett
Eastern Michigan University
Ypsilanti, MI 48197
mevett@emich

ABSTRACT

Overfitting is a fundamental problem of most machine learning techniques, including genetic programming (GP). Canary functions have been introduced in the literature as a concept for preventing overfitting by automatically recognizing when it starts to occur. This paper presents a simple scheme for implementing canary functions using cross-validation. The effectiveness of this technique is demonstrated by applying it to the numeric regression problem. A list of conditions and criteria for applying this technique to other problem domains is also identified. Other strategies for dealing with overfitting in GP are discussed.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming – *program synthesis*; I.2.6 [**Artificial Intelligence**]: Learning – *induction, parameter learning*.

General Terms: Algorithms, Experimentation, Performance.

Keywords: Genetic Programming, Overfitting.

1. CANARY FUNCTIONS

Ideally, genetic programming (GP) would learn the true relationship between the inputs and the outputs over the entire problem space. Instead, GP can have a tendency to find solutions that are biased towards the training set. This can prevent GP from scaling to more difficult problems. In complex and sparse search spaces where GP is unlikely to easily find optimal solutions, the learning process focuses on minimizing the training error, which, in turn, increases the generalization error.

Several methods have been suggested for limiting the effect of overfitting in GP, including Editing [3], Minimum Description Length [5] and Dynamic Subset Selection [2].

Some researchers have hypothesized that we may be able to reduce the effects of overfitting by limiting the amount of time spent on training with a particular fitness function and a particular set of training data. That is, if we were somehow able to recognize when overfitting has started to occur, then we could take action to help produce more generalizable solutions. With that said, finding a reliable way of automatically determining when overfitting has begun to occur is critical. Evett *et al* [1] proposed a concept, called *canary functions*, for doing just that: "The canary function should be distinct from the fitness function, but also related toward meeting the same goal as the fitness function. The hope is that, since the canary function differs from the fitness function, its value will begin to degrade significantly from the fitness function at about the same time overfitting occurs." During each generation, the canary function is evaluated on some of the best of generation models. When the performance of the canary function on these models begins to consistently degrade, then we can have confidence that overfitting has begun.

The use of canary functions in [1] was peculiar to a particular domain. Our goal is to find a way to generalize the approach. This paper presents a simple scheme for implementing canary functions based on cross-validation, and applies it to the numeric regression problem. The objectives of this investigation are:

1. To determine if it is possible to limit the effects of overfitting, or at least recognize when it starts to occur, by monitoring the performance of the population with respect to a secondary validation dataset.
2. To identify some of the conditions or criteria necessary for applying this technique to other domains.

Cross-validation has been successfully used to overcome the problem of overfitting and to help select the best models in other machine learning algorithms like neural networks [4] and decision trees. It calls for using a separate set of fitness cases, distinct from the training examples, called the validation set. The algorithm monitors the error of the models with respect to the validation set, while using the training set to drive the search process. When the algorithm is complete, the model that has the lowest error over the validation set is selected because it is most likely to generalize upon unseen examples.

Cross-validation can be incorporated into the GP algorithm as a canary function in the following way:

1) When the performance of the models on the v*alidation set* begins to significantly diverge from that of the *training set*, take action to prevent overfitting.
2) After the algorithm is complete, the best of generation individuals should be evaluated upon the *test set* to determine the true quality of the predictive models.

Determination of divergence is the crucial step, and unfortunately is also the least well-defined step, because, as of yet, no one has implemented an *online indicator* for GP that signals when the performance of the models on the v*alidation set* begins to significantly diverge from that of the *training set* in an automated, generalizable, and reliable fashion. Prechelt [4] has made some progress with this problem in the field of neural networks.

GECCO'05, June 25-29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

Our experiments made use of two of Prechelt's online indicators: The *generalization loss* at generation g, GL(g), is defined to be the relative increase of the validation error over the minimum found so far, in percent. The *productivity quotient*, PQ(g) is defined to be the ratio of the *generalization loss* over the *training progress* (which is a measure of the progress of the training set, over a window.) Details as to the calculations of these values can be found in our full paper.

In addition to Prechelt's online indicators, another metric that may be useful for determining when overfitting starts to occur during GP runs is the *correlation coefficient*. A correlation coefficient of 0.80 means that 80% of the variation in one of the variables may be explained by variations in the other variable.

The correlation coefficient between the fitness of the best of generation individual with respect to the training set, and the fitness of the best of generation individual with respect to the validation set at the end of each generation of the GP run should be a strong indicator as to when indicate when they diverge.

2. THE EXPERIMENT

Symbolic regression was chosen as the problem domain upon which to demonstrate this technique, because it is easy to manipulate and it is easy to obtain datasets to serve as fitness cases. We used GP to evolve programs that approximated the functions (1) $F_1(x) = x^4 + x^3 + x^2 + x$, and (2) $F_2(x) = \cos(3x)$, respectively, across 20 data points.

Throughout the experimental trials, four main types of overfitting were identified: 1) Dramatic Divergence, 2) Slight Divergence, 3) Negative Performance, and 4) Negative Performance with Crossover. We explain just the first in this abbreviated paper.

Dramatic Divergence is depicted in Figure 1, and is a type of overfitting that is characterized by an early, sharp separation in the performance of the adjusted fitness of the best of generation solution between the validation set and the training set.

Figure 1a shows the adjusted fitness of the best of generation solution (BOGS) using the training and validation sets. The divergence of the two curves around generation 10 indicates that overfitting starts to occur there.

The correlation coefficient, in Figure 1b, provides a measure of the similarity of the data series plotted in Figure 1a (Adjusted Fitness). While generally high, the correlation coefficient is on a downward trend during the interval between generations 10 and 20, where we deduce that overfitting starts to occur.

The generalization loss, GL(g), is plotted in Figure 1c. It provides a measure of the rate of the change in generalization error with respect to the most accurate solutions found so far on the validation set. The generalization loss reaches a global maximum at generation 10, and another high peak is attained just after generation 20.

The progress of the performance of the training set is plotted in Figure 1d. P(g) is being calculated with respect to the previous five generations. As expected, the progress is high at the beginning of the GP run, and then tapers off gradually until a *fitness plateau* is reached.

The productivity quotient, PQ(g), plotted in Figure 1e, measures the ratio of the generalization loss over the progress of the training set. This figure has a very similar shape to that of Figure 1c. Unlike the curve in Figure 1c, a single global maximum no longer stands-out at generation 10. Rather, there are several high peaks that could all be used as stopping points.

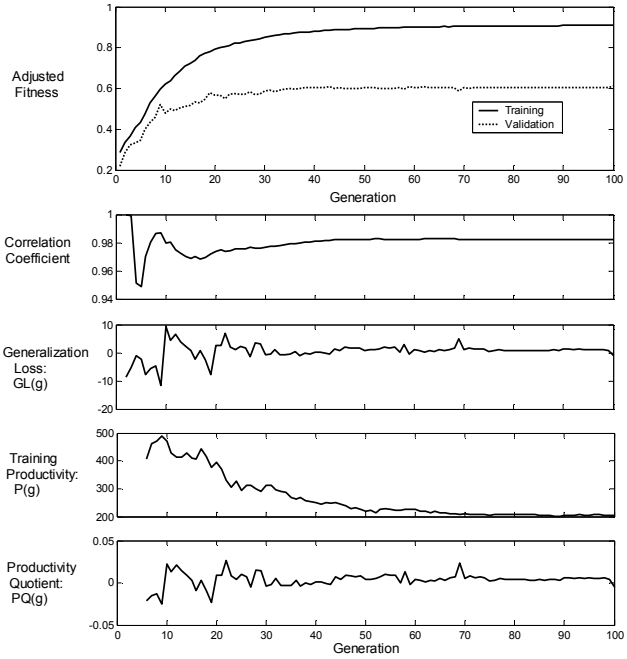

Figure 1a-e. Dramatic Divergence

Analysis of these graphs for each of the four types of overfitting we identified leads us to the following rule:

Rule: IF PQ(g) > α, AND the Correlation Coefficient at generation g is on a downward trend, THEN stop training because overfitting has occurred.

3. CONCLUSION

Our work has demonstrated that canary functions can be used to determine overfitting in symbolic regression problems. Much work remains to determine if the technique can be applied to other domains.

4. REFERENCES

[1] Matthew Evett, Taghi Khoshgoftar and Pei-der Chien and Edward Allen. GP-based software quality prediction, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 60—65, Morgan Kaufmann, 1998.

[2] Chris Gathercole and Peter Ross. Small populations over many generations can beat large populations over few generations in genetic programming. Genetic Programming 1997: Proceedings of the Second Annual Conference.

[3] Koza, J. R. Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge, MA, USA, 1992

[4] Lutz Prechelt. *Automatic early stopping using cross-validation: quantifying the criteria*. Neural Networks, Vol. 11, Number 4, pages 761—767, 1998.

[5] Byoung-Tak Zhang and Muhlenbein, H., Balancing Accuracy and Parsimony in Genetic Programming, Evolutionary Computation, vol.3, no.1, pages17--38, 1995