Genetic algorithms in constructive induction

Jacek Jelonek Maciej Komosinski

Institute of Computing Science, Poznan University of Technology
Piotrowo 2, 60-965 Poznan, Poland
{jacek.jelonek, maciej.komosinski}@cs.put.poznan.pl

Abstract. In this paper, genetic algorithms are used in machine learning classification task. They act as a constructive induction engine, which selects features and adjusts weights of attributes, in order to obtain the highest classification accuracy. We compare two classification approaches: a single 1-NN and a n^2 meta-classifier. For the n^2-classifier, the idea of an improvement of classification accuracy is based on independent modification of descriptions of examples for each pair of n classes. Finally, it gives $(n^2 - n)/2$ spaces of attributes dedicated for discrimination of pairs of classes. The computational experiment was performed on a medical data set. Its results reveal the utility of using genetic algorithms for feature selection and weight adjusting, and point out the advantage of using a multi-classification model (n^2-classifier) with constructive induction in relation to the analogous single-classifier approach.

Keywords: evolutionary computation, machine learning, knowledge representation, feature selection, multi/meta-classification systems.

1 Introduction

The main problem concerning usage of any classification system is definition of a set of features, which allows obtaining the highest possible classification accuracy. In many practical applications of machine learning, such as pictorial image classification, speech recognition, identification tasks etc., it is easy to suggest (or even automatically generate) many features. However, only a part of them is usually relevant. An influence on the accuracy of those remaining may be not significant; sometimes using them may even deteriorate the classification result.

In order to solve that problem many feature selection algorithms have been developed [2], [14], [23], [10]. In general, feature selection can be treated as a search problem. Each state in the search space represents a subset of possible features. Following the typical view of feature selection algorithms [2] one can define: *search algorithm* – which looks through the space of feature subsets; *evaluation function* – used to evaluate examined subsets of features; and *classifier* – which is constructed basing on final subset of features. These elements can be integrated in two basic ways forming *filter* or *wrapper model* [4]. In the filter model, features are selected as a pre-processing step before a classifier is used, depending on the properties of the data itself. In the wrapper model, the search algorithm conducts a search for a good subset of features using a classifier itself as the evaluation function. Evaluation is usually done by estimating classification accuracy.

Feature selection and attribute weighting can be treated as the basic methods belonging to *constructive induction* (CI) methodology – regarded as supporting automatic, problem-oriented transformation of representation space to facilitate learning [21], [18], [24]. An improvement of accuracy in CI is usually obtained by construction of new features, modification of existing ones and reduction of irrelevant ones. Most systems use specific techniques within one basic computational method: data-driven, hypothesis-driven or knowledge-driven [24].

In this paper we employ a multi-classification system, n^2-classifier (proposed by Jelonek and Stefanowski [9]), in a constructive induction framework. The n^2-classifier is composed of $(n^2 - n)/2$ binary base classifiers (n is a number of classes). Each base classifier is specialized to discriminate one pair of decision classes. A quite similar approach was independently proposed by Friedman [6], and later extended and modified [8], [19]. Our previous experiments have shown that the n^2-classifier with inherent capability of reducing irrelevant features usually yields better classification results than the analogous, single classification model [10], [12]. This observation led us to the hypothesis that creation of dedicated sets of features (e.g. by CI) for discrimination of pairs of classes by independent classifiers gives higher accuracy than using a single set of features dedicated for classification of all classes simultaneously. That was verified on a medical data set in [13], where a genetic algorithm was used for feature selection. Here we extend the capabilities of a genetic algorithm, so that it can not only select features, but also adjust their weights in a nearest-neighbor classification and the wrapper model.

The final version of this paper appeared in Foundations of Intelligent Systems, ISMIS 1999, LNCS, vol 1609. https://doi.org/10.1007/BFb0095156.

To sum up, we compare here a constructive induction process for both single classifier and n^2-classifier. CI is performed by a genetic algorithm, which provides a global, evolutionary technique for finding best descriptions of examples [15]. In the n^2-classifier case, CI process is repeated $(n^2 - n)/2$ times in order to obtain best-discriminating description spaces for all pairs of classes. Finally, the best sets of weighted features found during the GA runs are used for constructing the n^2-classifier. Our computational experiment based on medical data set confirms the advantage of using a multiple CI algorithm instead of a single CI algorithm. Furthermore, the results obtained show that extending the capabilities of genetic algorithm by weight adjusting yields greater improvements in n^2-classifier, than in a single classifier.

The paper is organized as follows: in the next section we describe a constructive induction framework based on the feature selection. Going into details in subsections 2.1 and 2.2 we present two main components of the CI loop, i.e. *function optimization engine* (a genetic algorithm) and *features evaluation* obtained by using a single or a multi-classification model. The details of computational experiments are included in section 3, where we describe the medical data set, our implementation of the classifiers, the parameters of genetic algorithms and the results of a single and a multi-classifier approach. Section 4 groups final remarks and conclusions.

2 Constructive induction based on feature selection

Figure 1 shows a typical loop of a constructive induction framework. In this scheme function optimization is based on maximization of classification accuracy by modification of input data space (feature selection and their weighting is applied here). In each iteration the function optimization engine calls the features evaluation module to estimate the quality of the features. We applied genetic algorithms as a function optimization engine and a typical *k*-nearest neighbor classifier (1-NN) as features evaluator [1]. The 1-NN algorithm was used both in a single and in a multi-classifier (n^2-classifier) model.

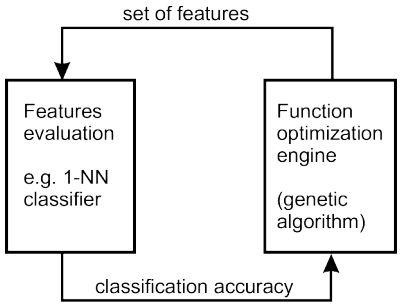

Fig. 1. Main loop of a constructive induction framework

2.1 Genetic algorithms for function optimization

Genetic algorithms (GA) have been successfully applied to many difficult optimization problems. This search method was proved efficient and robust [7]. GA simulate natural mechanisms, like selection and reproduction, while processing a *population* of *individuals* – each individual representing one solution. The algorithm is iterative – in each *generation* best individuals are promoted (*selection*), information between them is exchanged (*crossover*) and *mutation* takes place [5]. Those operations are conducted in order to improve fitness of the next population and to avoid premature convergence to local optima. The operations are usually non-deterministic – they happen with certain probabilities. Despite this non-determinism, GA manage to find the neighborhood of optimal solutions where other search algorithms cannot be applied or do not work effectively [20].

Genetic algorithm is a good tool for feature selection problems. It is not a greedy algorithm – it can search the space of solutions in a global manner, discovering important features and their weights, and passing them to future generations. In our case, each individual represents a weight vector (each weight corresponds to one attribute). In a feature selection task, weights can be equal to 0 or 1 only. Evaluation of each individual (a vector of weights of features) is done by estimating the classification accuracy it yields. Best individuals are the best weight vectors – the weights that give the highest classification accuracy in the classification problem.

2.2 Classifiers

2.2.1 Single classifier

In the wrapper model, the classifier is used to estimate sets of features. It is called every time an individual is evaluated. In GA, a population of individuals is evaluated in each generation, so the classifier should be quick. As we want the CI to be performed by the GA only, the classifier should be simple – should not adapt to the data, remove redundant features nor modify them (i.e., by weighting). If it did, the classification accuracy would not concern original features, but an altered solution.

This is why we used 1-NN classifier (nearest neighbor). In a single-classifier approach, 1-NN is the main classifier. In the n^2-classifier approach, 1-NN is used as a binary classifier which distinguishes between pairs of classes.

2.2.2 The n^2-classifier

The n^2-classifier belongs to a group of multiple learning algorithms dedicated to solving multiclass learning problems [4], [16]. The main principle of the n^2-classifier is the discrimination of each pair of the classes: (i, j), $i, j \in \langle 1..n \rangle$, $i \neq j$, by an independent binary classifier $C_{i,j}$. Each base binary classifier $C_{i,j}$ corresponds to a combination of two classes: i and j only. Therefore, the training of each base classifier $C_{i,j}$ consists in presenting it with a subset of the entire data set that contains examples from class i and j only. The classifier $C_{i,j}$ yields a binary classification indicating whether a new example, x, belongs to class i or j. Let $C_{i,j}(x)$ be the classification of example x by the base classifier $C_{i,j}$. We assume that if $C_{i,j}(x)$ equals 1, then the example x is classified by $C_{i,j}$ to class i, otherwise ($C_{i,j}(x) = 0$) x is classified to class j.

The complementary classifiers: $C_{i,j}$ and $C_{j,i}$ (where $i, j \in \langle 1..n \rangle$, $i \neq j$) solve the same classification problem. Their predictions can be computed as the opposite predictions of their complementary equivalents:

$$C_{i,j}(x) = 1 - C_{j,i}(x) . \tag{1}$$

The final decision is obtained by aggregation of votes of binary classifiers. In order to accomplish this, the credibility of each base classifier is estimated. It can be done in several ways, however in this study we associate with each classifier $C_{i,j}$ its credibility coefficient, $P_{i,j}$, defined in the following way:

$$P_{i,j} = \frac{v_i}{e_j + v_i} , \tag{2}$$

where e_j is a number of misclassified examples from class j, and v_i is a number of correctly classified examples from class i. The computation of the credibility coefficients is performed during the learning phase of the classification task (in our case, it is done on the training data set, during the feature selection stage).

The aggregation rule should take into account the reliability of each base classifier. We decided to treat the credibility coefficients $P_{i,j}$ as weights of the sum of the responses yielded by base classifiers for each row of classifier matrix **C**. Now, the whole process of classification can be described in the following steps:

1. Apply example x to all classifiers, $C_{i,j}$ ($i < j$), and obtain their predictions $C_{i,j}(x)$.
2. Determine classification of the remaining classifiers, $C_{i,j}$ ($i > j$), using formula (1).
3. For each class i, $i \in \langle 1..n \rangle$, compute weighted sum:

$$S_i = \sum_{\substack{j=1; \\ j \neq i}}^{n} P_{i,j} \cdot C_{i,j} . \tag{3}$$

4. Find the greatest value of S_i and return its index, i, as a final classification (break ties arbitrarily in favor of the class that comes first in the class order).

In the context of constructive induction, presented in this paper, the n^2-classifier acts both as a framework in which feature selection is performed and as the final classifier.

3 Experiments

3.1 Data set

The data set used in our experiment is composed of 700 histological images. The images have been collected and classified by Janusz Szymaś, medical senior expert from the Department of Pathology, University School of

Medicine in Poznan. The whole data set includes 50 images for each of 14 classes of brain tumors (*tumors of neurepithelial tissue*). Each class is represented by 10 images of 5 patients. To avoid undesirable appearance of the data, coming from the same patient in learning and testing data set, we were obliged to perform a 5-fold cross validation technique. Neglecting this restriction leads to overoptimistic results, as it was shown in [11].

Our previous experiments showed that the histogram of intensities of three principal color components (RGB – red, green, blue) is a simple feature which gives a relatively good classification results. Thus, we have 3 (R, G, B) × 256 attributes. After a simple quantization (replacing each of four consecutive values by their sum), we obtained 192 features. 68 of them were redundant (equal to zero for all cases). Finally, our data set contains 700 objects described by 124 attributes.

3.2 Implementation

In our genetic algorithm, each individual is represented by a genotype string of 124 digits. In the feature selection task, the genes are binary: '1' gene means that the corresponding attribute is chosen, '0' means that it is discarded. In a weight-adjusting task, each gene can be a digit '0' to '9', which is the integer weight of the corresponding attribute. In a single-classifier approach, there is only one run of GA – we look for features that discriminate all the classes best. In a n^2-classifier approach, each run is to find the optimal features subset for a binary classifier, which distinguishes between two classes only. $(n^2 - n)/2$ classifiers and as many evolutionary runs are needed. In our case, for 14 classes, we need 91 binary classifiers.

We carried out separate experiments with a single classifier and n^2-classifier. For each of those approaches, we examined feature selection without attribute domain normalization, feature selection with normalization, and attribute weighting with normalization. The attribute normalization process (adjusting the domain of each attribute to a [0,1] interval) was very time-consuming, so we had to shorten the corresponding evolutionary runs.

Most of the genetic algorithm's parameters were identical for all runs and based on our previous experiences. Population size was constant (80 individuals for experiments without normalization and 40 individuals for experiments with normalization, due to the time constraints). Selection was carried out according to the remainder with repetitions rule. To achieve more stable and certain evaluation of individuals, cross-validation tests were repeated 10 times and the mean value was computed. Despite this, due to the random nature of cross-validation tests, the mean could vary a few percent depending on a fold distribution during those 10 repetitions. In a n^2-classifier experiment, the mean varied more because the number of examples was much smaller (only two classes for a binary classifier).

Crossing-over probability was 100%. Two-point crossing-over was used. For feature selection task, mutation was a simple bit-flipping (0 to 1 or 1 to 0). For feature weighting, mutation turned a gene to 0 when it was greater than 0, or set it randomly to a 1-9 digit when it was 0.

In our experiments, mutation probability was variable during the evolution. It was 5% in the beginning to diversify the population and prevent premature convergence, and decreased linearly to 1/(population size) [7] to allow more precise tuning of solutions. However, we do not expect convergence to a single solution; we keep track of the best individual found so far during the evolution. This is an *off-line* process.

Standard deviation (sigma) truncation scaling was used. The scaling coefficient decreased non-linearly from 5 in the beginning to 1 at the end of the evolution. Non-linear decrease yields a shorter low-pressure period and a longer high-pressure period.

3.3 Results

3.3.1 Single classifier

The evolution in "feature selection with no attribute normalization" task was supposed to last at most 500 generations. Mutation probability reached 1/80 after 50 generations, z reached 1 after 500 generations. The evolution lasted less than 500 generations, and was finished because there was no improvement during the last 100 generations.

Full search space for feature selection consisted of 2^{124} solutions (more than 10^{37}). In 500 generations and with 80 individuals, we examine 500×80 solutions only, which is less than 10^{-33} of the full space. For a weight-adjusting problem, the full search space size is 10^{124}.

The experiments with normalization of attribute domains were more complex and time consuming, and so the populations were smaller and the evolutionary runs shorter. The results are shown in Table 1 (best classification accuracy and standard deviations).

3.3.2 n^2-classifier

In this multi-classifier approach, we needed 91 evolutionary runs (one for each pair of classes). Figure 2 presents best-so-far classification accuracy for some sample binary classifiers, each distinguishing between a pair of classes.

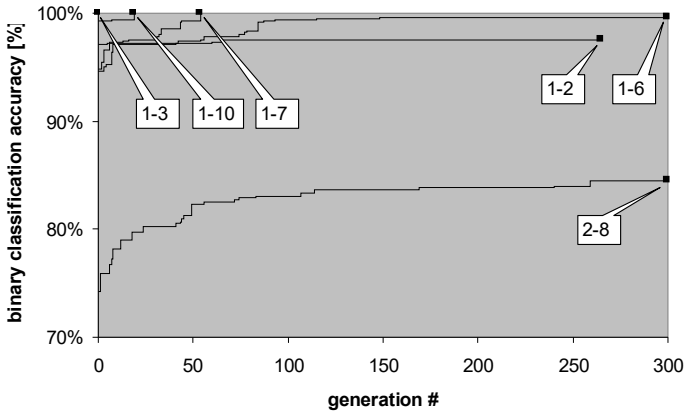

Fig. 2. Evolution of subsets of features in a n^2-classifier model (best-so-far accuracy).

There was no need to evolve discriminant subsets of features for classes 1 and 3; in the first generation, a random subset of features could separate them with 100% accuracy. Pair 1-10 could be 100%-discriminated after a few generations only. Pair 2-8 was hard to distinguish; classes 1 and 6 were easier, but not perfectly distinguishable. The evolution of subsets of features for pair 1-2 was finished due to no improvement in the last 200 generations.

The whole n^2-classifier system yielded 75% classification accuracy in feature selection task; this was still improved in the experiment with attributes' weighting. Table 1 summarizes all results.

Table 1. Classification accuracy for a single and n^2-classifiers

Data characteristics	Classification accuracy [%]		
	Single classifier	n^2-classifier	Improvement n^2 vs. single
all features no normalization	35.34 ± 0.97	35.81 ± 1.20	-
feature selection no normalization	49.89 ± 1.32	75.03 ± 0.88	25.14
feature selection normalized	52.49 ± 0.87	74.05 ± 1.38	21.56
weighting normalized	53.00 ± 0.74	81.61 ± 0.81	28.61

4 Conclusions

The experiments we have performed show the ability of using evolutionary algorithms for constructive induction tasks. The genetic algorithm described in this paper is capable of both selecting features and adjusting their weights for best classification accuracy. The latter capability gives the evolution more freedom and seems promising (even better results could have been achieved if the evolutionary runs lasted longer and had more individuals; that was not carried out because of time limitations).

It can be easily seen that the main improvement was achieved by using CI within a n^2-classifier. In this meta-classifier, letting the GA adjust weights improved the classification accuracy by more than 6%, comparing to feature selection. A disadvantage of the meta-classifier may be its high computational complexity (which depends on the complexity of the base classifiers).

Our future work will concern various improvements of the genetic algorithm (genetic operators, parameters, etc.), and comparison with another algorithms for feature selection, weight adjusting and CI. We are working on implementation of a two-level cross-validation, so that evaluation of individuals during the evolution will be done by a cross-validation in a learning subset of the whole data set. Such architecture of the CI system may

bring more objective evaluation of its performance and results, but will also be much more complex and time consuming.

It would be worthwhile to extend our approach by more "constructive", evolutionary induction (including creation of new attributes, rules etc. [15]). A tradeoff between the number of features, their easy interpretation and classification accuracy could also be introduced. This approach should also be tested on more known data sets to validate its promising utility.

Acknowledgments

The computational experiments have been carried out at the Poznan Supercomputing and Networking Centre affiliated to the Institute of Bioorganic Chemistry at the Polish Academy of Science. This work has been supported from State Committee for Scientific Research, KBN research grant no. 8T11C 013 13 and from CRIT 2 – Esprit Project no. 20288.

References

1. Aha, D.W., Kibler E., Albert M.K.: Instance-based learning algorithms. Machine Learning **6** (1991) 37-66.
2. Aha, D.W., Bankert R.L.: Feature Selection for Case-based Classification of Cloud Types: An Empirical Comparison, *Proceedings AAAI-94 Workshop Case-Based Reasoning*, (1994), 106-112.
3. Bloedorn, E., Michalski, R.S, Wnek, J.: Multistrategy Constructive Induction: AQ17-MCI. *Proceedings of the 2nd International Workshop on Multi-Strategy Learning*, Harper's Ferry, WV, (1993).
4. Chan, P.K., Stolfo, S.J.: Experiments on multistrategy learning by meta-learning. *In Proceedings of the Second International Conference on Information and Knowledge Management*, (1993), 314-323.
5. Davis, L.: *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, (1991).
6. Friedman, J.H.: Another approach to polychotomous classification, Technical Report, Stanford University, (1996).
7. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Co. (1989).
8. Hastie, T., Tibshirani R.: Classification by pairwise coupling, *Proc. NIPS97*.
9. Jelonek, J., Stefanowski J.: Using n^2-classifier to solve multiclass learning problems. Technical Report, Poznan University of Technology, (1997).
10. Jelonek, J., Stefanowski J.: Feature subset selection for classification of histological images, *Artificial Intelligence in Medicine* **9** (1997) 227-239.
11. Jelonek, J., Krawiec, K., Słowiński, R., Stefanowski, J., Szymaś, J.: Computer-assisted diagnosis of neurepithelial tumours based on clinical and pictorial data, *Computers in Medicine* **4** (1997) 170-175, Polish Society of Medical Informatics, Łódź.
12. Jelonek, J., Stefanowski J.: Experiments on solving multiclass learning problems by n^2-classifier, *Proceedings 10th European Conference on Machine Learning*, Chemnitz, Germany, (1998).
13. Jelonek, J., Komosinski, M.: Using n^2-classifier in constructive induction. In: Ch. Giraud-Carrier, M. Hilario (eds.), *ECML '98 Workshop on Upgrading Learning to the Meta-Level: Model Selection and Data Transformation*, Chemitzer Informatik-Berichte, Chemnitz, (1998), 21-29.
14. John, G., Kohavi R., Pfleger K.: Irrelevant features and the subset selection problem, *In Proceedings 11th International Machine Learning Conference*, (1994), 121-129.
15. Komosinski, M.: ECIS – Evolutionary Constructive Induction System, https://www.cs.put.poznan.pl/mkomosinski/ecis/
16. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. *Information and Computation*, 108 (2), (1994), 212-261.
17. Matheus, Ch.J.: The need for constructive induction, *In Proceedings 8th International Workshop on Machine Learning*.
18. Matheus, C.J. and Rendell, L.: Constructive Induction on Decision Trees, *Proceedings of IJCAI-89*, pp. 645-650, Detroit, MI, (1989).
19. Mayoraz, E., Moreira, M.: On the decomposition of polychotomies into dichotomies, *In Proceedings 14th International Conference on Machine Learning*, (1997), 219-226.
20. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, (1996).
21. Michalski, R.S.: Pattern recognition as knowledge-guided computer induction, Department of Computer Science Reports, No. 927, University of Illinois, Urbana, June 1978.
22. Michalski, R.S., Tecuci, G.: *Machine Learning. A multistrategy approach*. Volume IV, Morgan Kaufmann (1994).
23. Pudil, P., Novovicova J., Kittler J., Floating Search methods in feature selection, *Pattern Recognition Letters* **15** (1994) 1119-1125.
24. Wnek, J. and Michalski, R.S.: Hypothesis-driven constructive induction in AQ17: a method and experiments, *Proceedings of IJCAI-91*, Workshop on Evaluating and Changing Representation in Machine Learning, Sydney, Australia, (1991).