Using Fitness Distributions to Improve the Evolution of Learning Structures

Christian Igel
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
igel@neuroinformatik.ruhr-uni-bochum.de

Martin Kreutz
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
kreutz@neuroinformatik.ruhr-uni-bochum.de

Abstract- In this paper, the *absolute benefit*, a measure of improvement in the fitness space, is derived from the viewpoint of fitness distribution and fitness trajectory analysis. It is used for online operator-adaptation, where the optimization of density estimation models serves as an example. A new information theory based measure is proposed to judge the accuracy of the evolved models. Further, the absolute benefit is applied to offline analysis of new gradient based operators used for coefficient adaptation in genetic programming. An efficient method to calculate the gradient information is presented.

1 Introduction

In this paper we discuss the usefulness of judging evolutionary operators by their *absolute benefit*, a measure of improvement in the fitness space. Sections 2 and 3 motivate the absolute benefit from the viewpoint of fitness distribution and fitness trajectory analysis, respectively.

Two example applications of the absolute benefit are given. Both are structure optimization tasks that are solved using hybrid approaches: Evolutionary algorithms (EAs) using variable length representations determine the structure and local search (or learning) is used to adapt the parameters of the evolved models. The first task, described in Sec. 4, is the optimization of density estimation models. Online analysis of the absolute benefit is used to adapt the rate of application of the employed operators during the evolutionary process. A new information theory based measure is proposed to judge the accuracy of the estimated densities. The second task, described in Sec. 5, is symbolic regression using genetic programming (GP). New gradient based operators are proposed and analyzed offline using the absolute benefit. An efficient method to calculate the partial derivatives with respect to the numerical coefficients of the evolved expressions is presented.

2 Fitness distributions

Fitness distribution (FD) analysis has been proposed as a tool for designing efficient evolutionary algorithms [9, 13, 14]. The FD of a variation operator is described as the distribution of the offspring fitness given the fitness of the parents [13]. This distribution is generally quite complex and difficult to compute, so the analysis is restricted to certain features of the distribution like the probability of improvement and the expected improvement [22, 24].

Let $\Phi(.)$ denote the fitness or objective function to be maximized and let ω be a variation operator. For simplicity, we consider only asexual (or single-parent) operators for the moment. Then the probability of improvement IP_ω of an operator ω is defined as $\mathrm{IP}_\omega = P(\Phi(\omega(a)) > \Phi(a))$, where a is a random individual. But it is not only important how often offspring are better than their parents, but also how much better they are [1]. This can be expressed by the the expected improvement $\mathrm{EI}_\omega = E[\Phi(\omega(a)) - \Phi(a)]$, where $E[.]$ denotes the expectation. The *benefit* B_ω [27] combines both measures (see also [9]):

$$\mathrm{B}_\omega = E[\max\{0, \Phi(\omega(a)) - \Phi(a)\}] \ . \tag{1}$$

None of these measures necessarily reflects evolvability which is defined as the ability of an algorithm to produce individuals fitter than any existing, one of the key properties determining the quality of evolutionary search [1]. If we replace the parent fitness by the fitness of the best individual in the population a_{best} in the above definitions, the measures get linked to the evolvability. Hence, we get the *absolute benefit* (corresponding to the *local delta* in [8])

$$\mathrm{B}^*_\omega = E[\max\{0, \Phi(\omega(a)) - \Phi(a_{\text{best}})\}] \ , \tag{2}$$

and similary the *absolute probability of improvement* IP^*, etc. Note that the investigation of the absolute benefit for a single operator is similar to the analysis of a (1+1) evolutionary algorithm (as e.g. in [22, 24]).

Deeper insight into the evolutionary process can be gained by investigating the proposed measures in a generation dependent manner [14], i.e. the observed features become functions of the generation g. They can be estimated during one or more trials of the evolutionary algorithm, e.g.

$$\mathrm{B}^{*\,(g)}_\omega \approx \frac{1}{|\mathcal{P}^{(g)}_\omega|} \sum_{a \in \mathcal{P}^{(g)}_\omega} \max\{0, \Phi(\omega(a)) - \Phi(a_{\text{best}^{(g)}})\} \tag{3}$$

and $\mathrm{B}^*_\omega \approx \frac{1}{G}\sum_{g=1}^{G} \mathrm{B}^{*\,(g)}_\omega$, where $\mathcal{P}^{(g)}_\omega$ denotes the subset of the population at generation g that undergoes modification through operator ω and G is the last generation of the trial(s).

If the costs to produce an offspring are not equal for all applied variation operators, the observed measures should be normalized by these costs. In such a case, the costs of the application of ω to an individual a are measured by a cost function $c(.)$ and each fitness difference in Eqs. (1)–(3) is divided by $c(\omega(a))$.

There are several ways to extend the above fitness analysis to multi-parent operators. It was suggested to average the fitness values of the parents [13] or to take the fitness value of the best parent. Here we assume that there is one dedicated parent a that is modified by one out of a set Ω of possible operators, where some operators make use of information represented in the current population (e.g. recombination). An application of such an multi-parent operator ω can then be written as $\omega(a, \mathcal{P})$. The fitness of the single individual a is used for analysis.

To be precise, measures incorporating the generation number or the population's best individual cannot be derived from the original definition of the fitness distribution that does not know about populations and generations. However, the FD can be regarded as the basis of these measures.

3 Fitness trajectories and benefit

In this section, we establish the link between the common analysis of fitness trajectories (and therefore of the performance) and the absolute benefit described in Sec. 2.

A straight forward way to measure the performance of an EA is to inspect the development of the fitness over time,

$$\Phi^{(g)}_{\text{best}} = \max_{a \in \mathcal{P}^{(g)}} \Phi(a) \quad \text{and} \quad \Phi^{(g)}_{\text{offline}} = \frac{1}{g} \sum_{n=1}^{g} \Phi^{(n)}_{\text{best}} \ ,$$

eventually averaged over several trials. The moving average over the generations $\Phi^{(g)}_{\text{offline}}$ is called *offline* performance (introduced by K. A. DeJong, cf. [12]),

If the point of interest is the development of the best fitness, the value $\Delta\Phi^{(g)}_{\text{best}} = \Phi^{(g+1)}_{\text{best}} - \Phi^{(g)}_{\text{best}}$ can be derived from the previous equation. It reflects the current performance of the EA. If only one operator ω is employed, $\Delta\Phi^{(g)}_{\text{best}}$ can be written as

$$\Delta\Phi^{(g)}_{\text{best}} = \max_{a \in \mathcal{P}^{(g)}} \max \left\{ 0, \Phi(\omega(a)) - \Phi(a^{(g)}_{\text{best}}) \right\} \ , \qquad (4)$$

under the assumption that either an elitist selection strategy is used or the best individual is stored outside the population. If the aim of the investigation is to judge different operators, Eq. (4) can be extended to

$$\Delta\Phi^{(g)}_{\omega\,\text{best}} = \max_{a \in \mathcal{P}^{(g)}_{\omega}} \max \left\{ 0, \Phi(\omega(a)) - \Phi(a^{(g)}_{\text{best}}) \right\} \ .$$

If the outer maximization is replaced by the average and the result is normalized by the population size, we get the estimation of the absolute benefit for each operator $\omega \in \Omega$ (see Eq. (3)). If the probability of being better than the best is very low (as in GP, cf. [14]) and in every run normally at most one of the applications of a single operator ω yields an offspring that is better than the best in each generation, then $\Delta\Phi^{(g)}_{\omega\,\text{best}}$ is proportional to $\mathrm{B}^{*\,(g)}_{\omega}$. If more than one application of an operator produces an offspring better than the best, averaging (i.e. calculating the benefit) leads to a more reliable statistic than maximization (i.e. observing $\Delta\Phi^{(g)}_{\omega\,\text{best}}$), because more operations are considered. It appears not only to be more intuitive to consider the global improvements that occur at one generation independently from each other, this method also uses the available data in a more efficient manner.

In the same way as the generation dependent absolute benefit can be derived from the observed fitness trajectories of the population-best, the absolute benefit averaged over all generations (the *accumulated* absolute benefit, see Eq. (2)) can be derived from the offline fitness.

Offline analysis of EA trials is often intended to find a suitable operator setting. In order to determine the optimal setting including the subset of operators and the rates of application of the selected operators for a given class of problems, all subsets have to be tested. Although some subsets can be discarded at the outset, e.g. because they are not able to traverse the whole search space, this method is computationally not tractable. In the case of multiple operators, the benefit and other FD features allow to observe the behavior of different operators at the same time. This analysis has two advantages: Firstly, the values for the different operators at a certain generation can be compared, because the populations to which the operators are applied are comparable. This is in contrast to observing each operator alone (or with a minimum set of other operators needed to traverse the search space) in different trials, where the populations evolve differently due to the bias of the employed operators. Secondly, only the simultaneous evaluation of the FD features enables online adaptation of the operators.

One drawback of the proposed method when used for offline analysis is that the operators may influence each other. These interdependencies may lead to effects that are not observed when operators are dropped from the set.

4 Optimization of density estimation models

The estimation of the probability density function (pdf) of a data generation process given a finite sample of observations is a very fundamental tool for many tasks including data compression, classification and regression [4, 15]. A variety of methods have been proposed in the field of statistics as well as artificial neural networks (ANNs) [4]. However, the task of density estimation still remains difficult. This holds especially for the estimation of non-stationary probability distributions. In this section we employ an evolutionary algorithm (EA) for the optimization of both the parameters and the structure of a Gaussian mixture model in order to estimate a time-varying pdf. That is, the optimization process has to follow the distribution over the time. An outline of the employed model and a short description of the EA is presented. In the context of estimating non-stationary pdfs, the issue of operator-adaption arises. In this study the absolute benefit (introduced in Sec. 2) is used as a measure of the online performance and serves as the basis for the adaptation of operator probabilities in an EA.

4.1 Gaussian mixture models

The task of density estimation is to find a good approximation of the true pdf $p(\boldsymbol{x})$ given a finite sample $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of observations and a density model $\hat{p}(\boldsymbol{x}|\boldsymbol{\theta})$ which is characterized by the form of the function $\hat{p}$ and its parameters $\boldsymbol{\theta}$. Mixtures of densities have been considered as very general and computationally efficient semi-parametric models for density estimation. They consist of a linear superposition of m simple component densities. We employ normal densities:

$$\hat{p}(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{i=1}^{m} \frac{\alpha_i}{\sqrt{2\pi}^n \prod\limits_{k=1}^{n} \sigma_{ik}} \exp\left(-\frac{1}{2}\sum_{k=1}^{n}\left(\frac{x_k - \mu_{ik}}{\sigma_{ik}}\right)^2\right)$$

The vector of parameters $\boldsymbol{\theta} = (\alpha_i, \ldots, \alpha_m, \theta_i, \ldots, \theta_m)$ characterizing $\hat{p}$ includes the weighting coefficients α_i and the parameters θ_i of the component densities. In order to yield a valid density the α_i are constrained to sum up to one. This model has several appealing properties: they are universal in the sense that they can approximate any continuous probability distribution (similar to radial basis function networks [21]), they can cope with multi-modal distributions, and their complexity can be easily adjusted by the number of components.

A common measure for the quality of the estimation is the likelihood of the parameters given the data which corresponds to the probability of the occurrence of the sample $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ for the chosen probability density model $\hat{p}(\boldsymbol{\theta})$. Usually the logarithm of the likelihood is taken which leads to the well known log-likelihood function:

$$\ell(\boldsymbol{\theta}) \;=\; \log \prod_{l=1}^{N} \hat{p}(\boldsymbol{x}_l|\boldsymbol{\theta}) \;=\; \sum_{l=1}^{N} \log\, \hat{p}(\boldsymbol{x}_l|\boldsymbol{\theta}) \;\;.$$

where the $\boldsymbol{x}_l$ are assumed to be iid sampled with the probability density function $\hat{p}(\boldsymbol{x}|\boldsymbol{\theta})$. A maximum likelihood estimation corresponds intuitively to the most likely model in the light of the given data. Another sensible measure is proposed by the authors and relies on the following consideration: Let $p(\boldsymbol{x})$ denote the true density and $\hat{p}(\boldsymbol{x})$ the estimation of p, respectively. In the case of an infinite sample the log-likelihood reads:

$$\ell(\boldsymbol{\theta}) \;=\; \int_{-\infty}^{+\infty} \log \hat{p}(\boldsymbol{x}|\boldsymbol{\theta})\, p(\boldsymbol{x})\, d\boldsymbol{x} \;\;.$$

If the estimation $\hat{p}$ coincides with p the log-likelihood is just the negative of the Shannon entropy [7]:

$$p = \hat{p} \;\Rightarrow\; \ell(\boldsymbol{\theta}) = \int_{-\infty}^{+\infty} \log p(\boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x} = -H(\boldsymbol{x}) \;\;.$$

The Shannon entropy is a measure of uncertainty of a random variable and can also be considered as a measure of disorder of a probability distribution. The entropy will be maximum for a uniform distribution since the instances of a uniformly distributed random variable will show no order nor structure. On the other hand the entropy will be minimum for a peaked distribution since such a distribution is highly ordered: all probability mass is concentrated in only one peak. This relation between the log-likelihood and the Shannon entropy leads directly to a new optimization criterion which provides a tradeoff between underestimation which corresponds to a low log-likelihood and overestimation which corresponds to a low entropy:

$$Q(\boldsymbol{\theta}) \;=\; \frac{1}{N}\sum_{l=1}^{N} \log\, \hat{p}(\boldsymbol{x}_l|\boldsymbol{\theta}) + H(\boldsymbol{x}) \;\;. \tag{5}$$

In this sense the criterion $Q(\boldsymbol{\theta})$ accounts for what is called in statistics the *bias variance dilemma* (which can only be escaped in the limit of an infinite sample) [10]. Thus, this criterion includes a very intuitive regularization imposed by the Shannon entropy. Overspecialized solutions would increase the log-likelihood but, on the other hand, would decrease the Shannon entropy. This criterion can be viewed as a *penalized maximum likelihood criterion* [11] penalized with the entropy of the model. Another objective may be the minimization of the absolute value of $Q(\boldsymbol{\theta})$. In this paper we pursue the penalized maximum likelihood approach. It is not possible to solve the entropy integral analytically for Gaussian mixtures. It must therefore be approximated (e.g. by Monte Carlo sampling). This fact renders maximization methods like gradient descent or EM algorithms computationally inefficient. On the other hand, direct optimization methods like EA only rely on the value of the objective function and, therefore, provide a well suited framework for optimizing non-differentiable and noisy functions.

4.2 Evolution of Gaussian mixtures

The optimization of the structure of a model is a difficult problem, since the search space is in general non-differentiable and multi-modal. Evolutionary algorithms have been considered to be a promising method for dealing with this class of optimization problems. However, the evolutionary operators and the mapping between the genotype space (the representation on which the operators act) and the phenotype space (the structure which is evaluated using an appropriate quality measure) have to be chosen carefully and in general depend on the problem. For specific details of the EA used here the reader is referred to [17]. However, we shall point out some conditions which lead to specific operator settings.

The parameters θ_i of each component are encoded directly in the genome which is justified by the restriction of the interaction between the components to the vicinity of each component (due to the local nature of Gaussians). This generally leads to a low epistasis in the genome.

In the context of recombination the problem of *competing conventions* arises: Due to the invariance to permutations of the components distinct genomes map to functionally equivalent phenotypes (models). This problem is circumvented

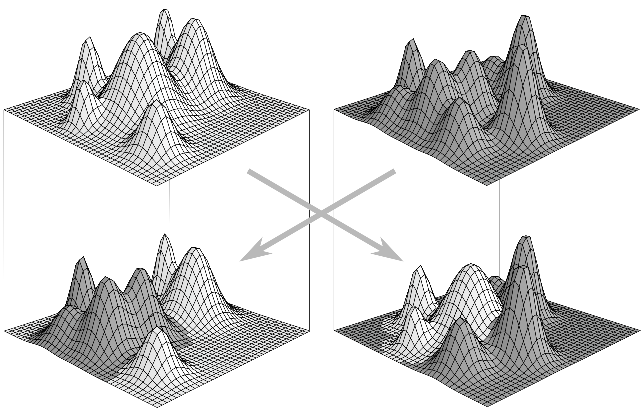

Figure 1: Crossover operator: The portions of mixture components of both parents lying in the crossover section are exchanged.

by using a crossover operator which samples the crossover points in the input space, see Fig. 1, rather than on the position in the genome (this idea has been adopted from [5]).

Several mutation operators for structure modification of the model have been considered. Straightforward choices include simple insertion and deletion operators. However, the insertion and deletion of components in the model can be very disruptive and violate the principle of a strong causal mapping between the genome and the phenotype [25]. In order to minimize these effects and to better control the impact of mutation we employ special *merge* and *split* operators which try to increase and decrease, respectively, the number of components while at the same time keeping the effect of mutation as small as possible. Details of these operators are discussed in [17].

4.3 Operator-adaptation

The evolutionary optimization of Gaussian mixtures outlined in the previous section has been successfully applied to both density estimation [17] and nonlinear regression [18]. In theses experiments the evolutionary setting (internal parameters of the operators and rates of their application) was held fixed. Yet, it has been long recognized that the choice of the evolutionary operators and the rates at which they are applied in an EA have a significant impact on the performance of the EA. The optimal rate of application of each operator, however, depends not only on the given problem, that is on the structure of the fitness landscape, but also on the current state of the search [26]. This means that in order to maintain good or near optimal rates, these rates must be tracked by some kind of adaptation mechanism during search. An intuitive approach is to keep track of the relative performance of the different operators by comparing their "success" in producing good offspring. The rates of application are then adjusted according to these performance values (examples have been proposed by [8, 27]). In our investigation the "success" is measured by the absolute benefit introduced in Sec. 2.

For the optimization of Gaussian mixtures we employ various crossover (see e.g. Fig. 1) and mutation operators: different merge and split strategies, combinations of merge and split, etc. (cf. [17, 18]). All operators are followed by a local adaption of the model parameters which is performed by a single EM [23] iteration. Totally, we get 18 different operators including the simple local parameter adaptation. Initially, all operators are applied at the same rate $P(\omega) = 1/|\Omega|\ \forall \omega \in \Omega$. During evolution a statistic of the absolute benefit of each operator is built over the whole population and a period of N_{adapt} generations. After each N_{adapt} generations the rates of all operators are updated according to their estimated absolute benefit. In order to avoid fluctuations of the rates a momentum term parameterized with a factor γ is used. Furthermore, the rates are restricted by a lower bound in order to prevent the complete elimination of an operator. In our experiments, this bound was set to $P_{\min} = 0.1/|\Omega|$. The final adaptation rule for the new operator probabilities $P^{(\text{new})}(\omega)$ reads:

$$P'(\omega) \quad = \quad \gamma \cdot \max\left\{ P_{\min}, \frac{\sum_{\tau=1}^{N_{\text{adapt}}} \mathrm{B}_{\omega}^{*\,(g-\tau)}}{\sum_{\tilde{\omega}\in\Omega} \sum_{\tau=1}^{N_{\text{adapt}}} \mathrm{B}_{\tilde{\omega}}^{*\,(g-\tau)}} \right\} + (1-\gamma) \cdot P^{(\text{old})}(\omega) \quad , \quad P^{(\text{new})}(\omega) \; = \; \frac{P'(\omega)}{\sum_{\tilde{\omega}\in\Omega} P'(\tilde{\omega})} \; .$$

4.4 Experimental results and discussion

In order to assess the performance of the proposed adaptation scheme we employ our method to the estimation of a non-stationary density both with and without operator-adaptation. The EA tries to fit the Gaussian mixture model to a normal density with $\sigma = 1/16$ along a ring with radius $r \in [1, 2]$ while the radius is changing over the time: The radius r describes one period of a sine wave over the interval of 1000 generations with $r_{min} = 1$ and $r_{max} = 2$, respectively. One additional difficulty is the choice of the right regularization [4]. Especially for the estimation of non-stationary densities the amount of required regularization may vary over time. Since the proposed measure Q is independent from any regularization parameters, it is particulary applicable to this problem domain. As depicted in Fig. 2 the EA is able to follow the distribution and the operator-adaptation clearly outperforms the EA with fixed operator probabilities with respect to the quality measure $Q(\boldsymbol{\theta})$ (see Eq. (5)). The momentum factor γ was set to 0.5.

Since Q is evaluated by Monte Carlo sampling the curve in Fig. 2 exhibits rather large fluctuations. This can be avoided by increasing the number of Monte Carlo samples. In favor of efficiency in terms of evaluation time this number is chosen relatively small.

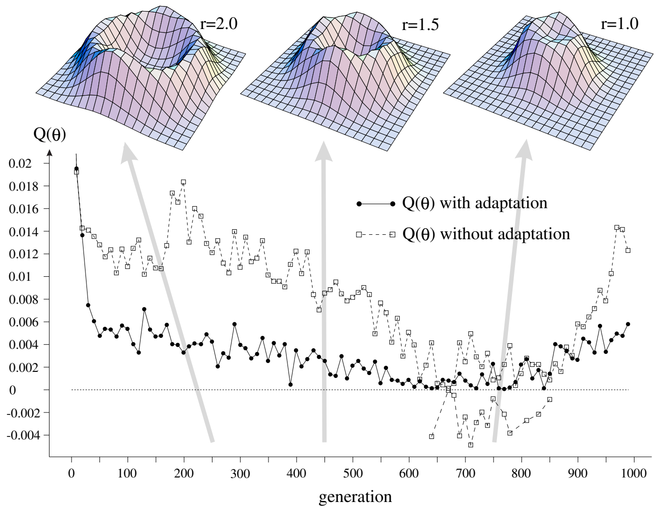

Figure 2: The plot shows the absolute value of the quality measure $Q(\boldsymbol{\theta})$ over the course of evolution. The solid line corresponds to an EA with operator-adaptation and the fine dashed line to an EA without adaptation. In the latter case the value of Q becomes negative between generation 640 and 860 which corresponds to an underfitting of the true density, shown by the long dashed line (additionally $|Q|$ is plotted). For the optimal model Q would be zero, see dotted line. The three small plots above the main plot are snapshots taken from the EA with operator-adaptation at generation 250, 500 and 750, respectively. All values are averaged over 16 trials and smoothed over 10 generations.

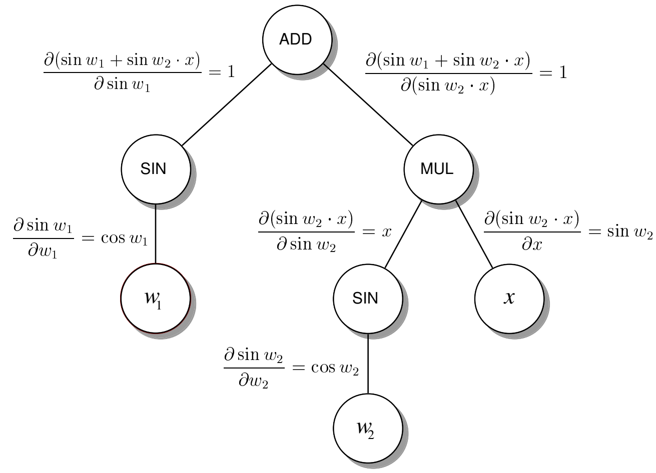

Figure 3: This example tree calculates the function $\hat{y}(x) = \sin w_1 + \sin w_2 \cdot x$. The additional information stored at the nodes can be computed during a single evaluation of $\hat{y}$. Multiplying the stored results along the path from a numerical coefficient to the root node yields the corresponding gradient, e.g. $\partial\hat{y}(x)/\partial w_2 = \cos w_2 \cdot x \cdot 1$.

5 Symbolic Regression using GP

5.1 Adapting the numerical coefficients in GP

In genetic programming (GP) expressions represented as parse trees are to be evolved [16]. The goal of real-valued symbolic regression in GP is to find a mathematical expression that fits a function given a sample of input-output patterns. Both the functional form and the numeric coefficients are to be evolved. Here we are concerned with the adaptation of the real-valued coefficients. In traditional GP, once the coefficients have been generated they remain fixed. It is left to the GP operators to arrange the constants and evolve expressions that calculate the needed numerical values [16]. Better results can be obtained by using an additional mechanism for the adaptation of the real-valued coefficients, like Gaussian mutation (e.g. in [6]). However, if the the evolved expressions are differentiable, or at least piece-wise differentiable, with respect to the numerical coefficients, directed mutation using the gradient information can be employed. This is similar to gradient based learning of ANNs.

Let T be a parse tree with c numerical constants $w_1, \ldots, w_c$. Let $\hat{y}(.)$ denote the real-valued function represented by T. Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ be the sample patterns and σ_y^2 the variance of the target values. To maximize the fitness, we minimize the normalized mean squared error given by

$$E = \frac{1}{\sigma_y^2}\frac{1}{N}\sum_{n=1}^{N}(\hat{y}(\boldsymbol{x}_n) - y_n))^2 \ . \tag{6}$$

For each numeric coefficient, the gradient information

$$\frac{\partial E}{\partial w_i} = 2\frac{1}{\sigma_y^2}\frac{1}{N}\sum_{n=1}^{N}(\hat{y}(\boldsymbol{x}_n) - y_n)\frac{\partial \hat{y}}{\partial w_i}(\boldsymbol{x}_n) \tag{7}$$

can be used to guide the adaptation of w_i in a direction minimizing Eq. (6).

In order to calculate the gradient, the partial derivatives of $\hat{y}$ with respect to the numerical coefficients of T have to be determined. This can be done in a computationally efficient manner using a method inspired by the back-propagation procedure for ANNs. Applying the chain rule, the partial derivatives factorize to simpler derivatives, which can be computed during the function evaluation using the values returned by the child trees. These factors are stored at the nodes. This means, each fitness evaluation produces the needed intermediary results of the partial derivatives as a byproduct; therefore only a single evaluation of the tree is sufficient to compute Eq. (7) for all w_i. An example is given in Fig. 3. Moreover, if a calculation of Eq. (6) is followed by a calculation of Eq. (7) for the same tree and the same set of patterns, both results can be achieved by a single evaluation, i.e. the algorithm has to go through the patterns only one time. For example, this occurs when a *line search* algorithm [4] is used for exploiting the gradient information.

The derivative of the error function exists if the evolved expression is differentiable. This is guaranteed if the functions in the function set are differentiable. However, the assumption that the functions are at least piecewise differentiable (like the protected division [16]) is sufficient; the points where the derivative does not exist can be detected during the evaluation of the expression and can be excluded from the gradient calculation easily without making the overall result misleading.

In preliminary experiments, we found standard *steepest gradient descent* not to be useful for the optimization of the symbol regression expressions since the choice of an adequate learning rate appeared to be difficult. The evolved structures were too different to find a suitable single learning rate for all individuals and adaptation of individual learning rates needed too many function evaluations. So we constructed the following local search operators:

LS This operator performs a line search in gradient direction.

BFGS This operator optimizes the coefficients using a *quasi-Newton* method called *Broyden-Fletcher-Goldfarb-Shanno* algorithm. When used as an search operator, as done in Sec. 5.2, an upper bound of iterations parameterizes the intensity of local search. In this study, the bound was set to three.

Adaptation of the coefficients using the BFGS algorithm converges to locally *optimal* numerical parameters. Of course, the coefficients are locally optimal with respect to the training set, so over-fitting may lead to bad generalization of the adapted model. However, all the knowledge from ANNs can be employed to avoid this (regularization, early stopping etc.). These properties make the proposed method well suited for fine tuning already evolved expressions and so completing the work of the evolutionary process. Possible fields of application may include GP for time series modeling [19], multiple interacting programs (MIPs) [3] and the training of higher order ANNs or neural trees [28].

In the next section, we investigate the usefulness of employing the gradient based operators during the evolutionary process by investigating the absolute benefit of different coefficient adaptation strategies.

5.2 Experiments

We used a toy problem to analyze the operators. The task was to find a solution of the simple harmonic motion differential equation $\ddot{\Theta} + \Theta = 0$ given by $\Theta(t) = a \sin t + b \cos t$. The initial conditions (position and angle velocity) were set to $a = 5.558$ and $b = 4.2$. The terminal set included real-valued parameters and t. The function set was $\{+, -, *, \sigma, \sin\}$ with $\sigma(x) = 1/(1 + exp(-x))$. A training set of 100 and a test set of 20 patterns sampled uniformly out of the interval $[0, 4\pi]$ were used.

Besides the *LS* and the *BFGS* operators described in Sec. 5.1, the following operators were used that solely modify numerical coefficients:

OneC A Gaussian mutation is performed on a randomly chosen numerical coefficient.

AllC A Gaussian mutation is performed on all numerical coefficients [6].

HC This hill-climbing operator perturbs each numerical coefficient with Gaussian mutation and checks each time whether the mutation was successful. Unsuccessful changes are withdrawn.

All Gaussian mutations had zero mean and a standard deviation of 0.1. The following operators were employed to evolve the structure of the trees:

OneNode The operator replaced a randomly selected node with a different node of the same arity.

Promote A random subtree, say T', is selected and replaced by a random function node. T' becomes a child of the new function node, the other children are initialized to random terminals.

Demote The operator selects a random function node an replaces it with the largest of its children.

Trunc A randomly selected function node was replaced by a random terminal [6].

Grow A random leaf is replaced with a newly generated subtree [6].

Crossover The crossover operator used in this investigation chose a mate out of the population and selected pivot points in the parent and mate tree independently with equal distribution. The former parent tree was returned as the offspring.

SHCCrossover Another tree was chosen randomly out of the population. This *pattern tree* was randomized by replacing every node by an arbitrary node of the same arity. Then crossover was applied between the selected parent and the new created tree as described above. The modified parent tree was returned as the offspring. This mutation operator was inspired by strong headless chicken crossover introduced in [2].

The first three operators were introduced in [20] and were designed to make as small steps in the genotype space (measured by an tree edit distance) as possible.

In each generation, for every tree in the population, say T_i, one of the operators was chosen at random with equal probability and applied to produce a single offspring. This operation was repeated if the application of the selected operator did not lead to an offspring with a genotype different from the parent. The fitness of T_i was used to calculate the absolute benefit.

The time consumption of the various operators is very different, so their benefit has to be normalized by a cost function as described in Sec. 2. The costs were set equal to the number of parse tree evaluations performed by the operator, i.e. the number of times the sample patterns have to be presented. For example, if one *BFGS* application needed 40 evaluations to calculate the needed function and derivative information, its absolute benefit was divided by 40.

Evolutionary programming style tournament selection [6] with tournament size 20 was applied to form the next generation. The population size was 500 and the length of the individuals was limited to 50 nodes in all the experiments. 30 independent trials were performed.

5.3 Results & discussion

The normalized absolute benefit for all operators is presented in Tab. 1 and the time dependent normalized absolute benefit for the three local search operators is presented in Fig. 4. The main results can be summarized as:

- The gradient based operator performed considerably better in terms of B^* than the other operators that solely change the numerical coefficients, despite the fact that they need more function evaluations. The absolute benefit of the 3-iteration

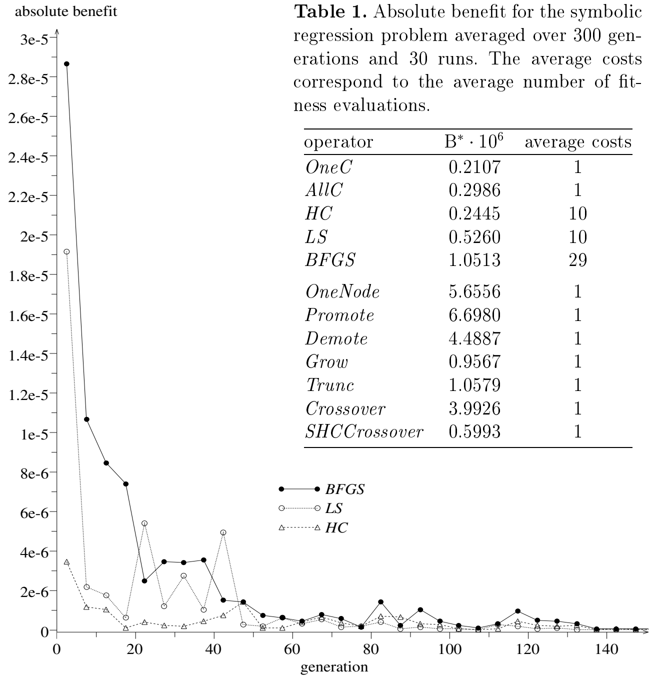

Table 1. Absolute benefit for the symbolic regression problem averaged over 300 generations and 30 runs. The average costs correspond to the average number of fitness evaluations.

operator	$B^* \cdot 10^6$	average costs
OneC	0.2107	1
AllC	0.2986	1
HC	0.2445	10
LS	0.5260	10
BFGS	1.0513	29
OneNode	5.6556	1
Promote	6.6980	1
Demote	4.4887	1
Grow	0.9567	1
Trunc	1.0579	1
Crossover	3.9926	1
SHCCrossover	0.5993	1

Figure 4: Absolute benefit of the three local search operators averaged over 300 generations and 30 trials. Although *BFGS* is very good throughout the evolutionary process, the operator generates the largest benefit in the early generations where the coefficients are badly adapted.

BFGS operator was twice as high as the B^* of *LS*, which was still about twice as high as the absolute benefit produced by *OneC*, *AllC* and *HC*.

• The average number of evaluations needed by *HC* and *LS* were approximately equal. The 3-iteration *BFGS* operator performs three line search operations and therefore produced on average about three times the costs of *LS*.

• The *AllC* operator produced higher B^* values than *OneC* and *HC*. Further investigation showed that *AllC* performed better than *OneC* in early generations due to larger improvements, but worse in later generations when the numerical coefficients were better adapted. The probability of generating better solutions than the population-best fell to zero, whereas *OneC* was still able to produce offspring better than the best. This is in agreement with the results in [14].

• Although the structure modifying operators have a much lower probability of being better than the best (cf. [14]) compared to the operators that solely adapt the numerical coefficients, most of them showed higher absolute benefit. This means, when a structural change led to an improvement, the amount of improvement was on average larger than the improvements achieved by coefficient adaptation.

• The three "local" structure operators introduced in [20] showed high B^* values, especially *Demote* performed better than *Trunc* and *Promote* outperformed *Grow*.

• The recombination operator *Crossover* produced considerable higher benefit results than the closely related *SHCCrossover* operator.

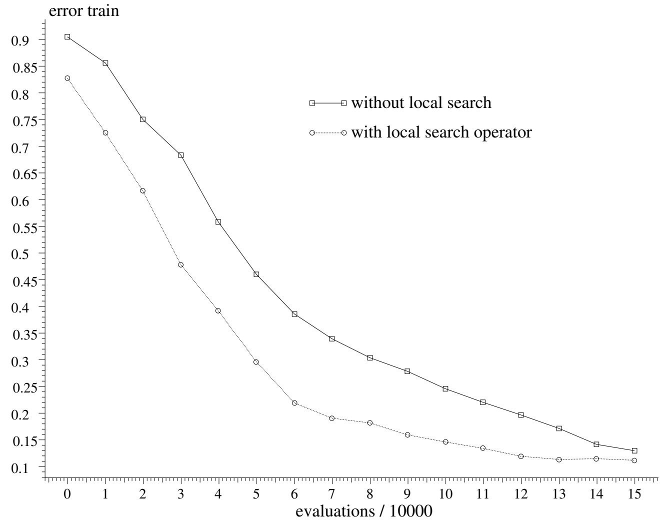

Figure 5: In these experiments, the rates of application of operators that solely modify the numerical coefficients were the same. In the experiment without local search only *AllC* and *OneC* were employed, in the other experiment *BFGS* was applied twice as often as *AllC* and *OneC*, respectively. For each setting, twenty trials starting from the same initial populations were performed. The experiments that involved gradient based operators performed considerable better, over-fitting was not observed. However, other experiments using only *BFGS* without explorative operators showed lower errors in the early generations, but led to worse final results due to premature convergence.

Although the experiment provides deeper insight into the behavior of the operators, it also shows the limitations of the proposed method:

• Because the average numbers of fitness evaluations of the coefficient adaptation operators are far different, the good performance in terms of B^* need not carry over to better final solutions when simply substituting operators in an EA. For example, the *BFGS* operator spends about 30 evaluations for adapting *one* individual. This means that *OneC* and *AllC* can be applied to 30 different individuals for one *BFGS* application. This dissimilar distribution of the fitness evaluations among individuals and generations leads to different dynamics.

• The question, whether there are interdependencies between the local (exploitative) operators and the global (explorative) operators cannot be answered from this investigation. Are the operators *OneC* and *AllC* needed to detect new basins of attraction that the employed gradient based operators, that can never escape local minima, can exploit (cf. Fig. 5)? A general approach to answer such questions might be to analyze the history of modifications that led to a very good individual [8]. An example how using the *BFGS* operator increases the performance both in accuracy of the obtained results and speed of convergence is shown in Fig. 5.

Further experiments are needed to analyze the interaction between local and global search for coefficient adaptation in symbolic regression, e.g. other operators using the gradient information have to be tested. Future work should include the application of the proposed operators to other domains (see end of Sec. 5.1) and real-world tasks.

6 Summary and conclusion

The absolute benefit was presented as a suitable measure for both online and offline analysis of structure evolution tasks. It was derived from the viewpoint of fitness trajectory and fitness distribution analysis.

The absolute benefit was successfully applied to online operator-adaptation in the framework of density estimation, leading to more accurate models. A new information theory based measure was employed to judge the quality of the estimated densities. It shows not only the accuracy of the evolved structure, but also whether over-fitting occurred. This preliminary investigation shows that operator-adaptation can be helpful in structure optimization tasks.

Exploiting gradient information using higher order optimization algorithms was proposed as a well suited method for adapting the real-valued coefficients in evolved symbolic expression. Such gradient based operators lead to a faster adaptation of the parameters as undirected mutations do and are especially well suited for fine tuning the final result of an EA. The gradient information can be obtained in an efficient manner by a technique similar to back-propagation.

Bibliography

[1] L. Altenberg. The evolution of evolvability in genetic programming. In K. E. Kinnear, Jr., ed., *Advances in Genetic Programming*, chap. 3, pp. 47–74. MIT Press, 1994.

[2] P. J. Angeline. Subtree crossover: Building block engine or macromutation? In J. R. Koza et al., eds., *Genetic Programming 1997: Proc. 2nd Ann. Conf.*, pp. 9–17. Morgan Kaufmann, 1997.

[3] P. J. Angeline. Evolving predictors for chaotic time series. In S. Rogers et al., eds., *Proceedings of SPIE: Application and Science of Computational Intelligence*, volume 3390, pp. 170–180, 1998.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[5] B. Carse and T. C. Fogarty. Tackling the "curse of dimensionality" of radial basis function neural networks using a genetic algorithm. In H.-M. Voigt et al., eds., *Parallel Problem Solving from Nature IV*, pp. 710–719. Springer, 1996.

[6] K. Chellapilla. Evolutionary programming with tree mutations: Evolving computer programs without crossover. In J. R. Koza et al., eds., *Genetic Programming 1997: Proc. 2nd Ann. Conf.*, pp. 431–438. Morgan Kaufmann, 1997.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[8] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, ed., *Proc. 3rd Int. Conf. on Genetic Algorithms*, pp. 61–69. Morgan Kaufmann, 1989.

[9] D. B. Fogel and A. Ghozeil. Using fitness distributions to design more efficient evolutionary computations. In *Proc. 1996 IEEE Int. Conf. on Evolutionary Computation*, pp. 11–19. IEEE Press, 1996.

[10] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[11] P. J. Green. Penalized likelihood. *Encyclopaedia of Statistical Sciences, update volume*, 1996.

[12] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man, And Cybernetics*, 16(1):122–128, 1986.

[13] J. J. Grefenstette. Predictive models using fitness distributions of genetic operators. In L. D. Whitley and M. D. Vose, eds., *Foundations of Genetic Algorithms 3*, pp. 139–161. Morgan Kaufmann, 1995.

[14] C. Igel and K. Chellapilla. Fitness distributions: Tools for designing efficient evolutionary computations. In L. Spector et al., eds., *Advances in Genetic Programming 3*, chap. 9. MIT Press, 1999. in press.

[15] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[16] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[17] M. Kreutz, A. M. Reimetz, B. Sendhoff, C. Weihs, and W. von Seelen. Optimisation of density estimation models with evolutionary algorithms. In A. E. Eiben et al., eds., *Parallel Problem Solving from Nature V*, pp. 998–1007. Springer, 1998.

[18] M. Kreutz, A. M. Reimetz, B. Sendhoff, C. Weihs, and W. von Seelen. Structure optimization of density estimation models applied to regression problems with dynamic noise. In D. Heckerman and J. Whittaker, eds., *Artificial Intelligence and Statistics 99: Proc. 7th Int. Workshop*, pp. 237–242. Morgan Kaufmann, 1999.

[19] H. Oakley. Two scientific applications of genetic programming: Stack filters and non-linear equation fitting to chaotic data. In K. E. Kinnear, Jr., ed., *Advances in Genetic Programming*, chap. 17, pp. 369–389. MIT Press, 1994.

[20] U.-M. O'Reilly and F. Oppacher. Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In Y. Davidor, H.-P. Schwefel, and R. Männer, eds., *Parallel Problem Solving from Nature III*, pp. 397–406. Springer, 1994.

[21] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[22] I. Rechenberg. *Evolutionsstrategie '94*. Frommann-Holzboog, 1994.

[23] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

[24] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, 1995.

[25] B. Sendhoff, M. Kreutz, and W. von Seelen. A condition for the genotype–phenotype mapping: Causality. In T. Bäck, ed., *Proc. 7th Int. Conf. on Genetic Algorithms*, pp. 73–80. Morgan Kaufman, 1997.

[26] J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997.

[27] A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184, 1998.

[28] B.-T. Zhang, P. Ohm, and H. Mühlenbein. Evolutionary induction of sparse neural trees. *Evolutionary Computation*, 5(2):213–236, 1997.