Optimized Nearest-Neighbor Classifiers Using Generated Instances

Matthias Fuchs
Center for Learning Systems and Applications
Computer Science Dept., University of Kaiserslautern
P.O. 3049, D-67653 Kaiserslautern, Germany
e-mail: fuchs@informatik.uni-kl.de

Andreas Abecker
German Research Center for Artificial Intelligence
DFKI GmbH, Kaiserslautern
P.O. 2080, D-67608 Kaiserslautern, Germany
e-mail: aabecker@dfki.uni-kl.de

Abstract

We present a novel approach to classification, based on a tight coupling of instance-based learning and a genetic algorithm. In contrast to the usual instance-based learning setting, we do not rely on (parts of) the given training set as the basis of a nearest-neighbor classifier, but we try to employ artificially generated instances as concept prototypes. The extremely hard problem of finding an appropriate set of concept prototypes is tackled by a genetic search procedure with the classification accuracy on the given training set as evaluation criterion for the genetic fitness measure.

Experiments with artificial datasets show that—due to the ability to find concise and accurate concept descriptions that contain few, but typical instances—this classification approach is considerably robust against noise, untypical training instances and irrelevant attributes. These favorable (theoretical) properties are corroborated using a number of hard real-world classification problems.

Keywords: Nearest-Neighbor Classification, Genetic Algorithm, Instance-based Learning

1 Introduction

Memory-based approaches to classification have shown their usefulness in many applications. In domains with weak or intractable background theory [BPH90], when searching for goal concepts consisting of many small disjuncts or when being faced with poor predictive power of attributes or with imprecise and polymorphous concepts [BPH90, AKA91, MST94], methods like nearest-neighbor algorithms (k-NN) or instance-based learning (`IBL`) often produce good classification results. Regarding a number of real-world applications, k-NN was reported by [MST94] to outperform all other classification approaches examined in the `StatLog` project. These results are achieved by conceptually very easily understandable and implementable algorithms:
In the learning phase, they simply record the given training examples (instances) together with their classifications. These training instances are attribute-value representations describing points in the attribute space (or, instance space) that are example instances of concepts. Concepts therefore can be understood as sets of points in this attribute space. Subsequently presented unseen examples are classified by searching the k most similar stored instances and determining the most frequent class in this set of nearest neighbors. Thus, a memory-based classifier consists of a set of stored training instances (the *concept description*) plus the k-NN classification rule ([CH67]) for some specific k. However, when storing *all* presented examples, large storage requirements and high classification costs become a serious problem.
Approaches like `IB3` [AKA91] or `TIBL` [Zha92] tackle this problem by storing only a *subset* of the presented instances that is expected to be sufficient for a good characterisation of the goal concept. This is achieved in `IB3` by considering the instances' classification performance on subsequently presented training examples in order to discriminate good classifying instances from noisy ones. While `IB3` hence evaluates the "usefulness" of instances indirectly via their classification behavior, `TIBL` employs an *explicit* measurement of "typicality" of instances based on inter-concept and intra-concept similarity. Despite their different motivation and technical approach both algorithms obtain smaller concept descriptions and better classification accuracy (especially in noisy domains) by concentrating on central points in clusters of instances with the same class.
However, both approaches still preserve the idea of using only *presented* instances for a concept description. But, obviously in real-world domains these will usually contain a more or less random selection of somehow biased and noisy examples that highlight only certain aspects of the goal concept. So, we think that conventional instance-based algorithms impose unnecessary limitations on themselves when considering these instances as a starting point for constructing a concept description.
If we want to liberate us from the *direct* use of the training set, we have to face an extremely hard search problem: The search space contains all possible concept descriptions, i.e. all sets of arbitrary points in the attribute space and there is no heuristic knowledge for guiding the search and no known structure of this search space. Furthermore, since the training set is the *only* information available (provided that there is no background knowledge used), we

must nonetheless finally rely on this data. Thus we propose to use it to evaluate a (partial) solution candidate by measuring its classification accuracy on this example set. So we have an indirect use as in `IB3`, but we are free to construct arbitrary concept descriptions not restricted by the example instances.
Recently, a number of publications demonstrated the power of *Genetic Algorithms* (GA) under such conditions: huge, unstructured search spaces and the only available knowledge given as an a posteriori evaluation of a solution candidate's quality. This led us to the idea of our algorithm GIGA: Generate Instances for concept descriptions using a Genetic Algorithm. GIGA employs genetic search to find optimal concept descriptions (to be used by a k-NN classifier). Since these concept descriptions are artificially generated and are not restricted to the use of presented instances, they can contain arbitrary instances and have thus the chance of finding *"ideal"* concept descriptions with better performance w.r.t. both classification accuracy and storage requirements. Moreover, because our candidate evaluation always regards all information available, it is not as exposed to the danger of being trapped in local optima as incremental instance-based approaches are.
This paper is organized as follows: In section 2, we give a more detailed description of the GIGA algorithm which is followed by a discussion of interesting properties of our approach (section 3). Some artificial datasets demonstrate GIGA's ability to find concise concept descriptions (containing few, but typical instances) and to handle untypical training data, and its robustness against noisy examples and irrelevant attributes. In section 4, we report first promising experimental results on well-known real-world datasets taken from publicly accessible repositories. In the last section 5, we discuss some related work, summarize the approach, its strengths and limitations, and conclude with possible future improvements.

2 GIGA: The Basic Algorithm

Section 1 outlined the motivation for generating instances in order to obtain a concept description instead of using (a subset of) given training instances. Generating a concept description essentially amounts to a search problem. The resulting search space is the set of all concept descriptions, i.e., the set of all sets of instances. In practice, it stands to reason to limit the number of instances of a concept description to some fixed $n_I \in \mathbb{N}$. Besides a (significant) reduction of the search space, such a limitation offers further benefits we shall address in section 3. Despite the reduction, the search space in general remains enormous and hence intractable with simple search methods such as hill-climbing or random search. Therefore, the use of a *Genetic Algorithm* (GA, [Hol92], [Jon88]) appears to be appropriate, because the GA has the potential to cope with intricate search spaces in the absence of any knowledge about their structure. Furthermore, a GA is less prone to getting trapped in a local optimum. Both properties are highly valuable for our purpose (cp. [JSG93]). In the sequel, we describe the basics of the GA in the light of our application, implemented by an experimental program called 'GIGA'.
Unlike other search methods, the GA maintains a set of (sub-optimal) solutions, i.e., several

points in the search space. In this context, a solution is preferably called an *individual*, and the whole set is referred to as a *population* or *generation*. Usually, the size of the population is fixed. In order to explore the search space, the GA applies so-called *genetic operators* to (a subset of the) individuals of its current population. This way, new individuals can be created and hence new points in the search space can be reached. In order to keep the population size fixed, it must be determined which individuals are to be eliminated in order to make room for the new ones. For this purpose a so-called *fitness measure* is employed which rates the fitness (i.e., the ability to solve the problem at hand) of each individual of the current population. The genetic operators are applied to the most fit individuals producing "offspring" which then replaces the least fit individuals ("*survival of the fittest*").

So, the GA basically proceeds as follows: Starting with a randomly generated initial population, the GA repeats the cycle comprising the rating of all individuals using the fitness measure, applying the genetic operators to (a selection) of the best individuals, and replacing the worst individuals with offspring of the best, until some termination condition is satisfied (e.g., an individual with a satisfactory fitness level has been created).

In our case an **individual** $\mathcal{I}$ corresponds to a concept description, i.e., a (finite) set of instances. The **fitness** of an individual is measured in terms of its classification accuracy regarding a given set T of training instances. To this end we apply the k-NN rule with $k = 1$. That is, for each instance $I \in T$ the nearest neighbor $I' \in \mathcal{I}$ is computed according to the Euclidean distance measure. A correct classification is registered if the classes associated with I and I' agree. The classification accuracy of $\mathcal{I}$ is the percentage of correctly classified instances of T. If two individuals have the same classification accuracy, we prefer the one which uses fewer instances as concept description.

The **genetic operators** are subdivided into *reproducing* and *mutating* operators. Reproducing operators produce offspring, while mutating operators alter this offspring. GIGA employs two reproduction operators, namely *crossover* and *cloning*. The **crossover operator** randomly selects two distinct parents from the pool of $r\%$ best (surviving) individuals.[1] A subset of the instances of each parent individual is chosen at random, and the union of these two subsets yields the "child" individual. Thus, this operator complies with the basic idea of crossover, namely providing the ability to combine good partial solutions.

The **cloning operator** simply copies a randomly selected parent individual. Cloning only makes sense in connection with mutation operators to be described shortly. It is reasonable in particular if the top ranking individuals are very close to an optimum, and slight variations (mutations) of them have a higher chance to actually yield an optimal individual than (mutations preceded by) crossover. Whenever offspring is to be generated, either crossover or cloning are chosen at random according to a given probability distribution.

The **mutation operators** modify individuals stemming from crossover or cloning. An individual $\mathcal{I}$ is subject to mutation with probability P_{mut}. If an individual *is* selected for mutation, the following mutation operators are applied in the given order. (1) *Deletion*:

[1] Offspring generated by the reproduction operators hence replaces the $100 - r\%$ least fit individuals.

One instance of $\mathcal{I}$ is chosen at random and discarded. This *deletion operator* is applied with probability P_{del}. It is useful to get rid of instances which do not improve classification accuracy and hence unnecessarily hinder finding concise concept descriptions. (2) *Mutate Instances*: Each instance I of $\mathcal{I}$ is subject to *random mutation* with probability P_{rnd}. That is, if $I \in \mathcal{I}$ *is* chosen for random mutation, then, with probability P_{comp}, each component a of I (i.e. a is an attribute-value pair) including the class associated with I may be replaced by a random value taken from the respective range. This kind of mutation realizes "pure" random influences and is hence helpful in introducing the necessary diversity at early stages of the genetic search. (3) *Addition*: A random number of $n \geq 1$ instances generated at random are added to $\mathcal{I}$ (without exceeding n_I of course). The *addition operator* is applied with probability P_{add}.

In summary, GIGA attempts to obtain a concept description which is both concise and accurate w.r.t. the given training set. Accuracy and conciseness are respectively the first and second objective GIGA pursues by virtue of the employed fitness measure. Note that the fitness measure is the only (indirect) connection between a concept description found by GIGA and the training set. The merely implict dependence on the training set gives rise to some interesting properties of our approach which the following section will explain and demonstrate.

3 Properties of Our Approach

Section 1 already mentioned properties of our approach which we shall now examine more closely. Please note that the experiments of this section in connection with artificial domains mainly are to illustrate the point that is being made. They are not supposed to exhibit experimental evidence for the capabilities of our approach or its (general) superiority to other approaches. Some experimental results regarding real-world database applications are given in section 4.

3.1 Generating Typical Instances

Classification using the nearest neighbor rule entails—in its standard form—storing all training instances as concept description. Apart from possibly considerable storage requirements, storing all training instances also slows down the classification of an unknown test instance, since it must be compared with each and every training instance in order to determine its k nearest neighbors. Therefore, one attempts to store only a "sufficient" subset of all training instances which allows for classifying these training instances with an "acceptable" accuracy rate.

Most efforts aiming at a reduction of storage requirements essentially center on discarding all those training instances that are correctly classified even when removed from the training set (e.g. [Gat72]). Refinements of this approach are based on selecting instances from the training set that satisfy certain criteria such as being *typical* or *near-boundary* instances (cp.

Table 1: Experimental results concerning the 5-of-10 concept

	GIGA				TIBL	
Training Set Size	avg. acc.	avg. #inst.	avg. #cycles	avg. run time	avg. acc.	avg. #inst.
100	100%	2	83.5	13.68 sec	85.6%	19.6
200	100%	2	56.4	21.91 sec	94.0%	15.1
300	100%	2	78.6	41.23 sec	98.8%	15.2
400	100%	2	59.8	40.14 sec	99.5%	10.8

[Zha92]). Roughly speaking, typical instances are similar to instances of the same class, but different from instances of other classes. (Difference resp. similarity are commonly expressed with the help of the distance measure at hand.) Near-boundary instances are the opposite of typical instances. Using a geometric interpretation, near-boundary instances are located at the borders of a cluster of instances belonging to a certain concept, whereas typical instances are located at the center[2] of such a cluster.

Reviewing our approach as implemented by GIGA, it becomes clear that GIGA is searching for exactly these typical instances without having an explicit notion of typicality. Due to the fitness measure, concept descriptions are sought which (a) contain as few instances as possible while (b) representing the concept as accurately as possible. The centers of clusters of instances belonging to the same class are perfect candidates for such concept descriptions, because they allow for subdividing the instance space with the desired parsimony and accuracy in connection with the 1-NN rule. The striking advantage of generating instances is that we do not depend on the occurrence of typical instances in the training set. Even in the "worst case" when the training set contains only near-boundary instances, GIGA can still search for typical instances where other approaches solely relying on the training data face serious problems.

We shall illustrate this claim with an example taken from the n-of-m concept domain (see also [Zha92]). In this domain there are m attributes with binary values from $\{0, 1\}$ and two classes $C0$ and $C1$. If n or more attribute values of an instance are 1, then this instance belongs to $C1$. Otherwise it belongs to $C0$.

In [Zha92] the special case '5-of-10' was utilized to demonstrate the significant improvements the approach which selects typical instances from the training set (TIBL) can achieve compared to other instance-based approaches, w.r.t. both classification accuracy and the reduction of storage requirements. The results produced by this approach are shown in the last two columns of table 1. For this special case, the most typical instances I_{C0} and I_{C1} of the classes $C0$ and $C1$ are the instances with ten 0s and ten 1s as attribute values,

[2] The center of a cluster of instances is viewed as that point in the instance space which minimizes the distance between itself and all points corresponding to the (other) instances of this cluster. This view essentially conforms to the definition of typicality given in [Zha92].

respectively. As a matter of fact, a concept description $\mathcal{C} = \{I_{C1}, I_{C0}\}$ consisting of these two most typical instances (in that order)[3] together with the 1-NN rule allows for correctly classifying all 2^{10} instances. $\mathcal{C}$ is optimal in the sense that there is certainly no more accurate and more concise concept description.
The experimental environment set up in [Zha92] was replicated for our experiments with GIGA.[4] Four different sizes of training sets shown in the first column of table 1 were used. Training sets were generated at random. Each row displays the results obtained in connection with the respective training set size averaged over ten trials. All 1024 instances of the 5-of-10 concept were used as test data. For both GIGA (columns two through five) and TIBL the columns labeled 'avg. acc.' and 'avg. #inst.' display the average accuracy (w.r.t. the test data) and the average number of instances stored in the resulting concept description, respectively.
Columns two and three of table 1 reveal that GIGA was *always* able to find the optimal concept description $\mathcal{C}$ regardless of (the size of) the training set. Columns four and five list the average number of cycles and the average run time (CPU time)—obtained on a SPARC-station 10—GIGA required to accomplish this.[5] As for TIBL, the results in columns six and seven clearly exhibit the dependence on the size of the training set: Performance improves (higher accuracy, less stored instances) as the size of the training set increases, since larger training sets have a higher probability to contain I_{C0} or I_{C1}.

3.2 Near-Boundary Instances and Irrelevant Attributes

IBL approaches employing similarity measures based on spatial distance have serious difficulties when exposed to concepts involving irrelevant attributes. Irrelevant attributes do not convey any information concerning class membership. But in particular in connection with the k-NN rule, the spatial proximity of instances w.r.t. irrelevant attributes can have a distorting influence. As a consequence, classification accuracy can degrade significantly.
A similar problem arises when a training set mainly consists of near-boundary instances, i.e., instances which are (very) close to concept boundaries. Near-boundary instances located at opposite sides of the boundary of two distinct concepts can nevertheless be very close to each other. As a matter of fact, they might be closer to each other than to other instances of the respective concept. Consequently, they may be classified incorrectly.
Near-boundary instances and irrelevant attributes are two major causes for poor performance of common IBL approaches based on the k-NN rule. Extensions of the nearest

[3] It is assumed that $\mathcal{C}$ associates the instances with five 1s with the class $C1$, because I_{C1} occurs before I_{C0} in $\mathcal{C}$. This mechanism for resolving ambiguities concerning the distance measure is applied by GIGA's fitness measure. Therefore, $\mathcal{C}$ would receive a "perfect" fitness rating.

[4] Parameter setting for GIGA: population size: 100; survival rate: $r = 30\%$; maximal number of instances of an individual: $n_I = 20$; 75% of the offspring was created by cloning, the remaining 25% via crossover; all offspring was subject to mutation, i.e., $P_{mut} = 100\%$; Furthermore, $P_{del} = 80\%$, $P_{rnd} = 25\%$, $P_{comp} = 25\%$, $P_{add} = 0\%$.

[5] Note that the computations performed by the fitness measure need the more time the larger the training set is. The number of cycles, however, is mainly influenced by random effects.

Table 2: Near-boundary instances and irrelevant attributes

	1-NN	GIGA				
Training Set Size	avg. acc. test	avg. acc. test	avg. acc. training	avg. #inst.	avg. #cycles	avg. run time
40	58.74%	97.12%	99.75%	2.5	144	7.37 sec
60	60.75%	98.93%	100%	2.6	209.5	15.05 sec
80	63.53%	100%	100%	2	80.6	7.76 sec
100	68.02%	100%	100%	2	93.5	10.27 sec

neighbor algorithm in general cannot alleviate this drawback satisfactorily. The main reason is the fixation on (a subset of) the training set which might not provide the instances necessary for an appropriate concept description. When generating instances, however, we have the chance to find apt concept descriptions involving instances that do not exist in the training set. The following simple example illustrates these aspects.

There are two classes C_1 and C_2, and three attributes x, y and z whose values range from 1 to 20. An instance belongs to class C_1 if its attribute value for attribute x is in $\{1, \ldots, 10\}$. Otherwise it belongs to C_2. Training and test instances are randomly generated along the boundary—a plane—separating C_1 and C_2. This means that the attribute value for x is 10 w.r.t. instances of C_1 and 11 w.r.t. instances of C_2. Values for y and z are randomly chosen from $\{1, \ldots, 20\}$. The attributes y and z are irrelevant since class membership can be decided with the help of x alone. This example hence combines near-boundary instances and irrelevant attributes.

Any concept description consisting of two instances I_1 and I_2 satisfying the following conditions is optimal both w.r.t. classification accuracy and parsimony: The attribute values for x are $10 - a$ and $11 + a$ ($a \in \{0, \ldots, 9\}$), respectively. For both I_1 and I_2, the attribute values for y and z are b and c (b, $c \in \{1, \ldots, 20\}$). Even under these simple conditions it is very unlikely that two such I_1 and I_2 are in a training set of reasonable size, let alone the difficulty to extract exactly these two from the training set should they be there. GIGA, however, essentially searches for a concept description consisting of two such individuals since it represents a global optimum.

Our experiments regarding this two-class concept are summarized by table 2. We employed training sets of four different sizes (cf. first column of table 2). Half of the instances of such a training set belonged to C_1, and the other half to C_2. The results presented by each row of table 2 are the average of ten trials during each of which a random training set of the respective size was presented to GIGA and the k-NN algorithm, and tested with respect to ten also randomly generated test sets. Each test set had 50 instances of each class.

The second column of table 2 shows that the k-NN algorithm performs rather poorly. (We only list the results obtained with $k = 1$, but choosing $k > 1$ does not improve performance significantly, mostly even causing it to deteriorate.) Classification accuracy naturally in-

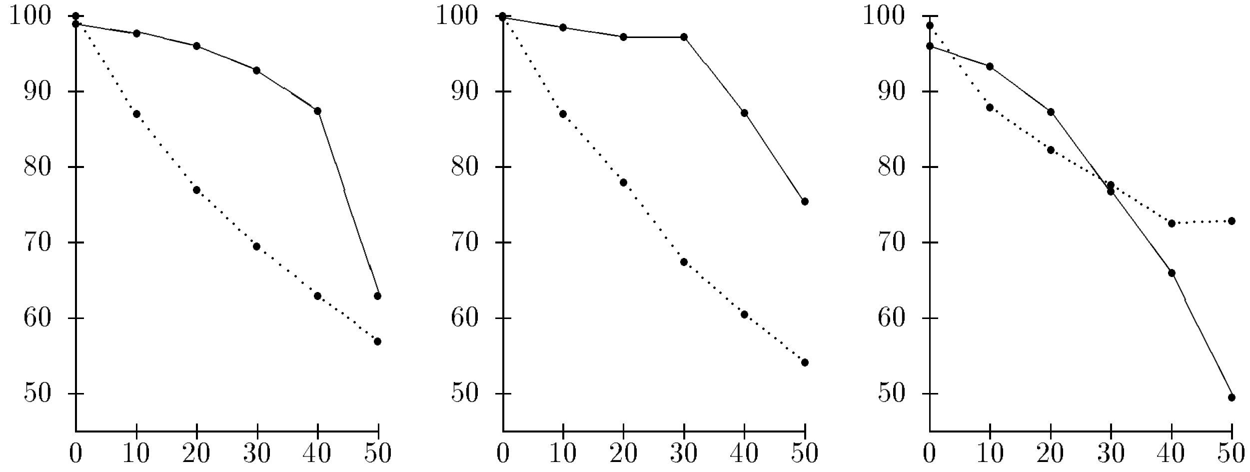
creases with the size of the training set. Under the present conditions, the performance of the k-NN algorithm can be improved by varying the Euclidean distance metric through the use of weighted attribute value differences. In case concept boundaries are not aligned with the attribute axes, rotations are necessary to fully profit from a weighted Euclidean distance measure. But determining rotation angles and weights also amounts to a search problem (cp. [KD91]). Note that our approach is independent of the orientation of concept boundaries w.r.t. attribute axes. To put it another way, the search problem remains the same whether concept regions are aligned with attribute axes or not. (The instances of an optimal concept description are rotated the same way as the whole concept.)
Columns three through seven list the results obtained with GIGA,[6] namely the average accuracy on the test and training sets, number of instances constituting the found concept description, number of cycles and CPU time. The classification accuracies attained by GIGA demonstrate that GIGA can cope with the present situation very well. In connection with training sets of the sizes 40 and 60, however, GIGA did not succeed in finding an optimal concept description in *one* of the ten trials.[7] This may be a coincidence, but it is also possible that smaller training sets entail search spaces with more deceptive local optima. (We want to emphasize at this point that GIGA is a "first generation" experimental program which certainly does not represent the state of the art of genetic search procedures. There is much room for improvements on that score. Therefore, limitations encountered when using GIGA should not be taken as evidence for a general limitation of our approach on account of intractable search spaces.)

3.3 Dealing with Noise

Noise tolerance is a property of classification systems which is particularly important in real-world applications which almost never are free of noise. Although our approach does not provide an *explicit* mechanism for dealing with noise (in contrast to, e.g., [AK89]), it nevertheless has a certain ability to tolerate noise. It derives this implicit ability from the fact that the concept descriptions which are searched for can be limited w.r.t. their size (i.e., n_I) resp. complexity.
If data is noisy, then there typically are subspaces of the instance space whose elements are all supposed to belong to a certain class, but they are pervaded to a degree by instances associated with alien classes which represent noise. In order to single out these noisy instances—which entails "cutting up" the instance space more strongly—concept descriptions have to be more complex. So, if the complexity (size) of concept descriptions is limited appropriately, then GIGA will search for *coarser* concept descriptions which kind of "ignore" noisy instances: Due to the restricted ability to cut up the instance space,

[6] Parameter setting: population size: 100; $r = 30\%$ survival rate; $n_I = 10$; offspring was produced via crossover only; $P_{mut} = 50\%$; $P_{del} = 0\%$; $P_{add} = 0\%$; $P_{rnd} = 25\%$; $P_{comp} = 50\%$.

[7] Besides finding an optimal concept description, having been trapped in a local optimum for more than 500 cycles is a further termination criterion. The average number of cycles and the run time naturally increase if termination was triggered by the latter criterion.



(a) GIGA: $n_I = 10$ (b) GIGA: $n_I = 5$ (c) 3-NN

Figure 1: Experiments with noisy data: The x axis displays the percentage of noise, and the y axis the (average) classification accuracy w.r.t. test data (straight lines) and training data (dotted lines).

GIGA can recognize larger coherent areas of the instance space which are associated with a certain class although they are "polluted" and pervaded by noise. Naturally, if the percentage of noisy instances exceeds a certain degree, then they cause distortions that cannot be compensated for anymore. The subsequent example is to illustrate this property.
There are three classes C_1, C_2, C_3, and two attributes x and y whose values range from 1 to 18. An instance belongs to class C_1 (C_2, C_3) if the value of its attribute x is in $\{1, \ldots, 6\}$ ($\{7, \ldots, 12\}$, $\{13, \ldots, 18\}$). The value of the (irrelevant) attribute y is chosen at random. Training and test data are generated at random, containing 150 instances (50 instances of each class). The test data is free of noise, whereas noise is introduced into the training data. To this end, for a certain percentage of randomly chosen instances of each class, the correct class label is replaced with an incorrect one (also chosen at random).
Figure 1 summarizes our experiments conducted in this domain. The accuracy results are averaged over ten trials (ten test sets per trial). We employed GIGA with two (maximal) sizes of concept descriptions, namely $n_I = 10$ and $n_I = 5$ (cp. figure 1a and 1b, respectively).[8] The remaining parameters were set as in the preceding subsection. Figure 1c displays the performance of the k-NN algorithm using $k = 3$.
Figure 1 shows that classification accuracy on the (noise-free) test data (straight lines) is significantly higher compared to 3-NN when using GIGA and forcing it to search for coarse concept descriptions. As a matter of fact, accuracy remains on a high level with as much as 30% noise in the training data. The accuracy of 3-NN drops more or less linearly right from the start. Note that the "coarser" setting involving $n_I = 5$ yields a slightly better

[8]Note that—similar to the example of subsection 3.2—there are concept descriptions consisting of three instances which achieve perfect classification accuracy (for noise-free data).

performance than the less coarse one ($n_I = 10$). The threshold for the amount of noise GIGA can satisfactorily cope with seems to lie somewhere between 30% and 40% for this experimental data. Exceeding this threshold causes the accuracy rate to drop sharply. Note that the accuracy on the training data (dotted lines) decreases in almost perfect correlation with the percentage of noise, which suggests that GIGA is able to in a way "sort out" (most of) the noisy instances as long as there is not too much noise. This effect can of course not be observed in connection with the 3-NN classifier.

4 Experimental Results

While in section 3 artificial datasets were used to demonstrate specific properties of our algorithm, we now examine some real-world datasets in order to evaluate the algorithm under hard conditions. For the sake of comparable and reproducable results we chose only publicly available datasets from the *Machine Learning Repository* at the University of California at Irvine [MA94]. We took datasets with well-documented results and used the `StatLog` Evaluation Assistant—a set of UNIX shell scripts and C routines for the test of classification algorithms and the production of standardized performance measures. We also used the same test method (leave-one-out, k-fold cross-validation, etc.) as reported in the literature for the respective examples. The following datasets were examined:

Diabetes (Pima Indians): diabetes diagnosis on the basis of physiological measurements and medical tests [SED$^+$88]; the best results are reported in [MST94]; 768 instances, 8 real-valued attributes, 2 classes.

Breast Cancer (Wisconsin): cancer diagnosis from cytological information; the best result is reported in [WM90, Zha92]; 699 instances, 10 integer-valued attributes, 2 classes.

Promoter sequences: prediction of biological promoter activity for a sequence of nucleotides; past usage by [HR87, TSN90]; 106 instances, 59 nominal attributes[9], 2 classes.

Heart Disease (Cleveland): diagnosis of coronary artery disease; best result described by [GLF89]; 303 instances, 14 integer-valued attributes, 2 classes.

Congressional voting: determine party affiliation from congressional voting; introduced by [Sch87]; 435 instances, 16 boolean attributes, 2 classes.

Results: Table 3 summarizes the classification accuracies obtained in this first experimental evaluation of GIGA. Using a standard parameter set-up for the GA[10] and a maximum number n_I of 25 instances allowed in an individual, GIGA was able to exceed the best reported results in the diabetes and the breast cancer domain. On the other hand, it took several adjustments of the n_I parameter[11] until we obtained comparable results in the promoter sequence, in the heart disease, and in the voting domain. Nevertheless, only for heart disease diagnosis, we were finally not able to produce an accuracy competitive with

[9] Nominal attribute values were replaced by integers according to their alphabetic order.

[10] Population size 400; 30% survival rate; $P_{mut} = 50\%$; $P_{del} = 10\%$; $P_{add} = 20\%$; $P_{rnd} = 10\%$; $P_{comp} = 100\%$; 60 cycles.

[11] This parameter specifies the minimal level of generalization GIGA is forced to achieve.

Table 3: Experimental results using real-world datasets

Dataset	test method	best reported algorithm	best reported acc.	GIGA acc.
Diabetes	12-fold cv	Logdisc	77.7%	80.5%
Breast Cancer	10-fold cv	Hyperplanes	95.9%	97.3%
Promoter	leave-one-out	KBANN	96.2%	98.1%
Heart Disease	10-fold cv	CLASSIT	78.9%	72.3%
Voting	train & test	STAGGER	90-95%	93.9%

the best published results.
These experiments show that GIGA is able to find optimized concept descriptions also in noisy domains with imprecise concepts and thus can contribute to better solutions for practically relevant classification tasks. Optimal parameter adjustment seems to be the crucial point in some domains. Mostly, the value of n_I turns out to be the main problem. But this is less serious, since this value can iteratively be incremented until an optimal *"complexity fit"* ([WK91]) of the concept description is reached. Because most papers concentrate on test-set accuracy, we report only this result which is also likely to be the strongest point of GIGA. Nevertheless, comparisons of run-time behavior, storage requirements[12] and robustness in noisy and dynamic situations provide interesting work for the near future. Another point for further investigations is the thorough examination of the instances generated by GIGA, especially in comparison with TIBL or prototype-learners.

5 Discussion

In this paper we presented GIGA, a novel approach to classification that employs a genetic search algorithm in order to approximate ideal concept descriptions to be used by a nearest-neighbor classifier.
Our algorithm follows the Pittsburgh approach to machine-learning oriented GAs [Smi83, Koz91, JSG93, Jan93] in that each individual of the population encodes a complete solution of the classification problem. But in contrast to other systems that learn explicit abstractions from examples (e.g. decision trees), our instance-based concept description permits a *scalable* output-representation language with arbitrary granularity.[13] Experiments with ML benchmark datasets (cf. section 4) show that the system is able to find an appropriately fine-grained level to describe hard real-world classification problems. This ability may be

[12]For these examples, depending on the size of the training set and the number of attributes, the run time needed to perform the 60 cycles ranged between 4 and 40 minutes on a SPARCstation 10. The size of the generated concept descriptions was comparable to the ones reported in [Zha92].

[13]Instance-based classifiers can piecewise-linearly approximate arbitrary concept boundaries.

seen as another form of dynamically adjusting system bias (cp. [JSG93]). But note that in our approach this adjustment is done implicitly, embedded in a rather elegant and easy to implement evolution cycle, whereas others need very sophisticated techniques.

Within the instance-based learning community, our approach consequently continues the idea of searching optimized concept descriptions as introduced by `IB3` [AKA91] and `TIBL` [Zha92]. However, the search for *"ideal"* instances seems to be unique in this community. The experiments with artificial datasets in section 3 illustrate that there exist situations in which our approach of *not* relying on the given instances to construe a concept description becomes a striking advantage. The use of *few, typical* instances essentially amounts to an implicit generalization from examples that makes it easier to tolerate a remarkable level of noise as well as many near-boundary instances or irrelevant attributes in the training set. [KD91] propose an approach that does not construct new instances but transforms the given instances by rotations and attribute scalings which are optimized by a GA in order to support the k-NN-classifier. This idea is somehow complementary to our approach. But it is not evaluated using real-world datasets, and it does finally not tackle the problem of bad input data (only the problem of bad input *representation*).

Mainly motivated by psychological studies [SM81, MS88], prototype learners (see e.g. [Maz91, DK95]) pursue a similar goal as we do in that they compute a concept prototype from attribute values and frequencies occurring in the training set. However, these systems are slightly more restricted to the given training instances as we are, since we can enforce an arbitrary level of generalization via the maximal number n_I of instances constituting a concept description. Furthermore, except for [DK95], prototype learners usually construct a *single* prototype instance per concept which is not always sufficient for an adequate concept representation. Of course, our flexibility is paid for by a high computational effort due to the genetic search. However, this search can still be improved through more sophisticated GA techniques and finally be performed in a highly-parallel way by multiprocessor machines. Simple incremental improvements of classifiers in order to take into account new information, and the advantages of an any-time algorithm are further benefits. Some limitations are naturally inherited from instance-based learning, for example poor explanation capability, the problem of finding appropriate distance metrics and the use of symbolic background knowledge. On the other hand, there are still promising future research directions, e.g. the co-evolution of the distance metric or the incorporation of individuals computed with Zhang's method or de la Maza's method into the first generation of the genetic algorithm.

To conclude we can say that GIGA represents an interesting complement for other learning approaches. In situations where instance-based learning seems appropriate, but is hindered by poor data quality, GIGA has the chance of considerably improving classification accuracy by spending large-scale computational power. This is the case in a number of practically relevant problems (see e.g. the breast cancer domain in section 4) where few reliable data and background knowledge is available, but classification improvements are highly valuable.

References

[AK89] D.W. Aha and D. Kibler. Noise-tolerant instance-based learning algorithms. In *Proc.* 11^{th} *IJCAI, Detroit, MI, USA*, pages 794–799, 1989.

[AKA91] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[BPH90] R. Bareiss, B. Porter, and R. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45(1–2), 1990.

[CH67] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[DK95] P. Datta and D. Kibler. Learning prototypical concept descriptions. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the* 12^{th} *International Conference ICML-95*. Morgan Kaufmann, San Francisco, CA, USA, 1995.

[Gat72] G.W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, pages 431–433, May 1972.

[GLF89] J.H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–61, 1989.

[Hol92] J.H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: Univ. of Michigan Press, 2^{nd} edition, 1992.

[HR87] C. Harley and R. Reynolds. Analysis of e. coli promoter sequences. *Nucleic Acids Research*, 15:2343–2361, 1987.

[Jan93] C.Z. Janikow. A knowledge intensive genetic algorithm for supervised learning. *Machine Learning*, 13:198–228, 1993.

[Jon88] K. De Jong. Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138, 1988.

[JSG93] K.A. De Jong, W.M. Spears, and D.F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188, 1993.

[KD91] J.D. Kelly and L. Davis. Hybridizing the genetic algorithm and the k nearest neighbors classification algorithm. In *Proc.* 4^{th} *ICGA, San Diego, CA, USA*, 1991.

[Koz91] J.R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1991.

[MA94] P.M. Murphy and D.W. Aha. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA, University of California, Department of Information and Computer Science, 1994.

[Maz91] M. De La Maza. A prototype based symbolic concept learning system. In *Proc. of the Eighth International Workshop on Machine Learning*, pages 41–45. Morgan Kaufmann, 1991.

[MS88] D.L. Medin and E.E. Smith. Concepts and concept formation. *Annual Review of Psychology*, (35):121–138, 1988.

[MST94] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[Sch87] J.C. Schlimmer. *Concept acquisition through representational adjustment.* PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA, 1987.

[SED+88] J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S Johannes. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care.* IEEE Computer Society Press, 1988.

[SM81] E.E. Smith and D.L. Medin. *Categories and Concepts.* Harvard University Press, 1981.

[Smi83] S. Smith. Flexible learning of problem solving heuristics through adaptive search. In *Proc.* 8^{th} *IJCAI, Karlsruhe, Germany*, 1983.

[TSN90] G. Towell, J. Shavlik, and M. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the* 8^{th} *National Conference on Artificial Intelligence (AAAI-90)*, 1990.

[WK91] Sh.M. Weiss and C.A. Kulikowski. *Computer Systems That Learn – Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann, 1991.

[WM90] W.H. Wolberg and O.L. Mangasarin. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences, USA*, volume 87, pages 9193–9196, December 1990.

[Zha92] J. Zhang. Selecting typical instances in instance-based learning. In *Proc.* 9^{th} *ICML, Aberdeen, Scotland*, pages 470–479, 1992.