UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

A Hybrid Approach to Feature Selection and Generation Using an Evolutionary Algorithm.

Oliver Ritthoff,
Ralf Klinkenberg,
Simon Fischer,
Ingo Mierswa

No. CI-127/02

Technical Report ISSN 1433-3325 February 2002

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

A Hybrid Approach to Feature Selection and Generation Using an Evolutionary Algorithm

Oliver Ritthoff, Ralf Klinkenberg, Simon Fischer, and Ingo Mierswa

Chair of Artificial Intelligence, Department of Computer Science,
University of Dortmund, 44221 Dortmund, Germany
{ritthoff,klinkenberg,fischer,mierswa}@ls8.cs.uni-dortmund.de

Abstract. Genetic algorithms proved to work well on feature selection problems where the search space produced by the initial feature set already contains the hypothesis to be learned. In cases where this premise is not fulfilled, one needs to find or generate new features to adequately extend the search space. As a solution to this representation problem we introduce a framework that combines feature selection and generation in a wrapper based approach using a modified genetic algorithm for the feature transformation and an inductive learner for the evaluation of the constructed feature set. The basic idea of this concept is to combine the positive search properties of conventional genetic algorithms with an incremental adaptation of the search space. To evaluate this hybrid feature selection and generation approach we compare it to several feature selection wrappers both on artificial and real world data.

1 Introduction

One aspect in machine learning crucial for successfully solving a learning task at hand is the formalism in which the hypotheses (i.e. possible solutions) are represented. Formally, the set of examples $\mathcal{E}$ is specified using a description language $\mathcal{L}_\mathcal{E}$. Single hypotheses h from the set of possible hypotheses $\mathcal{H}$ are described in a hypotheses language $\mathcal{L}_\mathcal{H}$. In conventional machine learning methods, the features used in $\mathcal{L}_\mathcal{E}$ and $\mathcal{L}_\mathcal{H}$ are usually identical.

Two learning tasks that handle the representation problem by properly transforming an inadequate feature space $\mathcal{L}_\mathcal{E}$, are *feature selection* and *feature generation.* [1] Models of *feature selection* [9] assume that the description language contains a superset of the features that are sufficient to describe the target hypothesis. Thus, learning comprises the selection of a feature subset that maximizes the learning performance of a classification or regression task.

Most machine learning methods for classification and regression are already designed to find the most suitable, i.e. relevant features in a given feature set. Thus, feature selection is already implicitly done. Nevertheless one often needs an additional pre-processing step prior to the application of the actual learning

[1] An overview of different approaches in feature abstraction, selection and construction is given in [17].

method. One reason is, that the prediction accuracy of many learning algorithms, including e.g. decision tree learners like C4.5 [21] decreases, when irrelevant[2] or redundant features are added [13]. Another problem particularly affecting the computation time is the lacking scalability of many learning methods, since their applicability significantly decreases on large-scale data sets [19].

Two of the possible dimensions proposed in [7] for categorizing different feature selection algorithms are *search organization* and *evaluation strategy*. With regard to the first dimension, the simplest search strategy is the *exhaustive* search, which guarantees to find the best feature subset[3] but needs exponential runtime and thus is not applicable in most cases. Another class, called *heuristic* search methods, uses an evaluation function that directs the search into areas of increased performance. A subclass of these algorithms, called *hill climbing* methods, incrementally chooses feature subsets that lead to the highest performance increase in one iteration. Two instances of hill climbing methods are *forward selection* and *backward elimination* [1]. Forward selection starts with an empty set of selected features and iteratively adds the feature leading to the highest performance increase to the set of selected features, until the performance cannot be enhanced any further by adding a single feature. Backward elimination starts with the complete feature set and iteratively removes the feature whose removal yields the maximal performance increase. A major shortcoming of such sequential hill climbing methods is their lacking ability to cope with feature interaction, which is one of the main difficulties in feature selection. Feature, or attribute, interaction is characterized as a situation, where the effect of a particular feature on the target depends on the value of other features [10]. A popular probabilistic search method, that can handle these feature interactions are *evolutionary algorithms* [3].

According to the second dimension we can distinguish methods that select feature subsets irrespective of their effect on the learning performance (known as *filter approaches* [2, 14]) from those that use a specific learning method for the evaluation of the feature subsets (known as *wrapper approach* [15]). Descriptions of several hybrid approaches that use evolutionary methods for the feature selection and different learning methods for the evaluation of particular feature sets, especially *neural networks* and *decision trees*, can be found in [20, 4, 28, 22].

In contrast to the learning task of feature selection, models of *feature generation*[4] enrich the hypothesis language with additional constructed and derived features, respectively [18]. Thus, one goal of feature generation and especially of constructive induction is to reveal and accordingly represent feature dependencies explicitly, that cannot be found by the applied learning method alone.

Feature generation and selection are closely related learning tasks. They can be viewed as two sides of the representation problem in machine learning, i.e. the problem of finding an adequate representation language for the learning task at

---

[2] Several definitions of feature relevance are stated in [7].

[3] Provided, that the search space already contains the searched hypothesis.

[4] A special case of feature generation using inductive generalization is also known as *constructive induction*.

hand. In cases, where the given representation language is insufficient to describe the (learning) problem, feature generation can assist in augmenting the original language by appropriately composed features. In cases, where the representation language contains more features than necessary to solve the learning task, feature selection can be used to simplify the language. Since some of the constructed features are irrelevant or redundant, they can be eliminated from the feature set by a proper selection method. Corresponding hybrid approaches that combine feature generation and selection can e.g. be found in [16, 6].

Approaches using a combination of feature generation and selection methods based on probabilistic search strategies have rarely been considered so far. [8, 5, 26] An according framework using bitstrings for the feature representation, GP operators for the feature transformation and C4.5 for the feature evaluation is described in [26]. In contrast to our GA-based wrapper approach, this framework does not provide the use of meta-data for the feature generation process to restrict the resulting search space to useful, i.e. admissible, subspaces and thus to accelerate the search process.

In the following section, we describe our hybrid feature selection and generation approach in more detail. Since we use a modified genetic algorithm adapted to the process of feature transformation, we start with a short introduction into the applied concepts of canonical genetic algorithms. Then, we present the modified genetic operators, comprising mutation, crossover, and an additional type-restricted feature generator. Afterwards, we delineate the feature evaluation concept using an arbitrary induction algorithm. A comparison of this hybrid feature selection and generation approach with several feature selection wrappers, both on artificial and real world data, is presented in the third part of this paper.

2 Architecture

In this section, we introduce a wrapper-based approach using a modified genetic algorithm for the incremental selection and generation of new features, employing an attribute-based induction algorithm for the evaluation of the feature sets at hand. The general idea is to combine the positive search properties of conventional genetic algorithms with the option to adapt the search space incrementally.

The combined approach as stated in Figure 1 is an adapted version of the wrapper-approach presented in [15]. According to our approach, the feature selection and generation by means of a modified genetic algorithm encloses the chosen induction algorithm. [5] The genetic algorithm conducts the search for a good feature subset using the induction algorithm for the evaluation of the current feature subsets. The training data set the induction algorithm is run on, is partitioned into internal training and hold out sets, with different sets of features removed from and added to the data. The process of creating feature sets, using the modified genetic algorithm and evaluating these sets is repeated until a given

[5] The induction algorithm we chose in our experiments was a support vector machine (SVM) [27], although any other inductive learning algorithm could have been used.

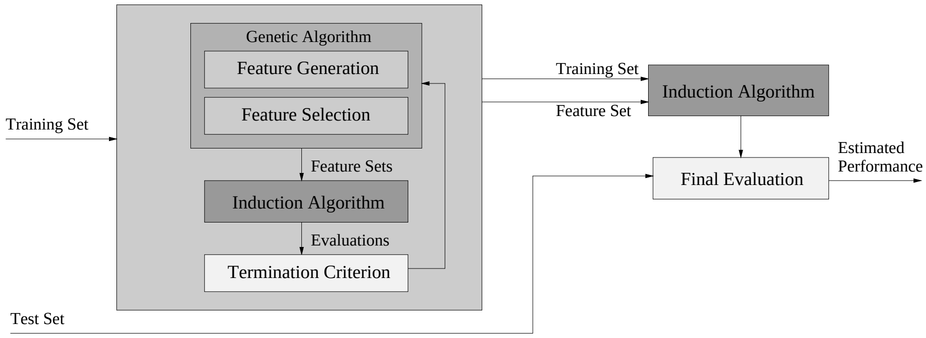

Fig. 1. Combined wrapper-based feature selection and generation approach

termination criterion is fulfilled. This criterion could e.g. be a maximum number of generations, a fixed time limit, the achievement of a preliminary fitness value, or the convergence of the genetic algorithm. The resulting feature set is chosen as the final set on which to run the induction algorithm. The final evaluation of the resulting classifier is done using an independent test set not used during the learning step.

Conceptionally, the key idea in using the incremental feature generator in the context of a genetic algorithm is, that any feature can be generated with a probability $p > 0$ in a finite number of iterations, i.e. generations of the genetic algorithm, given particular feature generators and the original features. Thus, more formally, given a feature set $\boldsymbol{f_t}$ at time point t, a set of feature generators G applicable in one step with $G(\boldsymbol{f_t}) = \boldsymbol{f_{t+1}}$ and f being the target feature, then the following proposition is assumed:

$$f \in \lim_{t \to \infty} G(\boldsymbol{f_t})$$

The following section gives a short introduction into genetic algorithms as they are described e.g. in [12, 11] and presents the modifications that have been conducted on the standard genetic operators to perform the task of feature selection and generation.

2.1 Genetic Algorithms

The canonical genetic algorithm works on an n-tuple of binary strings b_i of length l, where the bits of each string are considered to be genes of an individual chromosome, and where the n-tuple is said to be a population. Following the terminology of biologic evolution the operations performed on the population are called *mutation*, *crossover*, and *selection*.[6] Each individual b_i represents a feasible solution of a given problem and its objective function value $\phi(b_i)$ is said to be its fitness, which has to be maximized. The general framework of

[6] In this context we have to distinguish between the selection of features in a feature set and the selection of individuals in an evolutionary sense.

```
create an initial population
evaluate initial population
repeat
    perform selection
    perform crossover
    perform mutation
    evaluate population
until termination criterion is fulfilled
```

Fig. 2. The main algorithm of a canonical genetic algorithm

a canonical genetic algorithm is shown in figure 2. According to this figure, we first of all create an initial population, which is generally done by setting each bit of a chromosome, or individual, to 1 or 0 with probability 0.5 each. In the next step, the individuals are evaluated based on a given fitness function ϕ. In the main loop we select particular chromosomes for reproduction, vary them by mutation and crossover, and evaluate them using the fitness function. The standard selection scheme is the so-called *fitness proportional selection*. In *fitness proportional selection*, the probability for an individual b_i to be selected for recombination is proportional to its relative fitness value, or more formally given by $p_s(b_i)=\phi(b_i)/\sum_{j=0}^{n}\phi(b_j)$. After n individuals have been selected, the variation operators, mutation and crossover, are applied.

For the *crossover*, it first has to be determined, if a crossover should be applied at all. Therefore we generate a uniformly distributed random variable U in the interval [0,1]. A crossover will only be performed if $U \leq p_c$, with p_c being the given probability of performing a crossover. In this case we randomly chose two individuals from the set of selected individuals for the crossover procedure. The crossover type usually applied in the canonical genetic algorithm is the so-called *one-point crossover*. To apply this operator we first have to determine a crossover point and afterwards exchange the string sections of the two parent strings on the right hand side of this point. The *mutation* operator is the second variation operator and randomly flips single bits on a specific chromosome given a predefined mutation probability. This operation is generally necessary to reintroduce alleles (bit values), which correspond to features in our approach, that converged to a certain value and thus could never be regained by means of simple crossover. The process of choosing two individuals from a set of selected individuals and applying the variation operators on them is repeated $n/2$ times. The whole evolutionary cycle of selection, recombination, and evaluation proceeds until a given termination criterion is fulfilled.

2.2 Adapted Operators

In the previous section we introduced the main algorithm of a canonical genetic algorithm. From a machine learning point of view, genetic algorithms can be seen as a robust search strategy for large hypotheses spaces, where only little

```
create an initial population
evaluate initial population
repeat
    perform selection
    perform variable-length crossover
    perform mutation
    perform feature generation
    evaluate population
until termination criterion is fulfilled
```

Fig. 3. The main algorithm of the modified genetic algorithm

knowledge about the given search space is needed. In order to use genetic algorithms for feature selection and generation, we have to adapt the representation of the original feature space to the evolutionary setting. In our approach, each chromosome is interpreted as a binary representation (bit string) of an underlying feature set, i.e. each gene is associated with a particular feature. A gene value of "1" means, that the corresponding feature is selected in the current feature set, whereas a gene value of "0" represents a deselected feature. For the purpose of applying genetic algorithms to the task of adaptive feature space transformation, we have to partially modify the standard genetic operators *crossover* and *mutation*.

Crossover recombines different individuals, i.e. feature sets, whereas mutation selects, or deselects single features of a particular individual respectively. The idea behind the crossover operator is to combine good feature sets to create even better ones. Since the length of a single individual can vary by adding new (generated) features, we have to modify the standard crossover operator. In contrast to the crossover operator, the standard mutation operator can be adopted without changes. Additionally to the standard genetic operators, we introduce an operator that, given a particular generator, produces new features by combining those selected features in a given feature set that agree with the generator's type restrictions.

Figure 3 shows the main algorithm of our new approach combining feature selection and generation by means of a genetic algorithm. We use fitness proportional selection, combined with an elitist strategy. The elitist strategy ensures that the best individual of a generation remains in the population, i.e. all parents are replaced by the child population, except for the best individual of the current generation. The whole evolutionary cycle of selection, variation, feature generation, and feature set evaluation proceeds, until a given termination criterion is fulfilled.

Adapted crossover operator

Figure 4 shows an adapted version of the standard one-point crossover used in genetic algorithms that can cope with variable length chromosomes, resulting from

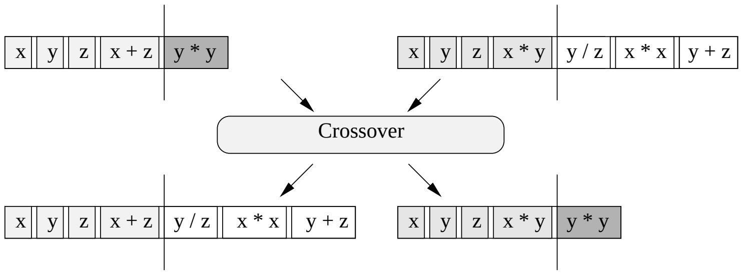

Fig. 4. Modified crossover operator for the adapted genetic algorithm

the application of a feature generator to individual features on a chromosome. The two parent individuals prior to the application of the crossover operator are shown on the top of the figure whereas the two resulting child individuals appear at the bottom. To use the crossover operator, a corresponding crossover point (indicated by the thin vertical lines) has to be determined. Subsequently, the features of the two individuals on the right side of this point are exchanged.

The left parent individual, shown on figure 4, contains the original features x, y, z and two generated features x+z and y*y, the right parent individual contains the original features x, y and z as well as three generated features y/z, x*x and y+z. [7]

Mutation Operator

Figure 5 shows an example for the process of mutating single individuals. In the setting of feature transformation, mutation of an individual by flipping one or more bits on the bit string corresponds to the selection (indicated by a gray shading) or deselection of the respective features (indicated by a white shading). The top of this figure shows an individual, containing seven features, five of which are selected. After mutation, feature y which originally was selected is now deselected and feature x*z which formerly was deselected is now selected.

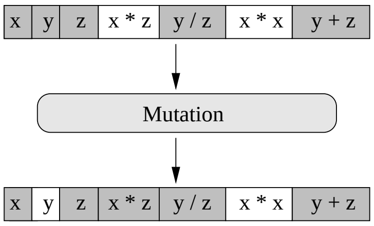

Fig. 5. Mutation operator for the adapted genetic algorithm

[7] The shades of gray indicate different substrings on the chromosomes to clarify the crossover process and should not be mistaken with the shades of gray on figures 5 and 6 that denote the selection and deselection of features in a feature set.

2.3 Type-restricted Feature Generation

Conceptionally, the variation operators crossover and mutation focus the search for a good feature set on different subspaces of the original feature space, whereas feature generation enlarges the original space. Given a feature set F, a set of selected features $F_S = \{f_1, .., f_n\} \subseteq F$, and a set of feature generators $G = \{g_1, .., g_k\}$, we first choose a particular feature generator $g_j \in G$ for the generation process. Then, by checking the types of all features in F_S, the compatible feature subsets $\{F_{c_1}, .., F_{c_l}\} \subseteq F_S$ are determined with regard to the type restrictions of the generator at hand. Finally, the chosen feature generator g_j is applied to the set of compatible features (or a subset thereof) and the resulting features $W_i = g_j(F_{c_i})$ are added to the original feature set F. The set of compatible features is not limited to original features, but can also contain compound features that have already been created by a generator. This property allows recursive feature generation and thus the construction of arbitrarily complex features.

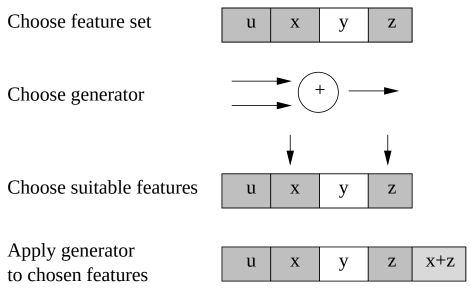

Fig. 6. Feature generation operator used in the combined approach

Figure 6 exemplifies a concrete feature generation process. The top of this figure shows an individual, containing four original features u, x, y and z, whereas only u, x and z are currently selected. The next step is to randomly choose a generator from the predefined set of generators. Feature generators may handle e.g. boolean, mathematical, transcendental or domain specific functions and operate on individual features as well as on entire feature sets. In this case the "+" generator is chosen, which combines two numeric features by simply adding their feature values. After the generator is determined, the corresponding features, in this case two numeric features, have to be chosen. Assuming, that u, x and z are all numeric features and thus fulfill the given type-restrictions, we randomly choose x and z. Finally, we append the generated feature x+z to the end of the bit string, i.e. add it to the original feature set.

Both, the maximal number of features that may be added to one feature set in a single step as well as the set of generators that can be used have to be predetermined by the user. The actual number of created features and the particular generators are determined stochastically.

An obvious advantage of using type-restricted constructors lies in the limitation of the set of constructible features to a well-formed subset. This restriction leads to an acceleration of the search for a good representation, by only extending the search space by useful subspaces.

2.4 Feature Evaluation

To calculate the fitness value of a particular individual, i.e. of a single feature set, and thus to determine the quality of this feature set some kind of evaluation method is required. In order to apply the evaluation method, we have to adapt the example set to the current feature sets, by removing the values for the deselected features and adding the values for newly generated features. This modified example set now serves as the training data for the learning method. The induced model is applied to the test data to evaluate the fitness of the given feature set. This evaluation procedure is embedded into two nested cross validations, where the inner one serves for the determination of the best feature set and the outer one for its overall validation.

3 Experiments

3.1 Artificial Data

To evaluate the presented approach, we first compared it with a genetic feature selection wrapper using an artificial data set. [8] The target function f was given by $f(\boldsymbol{a}) = a_1 + ... + a_5$, i.e. the sum of five given features, containing three basic features a_1 to a_3 and two constructed features $a_4 := a_1 * a_2$ and $a_5 := a_2/a_3$. The performance evaluation was done using two nested cross validations. The inner cross validation was used to find a good feature set, while the outer cross validation was used to evaluate the performance of this feature set on an extra validation set. For both approaches, the parameters for the genetic algorithm were set to 200 for the number of generations, 30 for the population size, 0.5 for the crossover probability, 0.2 for the mutation, i.e. feature selection probability, and 0.5 for the feature generation probability. The induction algorithm we chose was a regression SVM [27, 25, 24] with complexity $C = 1000$ and $\epsilon = 0.1$, using a dot kernel.

The results presented in Table 1 show that the combined approach clearly outperformed the genetic feature selection wrapper in cases of missing relevant features. In cases of different amounts of noise, i.e. additional irrelevant features and even in the optimal case where the original feature set was given, this approach turned out to be superior to the simple selection approach. Furthermore, in the case where no generated features from the original feature set were given in advance, the new approach exactly reconstructed these missing features during the search process.

[8] For the experiments, both on real-world and artificial data, the machine learning environment YALE [23] was used.

Searched feature set	Simple selection approach	Combined approach
Original features (=$a_1, .., a_5$)	0.0517 (0.0231)	0.0473 (0.0198)
Original features + one random feature	0.0538 (0.0201)	0.0537 (0.0179)
Original features + two random features	0.0592 (0.0242)	0.0571 (0.0215)
Original features + three random features	0.0596 (0.0230)	0.0586 (0.0236)
Original features - one constructed feature	1.4569 (0.1612)	0.0738 (0.0893)
Original features - two constructed features	1.4669 (0.1857)	0.0584 (0.0712)

Table 1. Comparing the performance of a simple feature selection approach using a genetic algorithm with the combined approach on an artificial data set in terms of the average absolute error (average relative error in parentheses).

3.2 Real-World Data

This section presents a comparison of the appropriateness of different chains of feature generation and selection methods for an application problem from chromatography. Chromatography is used in chemical industries to separate temperature sensitive substances. A mixture of components is injected for a certain amount of time into a column, filled with porous particles. Due to the different adsorption strengths of the substances on the porous particles, the components have various velocities in the column and reach its end at different times, where they can be separated. The learning task considered here is to predict the four characteristic coefficients of a two component mixture given the corresponding chromatogram time series. Two constants, namely the *Henry* and *Langmuir* coefficients, determine the chromatogram of a substance. The data set contains 200 examples with 5000 features each, i. e. with measurements at 2500 equidistant points of time of the chromatogram time series for each of the two components. Since the learning task again was a regression problem, we used a regression SVM throughout the following experiments. The support vector machine was applied, using a radial-basis kernel with $\gamma = 1$, $C = 1000$, and $\epsilon = 0.1$.[9] The learning performance was evaluated in terms of the absolute and relative prediction error, comparing predicted and real values of the *Henry* and *Langmuir* constants. Based on the structure of the overall learning task, we systematically compared the performance of different learning chains, comprising feature generation and selection. The experimental results are shown in Table 2.

In the first experiment, we simply used the original (time series) features to learn and evaluate an SVM model without any preprocessing steps. The corresponding learning chain, comprising learner, model applier and evaluator, was enclosed by a four-fold cross validation. Due to noise in the simulated measurements of the original features, this operator chain only yielded a poor prediction performance. The second chain additionally contained a pre-processing operator that generated numeric characteristics from the original time series data. These new features include the position (i.e. points of time) and the value (i.e.

[9] These parameter values have been determined in preliminary tests.

Applied operator chain	Henry constant substance 1	Langmuir constant substance 1	Henry constant substance 2	Langmuir constant substance 2	Runtime (sec.)
Original data	2.2044 (0.6285)	21.9080 (0.6497)	2.4718 (0.7046)	22.3290 (0.6883)	31540
Feature generation	0.5415 (0.1551)	17.6390 (0.4695)	0.6143 (0.1671)	17.7408 (0.4645)	5869
Feature generation & FS-wrapper	0.1541 (0.0336)	11.0932 (0.3271)	0.2288 (0.0349)	11.0196 (0.5407)	13086
Feature generation & BE-wrapper	0.0872 (0.0215)	0.1015 (0.0028)	0.0918 (0.0238)	0.1010 (0.0028)	6852
Feature generation & GA-wrapper	0.0733 (0.0193)	0.1052 (0.0028)	0.0748 (0.0232)	0.1015 (0.0029)	79927
Combined approach	0.0737 (0.0187)	0.1015 (0.0027)	0.0777 (0.0199)	0.1012 (0.0028)	44768

Table 2. Average absolute error for the target values *Henry* and *Langmuir* on a two-substance mixture, tested on different operator chains (average relative error in parentheses).

concentration of the substance at the column output) of the maximum of the chromatogram and the x- and y-coordinates of the two inflection points left and right of the maximum. Since the sensor readings may be noisy and perhaps slightly shifted along the time axis among different measurements, an individual feature in the time series of an example, i.e. the concentration measured at one particular point of time, is not very reliable. As expected, the construction of these robust features significantly improved the learning result.

Since it was not obvious, which of the new and which of the original features were really helpful in solving the learning task, an additional feature selection step was performed subsequent to the feature generation step to obtain a feature set well-suited for the given learning task. Different feature selection wrappers, namely *Forward Selection* (FS), *Backward Elimination* (BE) and a *Genetic Algorithm* (GA) were employed reducing the set of features and increasing the learning performance in terms of the absolute and relative prediction error. Finally, we tested our new approach combining feature generation and selection on the given learning problem, using the same parameter settings as in the case of the genetic feature selection wrapper. In this setting, feature generation and selection were not used as subsequent pre-processing steps, but intertwined in a feature wrapper approach. In the case of the genetic algorithm for feature selection we used 50 generations, a population size of 30 individuals, a mutation probability of 0.05, a crossover probability of 0.5, *fitness proportional selection*, and *one-point crossover*. The combined approach yielded a similar prediction performance compared to the feature selection wrapper using a genetic algorithm, but required significantly less runtime.

4 Summary

In this paper, we presented a general framework that connects feature generation and selection in a combined approach using a modified genetic algorithm for feature transformation and an inductive learner for the feature evaluation. To restrict the amount of generated features and thus to accelerate the search process, we introduced a type-restricted feature generation concept. We showed, that in cases where the original feature set is inadequate with regard to the given learning task, one can significantly improve the learning performance by adding relevant features by means of *feature generation* and removing irrelevant features by applying *feature selection* methods.

Although the learning performance in terms of predictive error and learning time significantly improved compared to a simple genetic feature selection wrapper by incrementally adapting $\mathcal{L}_{\mathcal{E}}$ and $\mathcal{L}_{\mathcal{H}}$, the search process could be further enhanced by using prior knowledge about the search space. Therefore the open question remains, how a demand-driven control of the feature generation process, especially for the case of regression problems, could be realized. In this context, demand-driven control means to precisely determine specific feature generators and chose particular features that produce individuals with high fitness values. Another open question is, to what extent a dynamic interaction of feature generation and selection in terms of a controlled sequence of different search space transformations can be realized.

5 Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG), Collaborative Research Center on Computational Intelligence (SFB 531) at University of Dortmund.

References

1. D. W. Aha and R. L. Bankert. A comparative evaluation of sequential feature selection algorithms. In D. Fisher and H.-J. Lenz, editors, *Learning from Data*, chapter 4, pages 199–206. Springer, New York, USA, 1996.
2. H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 547–552, Anaheim, CA, USA, 1991. AAAI Press.
3. T. Baeck, D. B. Fogel, and T. Michalewicz. *Evolutionary Computation 1, Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, UK, 2000.
4. J. Bala, K. A. De Jong, J. Huang, H. Vafaie, and H. Wechsler. Hybrid learning using genetic algorithms and decision trees for pattern classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 719–724, San Francisco, CA, USA, 1995. Morgan Kaufmann.
5. H. Bensusan and I. Kuscu. Constructive induction using genetic programming. In *Evolutionary Computing and Machine Learning Workshop (ICML-96)*, Morgan Kaufmann.

6. E. Bloedorn and R. S. Michalski. Data-driven constructive induction. *IEEE Intelligent Systems*, 13(2):30–37, 1998. Special Issue on Feature Transformation and Subset Selection.
7. A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 1(2):245–271, 1997.
8. E. I. Chang and R. P. Lippmann. Using genetic algorithms to improve pattern classification performance. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 797–803. Morgan Kaufmann Publishers, Inc., 1991.
9. M. Dash and H. Liu. Feature selection for classification. *International Journal of Intelligent Data Analysis*, 1(3):131–156, 1997.
10. A. Freitas. Understanding the crucial role of attribute interaction in data mining. 16(3):177–199, 2001.
11. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, USA, 1989.
12. J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.
13. G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*, pages 121–129, San Mateo, CA, USA, 1994. Morgan Kaufmann.
14. K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 129–134, Menlo Park, CA, USA, 1992. AAAI Press.
15. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence Journal, Special Issue on Relevance*, 97(1–2):273–324, 1997.
16. N. Lavrac, D. Gamberger, and P. D. Turney. A relevancy filter for constructive induction. *IEEE Intelligent Systems*, 13(2):50–56, 1998.
17. H. Liu and H. Motoda. *Feature Extraction, Construction, and Selection: A Data Mining Perspective*. Kluwer, Dordrecht, NL, 1998.
18. R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume 1, chapter 4, pages 83–134. Morgan Kaufmann, Palo Alto, CA, USA, 1983.
19. G. Paliouras. The scalability of machine learning algorithms. Master thesis, Department of Computer Science, University of Manchester, Manchester, UK, 1993.
20. W. Punch, E. Goodman, P. Hovland, and R. Enbody. Further research on feature selection and classification using genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 557–564, Palo Alto, CA, USA, 1993. Morgan Kaufmann.
21. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Machine Learning. Morgan Kaufmann, San Mateo, CA, USA, 1993.
22. T. Ragg and S. Gutjahr. Neural network optimization by searching guided by stochastic methods. In *Proceedings of the EUFIT'98 Conference*, Aachen, 1998. Mainz Verlag.
23. O. Ritthoff, R. Klinkenberg, S. Fischer, I. Mierswa, and S. Felske. YALE: Yet Another Machine Learning Environment. In R. Klinkenberg, S. Rüping, A. Fick, N. Henze, C. Herzog, R. Molitor, and O. Schröder, editors, *LLWA 01 – Tagungsband der GI-Workshop-Woche* Lernen – Lehren – Wissen – Adaptivität, Technical Report No. 763, Department of Computer Science, University of Dortmund, pages 84–92, Dortmund, Germany, October 2001.

24. S. Rüping. *mySVM-Manual.* Artificial Intelligence Unit, Department of Computer Science, University of Dortmund, Germany, 2000. http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.
25. Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. NeuroCOLT2 Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.
26. H. Vafaie and K. A. De Jong. Feature space transformation using genetic algorithms. *IEEE Intelligent System*, 13(2):57–65, 1998. Special Issue on Feature Transformation and Subset Selection.
27. V. N. Vapnik. *Statistical Learning Theory.* Wiley, Chichester, United Kingdom, 1998.
28. J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 380–385, San Mateo, CA, USA, 1997. Morgan Kaufmann.