To appear in: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), 2002 (Coevolution Workshop)

Coevolutionary Construction of Features for Transformation of Representation in Machine Learning

Bir Bhanu
Center for Research in Intelligent Systems
University of California
Riverside, CA 92521
bhanu@cris.ucr.edu

Krzysztof Krawiec
Institute of Computing Science
Poznan University of Technology
Piotrowo 3A, 60965 Poznan, Poland
krawiec@cs.put.poznan.pl

Abstract

The main objective of this paper is to study the usefulness of cooperative coevolutionary algorithms (CCA) for improving the performance of classification of machine learning (ML) classifiers, in particular those following the symbolic paradigm. For this purpose, we present a genetic programming (GP) -based coevolutionary feature construction procedure. In the experimental part, we confront the coevolutionary methodology with difficult real-world ML task with unknown internal structure and complex interrelationships between solution subcomponents (features), as opposed to artificial problems considered usually in the literature.

1 INTRODUCTION

Representation of knowledge and external stimuli is the key issue in the area of intelligent systems design. An inappropriate representation of the external world may seriously limit the performance of an intelligent agent, whereas a carefully designed one can significantly improve its operation.

This principle affects in particular machine learning (ML), a branch of artificial intelligence dealing with automatic induction of knowledge from data (Langley, 1996; Mitchell, 1997). Many ML classifiers do not perform well on some problems due to their limited ability to construct an internal representation of external inputs, i.e. values of attributes that describe instances to be classified. That affects mostly the classifiers implementing the symbolic paradigm of knowledge representation, like decision trees or decision rules.

On the other hand, there are several classification approaches, which do not suffer from this deficiency. Most of them represent the non-symbolic and/or sub-symbolic paradigm of knowledge representation and processing. For instance, neural nets, discriminant analysis and support vector machines are able to benefit from the synergy of selected attributes (e.g. by building linear combinations of their values). A rich internal representation allows for better discrimination between decision classes. Consequently, as far as the accuracy of classification is concerned, sub-symbolic ML systems often outperform the symbolic ones (see, for instance, (Lim et al. 2001)). However, the price we usually pay for that is the inexplicability of knowledge acquired by the classifier and, in particular, incomprehensibility of the internal representation.

In this paper, we continue our former research on GP–based change of representation for machine learners. The evolutionary process constructs new features, deriving them from the original ones, and searches for a suboptimal set of them. Evolving GP individuals encode feature definitions expressed as LISP-like expressions. The constructed features have therefore symbolic, comprehensible definitions. In particular, the central topic of this paper is the applying the cooperative coevolution to the feature construction task.

The following sections outline the background for the proposed method, the method itself (Section 4), and present the related work (Section 3). Then, an extensive computational experiment is described in Section 5 and its results are discussed in Section 6.

2 FEATURE CONSTRUCTION FOR CHANGE OF REPRESENTATION IN MACHINE LEARNING

Topics described in this study refer to several concepts and research directions known in artificial intelligence (AI), machine learning (ML) and related disciplines. The crucial role of representation has been appreciated in AI already in its infancy. That awareness was present also in ML community (e.g. Chapter 1.3 of (Langley, 1996)), where the representation problem appears at least at two points: in the context of *input representation* and in the context of *hypothesis representation* (referred also to as hypothesis *language* representation). In this paper, the former one is of interest.

In particular, we will focus here on the paradigm of learning from examples and attribute-value representation of input data (Mitchell, 1997). In this setting, the original representation is made up of a vector of *attributes* (features, variables) F_0 describing *examples* (instances, objects). This representation is the starting point for the process of representation transformation, which can be posed as follows: given the original vector of features F_0 and the training set L, invent a derived representation F better than F_0 with respect to some criteria. Undoubtedly, the criterion most often considered at this point is the predictive accuracy or, to be more precise, the accuracy of classification on the test set (denoted hereafter by T). The other measure mentioned relatively frequently is the size of the representation, i.e. $|F|$, however in this paper we will focus on the former one.

The approaches embedded in such environment and reported in the literature can be roughly divided into three categories:

- Feature *selection* methods (also referred to as *variable* selection). Here, the resulting representation is a subset of the original one, i.e. $F \subseteq F_0$.
- Feature *weighting* methods. In this case, the transformation method assigns weights to particular attributes (thus, formally the representation does not change here, i.e. $F=F_0$). The weight reflects relative importance of an attribute and may be utilized in the process of inductive learning. Unfortunately, the group of inductive learning methods that can successfully benefit from these extra data is rather limited and encompasses mostly distance-based classifiers (see (Dash & Liu, 1997) for an extensive overview and experimental comparison of various methods).
- Feature *construction* methods. Here, new features are invented and defined (in some language) as expressions, which refer to the values of original ones.

Principally, each of approaches listed here encompasses its predecessors. For instance, feature selection may be regarded as a special case of feature weighting with constrained domains of weights (e.g. to {0,1} set). Analogously, feature construction usually does not forbid creating features being ‘clones’ of the original attributes (i.e. $F_0 \subseteq F$), what is essentially equivalent to feature selection.

The subject of this study is the feature construction as the most general and, therefore, the most promising approach to representation transformation. According to (Matheus, 1989), feature construction may be further subdivided into constructive *compilation* and constructive *induction* of features. Feature compilation consists in re-writing the original representation in a new, usually more compact way, so the result is logically equivalent to the original. Constructive induction (CI) goes further and takes into account the inductive characteristics of learning from examples, which is inherently bounded with the incompleteness and/or limited representativeness of the training set. Therefore, CI enables building features that are not necessarily supported by the training set, but may potentially improve the predictive accuracy of the classifier.

A more precise taxonomy depending on the type of control mechanism used for feature construction has been introduced in (Michalski, 1983). In particular, in *data-driven* constructive induction (DCI) the input data (training examples) guides the feature construction process. In *hypothesis-driven* constructive induction (HCI), the feature construction process benefits from the form of the induced hypothesis. The methodology described further in this paper represent both aforementioned paradigms.

3 EVOLUTIONARY COMPUTATION FOR REPRESENTATION TRANSFORMATION

Evolutionary computation has been used in machine learning for quite a long time. Contemporarily it is recognized as a useful engine for many ML problems or even as one of its paradigms (Langley, 1996; Mitchell, 1997). It is highly appreciated due to its ability to perform global parallel search in the solution space with low probability of getting stuck in local minima.

Evolutionary computation is basically applied for learning in one of the following ways:

- Individuals (or groups of individuals) implement complete ML classifiers.
- Evolutionary search is responsible only for a part of learning process, for instance for feature selection.

Although the former case (complete concept induction) belongs to the most widely known applications of evolutionary computation in ML (with classifier systems (Goldberg, 1989) as probably the most prominent accent; also (De Jong et al., 1993)), some efforts have been made in the domain of representation transformation (see previous section).

Most research on GP-based change of representation for ML learners reported so far in literature focuses on feature selection and feature weighting. In our previous study on GA-based feature selection and weighting (Komosinski & Krawiec, 2000) we noted a significant improvement in accuracy of classification in an experiment concerning medical image analysis and feature extraction. Also experiments reported in (Raymer et al., 2000) proved usefulness of GA-based feature selection and feature weighting, which yield increase of accuracy of classification on three different ML domains, requiring only a fraction of available attributes (however, authors admit that they conducted some preliminary experiments to determine run parameters). Other experiments, reported for instance in (Vafaie & Imam, 1994), led to similar conclusions.

The topic of evolutionary feature *construction* received more modest attention in the literature. One of the first attempts to apply GP to feature construction for machine learners were reported in (Bensusan & Kuscu, 1996). In (Kishore et al., 2000) an evolutionary approach to multi-category pattern classification has been proposed and a GP-based classifier has been applied to the problem of remotely sensed satellite data.

4 COOPERATIVE COEVOLUTION FOR CHANGE OF REPRESENTATION

4.1 GENETIC PROGRAMMING FOR FEATURE CONSTRUCTION

In this study we employ GP for constructive induction of features. Therefore, we do not expect the evolutionary computation to do the entire task of concept induction. On the other hand, we attack the most advanced form of representation transformation (feature construction). Thus, the proposed methodology may be considered as located on the boundary between the two approaches mentioned above.

Our rationale for such choice is as follows. Firstly, feature selection and weighting do not offer much as far as the change of representation is concerned. On the other hand, some experience we gained from experimenting with GP-based visual pattern classification (Krawiec, 2001) and GP-based ML feature construction (Krawiec, 2002) led us to the conclusion that in most real-world cases it is rather unreasonable to expect the GP individuals to evolve to complete, well-performing classifiers, even for the two-class discrimination problem. Therefore, the GP-based constructive induction of features seems to be a good compromise.

Let us introduce the background more fomally. It is assumed that all examples $x \in L$ are described by the set of features F_0, which will be further referred to as *original* features (or *variables*), as opposed to the features constructed further in the search. The evolving individuals encode feature definitions f_j, $j = 1..n$. Each feature $f_j \in F$ is defined by a LISP-like expression built from the values of original features F_0, in the manner usual for GP (Koza, 1994). Therefore, an individual consists of a vector of GP expressions. Given a training instance $x \in L$ and the values of original features F_0 that describe it, an individual is able to compute the values of feature(s) F it implements.

4.2 PROBLEM DECOMPOSITION

The task outlined above belongs undoubtedly to the complex ones. That complexity manifests itself, among others, in the fact that *many* features are required to obtain competetive accuracy of classification (fitness); when facing real-world problems, no one expects reasonable results by constructing just one feature. It is the features' synergy that makes the representation useful.

Coevolution is at least for decade reported as an interesting approach to handle the increasing complexity of problems posed in artificial intelligence and related disciplines. In particular, its collaborative variety, the cooperative coevolution algorithms (CCA) (Potter & De Jong, 2000), besides being appealing from the theoretical viewpoint, has been reported to yield interesting results in some experiments (Wiegand et al., 2001).

These reports encouraged us to consider the use of CCA to the task of representation transformation by feature construction. In particular, we expected the CCA to cope better with the feature development for inductive learners than the plain evolutionary algorithm.

The main question that arises at this point is what should be the general framework of competence sharing between evolving species. In general, as the task here is the representation transformation, each individual should be made responsible for a part of that task. Therefore, it should be equipped in a kind of input and output. Then, a particular scheme of cooperation may be conveniently represented in a form of directed graph showing the interconnections between particular individuals. Although such a scheme could be arbitrary, the following two approaches seem to be most canonical:

- *parallel* – transformed representation consists of a set of features, with each species responsible for one feature;
- *sequential* – representation transformation consists of a sequence of chained steps, with each species responsible for one step.

This study is exclusively devoted to the former of mentioned approaches. In particular, each CCA species is responsible for developing one feature for the final representation, and each individual representing particular species implements single feature. The selection of representatives follows the optimistic CCA-1 approach: each individual (feature) is evaluated in the context of best individuals representing remaining subpopulations (species) with respect to the previous evaluation process. This method has been selected mostly due to the positive results reported in (Wiegand et al., 2001).

5 EXPERIMENTAL EVALUATION

The described methodology has been verified in an extensive computational experiment. Two primary objectives of the experiment were: (i) to explore the usefulness of genetic programming-based construction of features, and (ii) to compare the cooperative coevolution (GP-CCA) with the standard approach (GP) on a real-world, difficult data set.

5.1 THE DATA

The experimental data was the GLASS benchmark from the Irvine repository of ML databases (Blake & Merz, 1998). Its training set L and test set T contain 142 and 72 examples respectively. Each example describes, by means of 9 numeric attributes, selected physiochemical properties of a glass sample. The task of the machine learner is to identify the glass type (float window, non-float window, container, tableware, headlamp) for the purpose of criminological investigation. The decision classes are highly imbalanced: the majority class occupies 35.6%[1] of the database, whereas the least representative one – only 4.2%. There are no missing values of attributes. All conditional attributes were normalized before starting the experiment to make their values reasonably comparable in GP expressions.

5.2 EXPERIMENT DESIGN AND PARAMETER SETTINGS

A single experiment consisted of the following steps:

1. Performing evolutionary construction of features using training data *L*, original attributes F_0, and set of GP terminals and nonterminals described further in this section.
2. Inducing the classifier from the training set *L*, using the features constructed in the evolutionary search (*F*).
3. Testing the induced classifier on the external test set *T*.

To make the results statistically significant, this scheme was repeated 20 times for different initial populations and for each considered number of features *n* (*n*=[2,9]). The settings of evolutionary run were almost the same as the defaults provided in the ECJ package (Luke, 2001). The function set used was rather limited and included +, -, *, % (protected division), LOG (logarithmic function), LT, GT, and EQ (arithmetic comparison operators). The terminal set encompassed the ephemeral random constant and the original attributes from F_0. Weak typing has been used, so no constraints were imposed on the mutation and crossover operators, except for those concerning individual's maximal depth (set to 5 for both crossover and mutation). Standard tournament selection with tournament size 7 and common GP recombination operations were applied. Each run was terminated after 50 generations.

For a given number of features *n* and population size *s*=100, a pair of corresponding GP and GP-CCA experiments consisted of:

- GP run involving one population of size *s*, with each individual encoding *n* features, and
- GP-CCA run involving *n* subpopulations, each of size *s*, with each individual encoding *one* feature.

Such a design of the experiment provides that the number of features potentially considered during the evolutionary search is the same ($n \times s$) for GP and GP-CCA, so the approaches have equal chances in searching the space of features[2].

In GP-CCA, each individual is evaluated in the context of the best individuals (with respect to the previous evaluation) representing the remaining species, i.e. according to the CCA-1 scheme (see, for instance, (Wiegand et al., 2001)). After each generation, we estimate the contribution of each subpopulation to the entire solution basing on the fitness differential of the best individual. Subpopulations that do not contribute to the solution (or even deteriorate its evaluation) are re-initialized.

The evaluation of the entire solution (i.e. vector of *n* features implemented by one GP individual or by *n* cooperating GP-CCA individuals) relies on the so-called *wrapper* methodology (Kohavi & John, 1997). The values of features defined by the evaluated solution are computed for all training examples from *L*, what produces a new, derived dataset. Then, a multiple train-and-test experiment (here: 3-fold cross validation) is carried out using an inductive learning algorithm and the resulting average accuracy of classification becomes the evaluation. For this purpose, the C4.5 decision tree inducer (Quinlan 1992), as implemented in WEKA (Witten & Frank 1999) with default settings (decision tree pruning on, pruning confidence level 0.25) has been used. The choice of this particular inducer was motivated by its relatively low computational complexity (of the training algorithm as well as of the querying process), readability of induced hypotheses, and popularity in ML community.

According to (Kohavi & John, 1997) and (Dash & Liu, 1997), wrapper has the advantage of taking into account the inductive bias of the classifier used in the evaluation function. It also maintains the discovery of synergy between particular attributes, as opposed to some local approaches, where particular features are evaluated on individual basis (see (Dash & Liu, 1997) for comparison of different feature selection strategies).

5.3 PRESENTATION OF RESULTS

Figures 1 and 2 present the performance of GP and GP-CGA approaches respectively on the training and test set. For the training set (Fig. 1) that is the fitness of the best individual of the run, whereas for the test set (Fig. 2) it is the accuracy of classification obtained on the test set *T* by the C4.5 algorithm trained using best evolved representation (see explanation on the beginning of Section 5). The charts are drawn as a function of the number *n* of constructed features. Both charts show means over 20 genetic runs with bars depicting 0.95 confidence intervals.

[1] This is the accuracy of classification of the so-called *default classifier*, which is a worst-case reference value for the experimental results.

[2] It should be noted, however, that for GP-CCA the number of calls of the global fitness function (wrapper) is *n* times greater than for GP. Therefore, unless some sophisticated programming tricks are used, the computing times are significantly longer for GP-CCA.

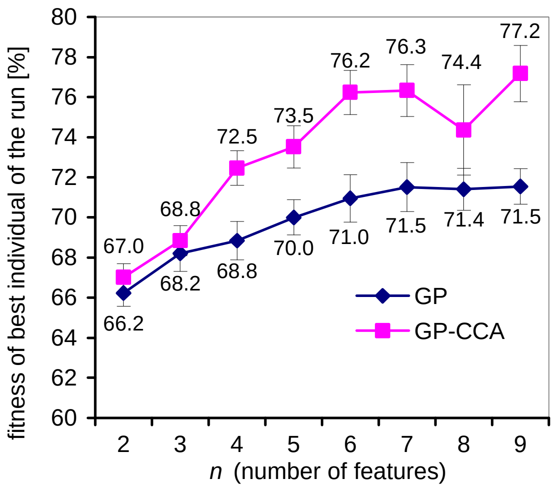

Figure 1: GP-CCA versus GP on the training set.

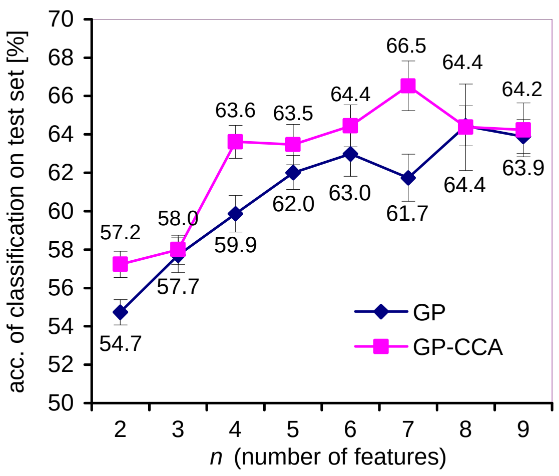

Figure 2: GP-CCA versus GP on the test set.

6 CONCLUSIONS

Results presented in Figure 1 clearly prove that coevolutionary search of representation space (GP-CCA) constructs better features with respect to the set of fitness cases (training set *L*) that 'vanilla' GP. With except for n=2 and n=3, which are apparently too small numbers of features to build a reasonable representation on, GP-CCA outperforms GP with respect to t-Student test at the confidence level 0.02. When comparing both methods in a broader view, the Wilcoxon's rank test on the averages produces p=0.012.

Figure 2 allows us to state that the superiority of GP-CCA concerns also the performance on the test set *T*, which, let us stress again, was 'invisible' for the evolutionary run. This time the 0.02 t-test significance holds only for n=4 and n=7, but the general tendency remains in favor of the coevolutionary approach (Wilcoxon's p=0.017).

The more general conclusion of this experiment is that the representations learned by means of GP-based feature construction often outperform the original representation F_0 as far as the predictive accuracy is concerned (C4.5 yields 62.5% accuracy of classification on the test set T when using F_0). In particular, this statement is true for $n \geq 4$ in case of GP-CCA and for n=6, 8, and 9 in case of GP. Note also that most of the increases have been obtained by means of a more compact representation (in all runs, n does not exceed the size of original representation $|F_0|$=9). Larger increases are likely to be achieved after more precise parameter tuning.

Let us note that the constructed features give extra insight into the knowledge hidden in the dataset. Carefully selected feature definitions (S-expressions) could be used for explanatory purposes after some rewriting, verification and expert's assessment. For some examples of such features see (Krawiec, 2002).

As far as CCA-related issues are concerned, more detailed analysis is required to investigate and explain cooperation patterns and dynamics taking place in CCA-GP feature construction. In particular, the cooperation scheme seems to be here more complex than in other studies related to the topic. We base this hypothesis on the fact, that the collaboration of separately coevolved features takes place by the mediation of the inductive learning algorithm implemented in the fitness function. The features developed by particular subpopulations are selectively utilized by the decision tree inducer embedded in the fitness function (C4.5), rather than being just put together to build up the solution. The inducer uses particular features at different stages of the top-down decision tree construction, so the effective contribution of particular feature to the final fitness value is partially determined by its location in that tree. Therefore, it is mostly the C4.5 *inductive bias* that guides the search for promising synergies between representation components.

The results obtained show also that overfitting is still a challenge for feature-constructing learners. For both GP and GP-CCA approaches, the external test set accuracy is much worse than the estimate produced by the wrapper-based fitness function (by ca. 10%). This is due to the infamous 'curse of dimensionality': the presence of new features increases dimensionality of the hypothesis space; instead of one representation space, we consider many of them. Consequently, the inducer is very prone to overfitting as it has much more 'degrees of freedom'. Potential benefits that overfitting prevention may draw from cooperative coevolution methodology might be one of other interesting topics for future research.

Acknowledgments

We would like to thank the authors of software packages: ECJ (Evolutionary Computation in Java) (Luke, 2001) and WEKA (Witten & Frank, 1999) for making their software publicly available. This work has been supported by the State Committee for Scientific Research, from KBN research grant no. 8T11F 006 19, and by the Foundation for Polish Science, from subsidy no. 11/2001.

References

H.N. Bensusan and I. Kuscu, "Constructive induction using genetic programming," in T. Fogarty and G. Venturini, Proc. Int. Conf. Machine Learning, Evolutionary computing and Machine Learning Workshop, 1996.

C.L Blake and C.J. Merz, "UCI Repository of machine learning databases" [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California: Irvine, CA, 1998.

M. Dash and H. Liu, "Feature Selection for Classification," Intelligent Data Analysis vol. 1(3), pp. 131-156, 1997.

K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Doctoral dissertation, University of Michigan: Ann Arbor, 1975.

K.A. De Jong, W.M. Spears, and D.F. Gordon, "Using genetic algorithms for concept learning," Machine Learning vol. 13, pp.161-188, 1993.

D. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley: Reading, 1989.

J.H. Holland, Adaptation in natural and artificial systems, University of Michigan Press: Ann Arbor, 1975.

J.K. Kishore, L.M. Patnaik, V. Mani, and V.K. Agrawal, "Application of Genetic Programming for Multicategory Pattern Classification," IEEE Trans. Evolutionary Comp. vol. 4(3), pp. 242-258, 2000.

M. Komosinski and K. Krawiec, "Evolutionary weighting of image features for diagnosing of CNS tumors," Artif. Intell. in Medicine vol. 19 (1), pp. 25-38, 2000.

R. Kohavi and G.H. John, "Wrappers for feature subset selection," Artificial Intelligence Journal, vol. 1-2, pp. 273-324, 1997.

J.R. Koza, Genetic programming – 2, MIT Press: Cambridge, 1994.

K. Krawiec, "Pairwise Comparison of Hypotheses in Evolutionary Learning," in Proc. Int. Conf. Machine Learning, C.E. Brodley and A. Pohoreckyj Danyluk (eds.), Morgan Kaufmann: San Francisco, 2001, pp. 266-273.

K. Krawiec, "Genetic Programming with Local Improvement for Visual Learning from Examples," in Computer Analysis of Images and Patterns (LNCS 2124), W. Skarbek, (ed.), Springer: Berlin 2001, pp. 209-216.

K. Krawiec, "Genetic Programming-based Construction of Features for Machine Learning and Knowledge Discovery Tasks". *Genetic Programming and Evolvable Machines*, 2002 (in press).

P. Langley, Elements of machine learning, Morgan Kaufmann: San Francisco, 1996.

T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms, Machine Learning, 2001 (to appear).

S. Luke, ECJ 7: An EC and GP system in Java. http://www.cs.umd.edu/projects/plus/ec/ecj/, 2001.

C.J. Matheus, "A constructive induction framework," in Proceedings of the Sixth International Workshop on Machine Learning, Ithaca: New York, 1989, pp. 474-475.

R.S. Michalski, "A theory and methodology of inductive learning," *Artificial Intelligence* 20, pp. 111-161, 1983.

T.M. Mitchell, An introduction to genetic algorithms, MIT Press: Cambridge, MA, 1996.

T.M. Mitchell, Machine learning, McGraw-Hill: New York, 1997.

M.A. Potter, K.A. De Jong, Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents, *Evolutionary Computation*, 8(1), pp. 1-29, 2000.

J.R. Quinlan, C4.5: Programs for machine learning, Morgan Kaufmann: San Mateo, 1992.

M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain, "Dimensionality Reduction Using Genetic Algorithm," IEEE Trans. on Evolutionary Computation, vol. 4, no. 2, pp. 164-171, 2000.

H. Vafaie and I.F. Imam, "Feature selection methods: genetic algorithms vs. greedy-like search," in Proceedings of International Conference on Fuzzy and Intelligent Control Systems, 1994.

R.P. Wiegand, W.C. Liles, K.A. De Jong. An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, Morgan Kaufmann: San Francisco, 2001, pp. 1235-1242.

I.H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann: San Francisco, 1999.

J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in Feature extraction, construction, and subset selection: A data mining perspective, H. Motoda and H. Liu (eds.), Kluwer Academic: New York, 1998.