IOGA:
AN INSTANCE-ORIENTED GENETIC ALGORITHM

Richard S. Forsyth.
email: forsyth_rich@yahoo.co.uk

[Cite as:

Forsyth, R.S. (1996). IOGA: an instance-oriented genetic algorithm. In: Voight, H.-M., Ebeling, W., Rechenberg, I. & Schwefel, H.-P. (eds.) *Parallel Problem Solving from Nature -- PPSN IV*. Berlin: Springer.
]

Abstract

Instance-based methods of classification are easy to implement, easy to explain and relatively robust. Furthermore, they have often been found in empirical studies to be competitive in accuracy with more sophisticated classification techniques (Aha et al., 1991; Weiss & Kulikowski, 1991; Fogarty, 1992; Michie et al., 1994). However, a twofold drawback of the simplest instance-based classification method (1-NNC) is that it requires the storage of all training instances and the use of all attributes or features on which those instances are measured -- thus failing to exhibit the *cognitive economy* which is the hallmark of successful learning (Wolff, 1991). Previous researchers have proposed ways of adapting the basic 1-NNC algorithm either to select only a subset of training cases (`prototypes') or to discard redundant and/or `noisy' attributes, but not to do both at once. The present paper describes a program (IOGA) that uses an evolutionary algorithm to select prototypical cases and relevant attributes simultaneously, and evaluates it empirically by application to a set of test problems from a variety of fields. These trials show that very considerable economization of storage can be achieved, coupled with a modest gain in accuracy.

Keywords: Dimensionality Reduction, Evolutionary Computing, Feature Selection, Nearest-Neighbour Classification.

1. Introduction

A very simple form of learning is *rote memory*, that is, the storage of previously encountered examples. Rote memory is the basis for what is

called *case-based reasoning*, in which previously solved problems are stored in a *case library* and novel problems are solved by first matching them to the most `similar' stored problem in that library and then applying (sometimes adapting) the associated solution to the current situation. Currently, case-based reasoning is a growth area in Artificial Intelligence (Kolodner, 1993; Althoff et al., 1995).

Essentially the same idea lies at the heart of the well-tried method of *nearest-neighbour_classification* (NNC) which was first proposed by Fix & Hodges (1951) and has been developed in several directions since then (Dasarathy, 1991). The basic form of this algorithm, single nearest-neighbour classification (1-NNC), works by holding the entire collection of training instances in memory: fresh cases are matched to each of these stored instances according to some similarity measure and assigned the class of the case they most closely resemble. It has been found on numerous past occasions (e.g. Forsyth, 1990; Aha et al., 1991; Fogarty, 1992; Michie et al., 1994; McKenzie & Forsyth, 1995) that, although conceptually very simple, this method frequently outperforms more sophisticated classification methods. Thus it was decided to use the 1-NNC algorithm as a basis for a novel evolutionary learning system.

2. Initial Benchmarking Trials

To begin with, a program was written in C, implementing essentially the 1-NNC algorithm as presented by James (1985). Some initial trials were conducted with this program on a collection of numeric data sets in order to: (1) establish a baseline performance level; (2) compare two different distance metrics (Euclidean versus City-block); (3) assess what level of decline in performance might typically be expected between training and test data when classifying with the nearest-neighbour technique.

No strong claims are made about this particular selection of data sets except that they have all been used by published authors in testing various statistical or machine-learning techniques. Table 1 gives information about the size of these data sets. For completeness, a name and brief description of each data set is given in Table 2. Further details may be found in the references cited.

Table 1 -- Details of Data Sets.

Name	No. of cases	No. of classes	No. of variables
BANKNOTE	206	2	7
CARDIAC	113	2	19
DIABETES	145	3	6
DIGIDAT	1001	10	12
DOGS	77	5	11
DRINKERS	345	3	5
FEDS	866	2	23
IRIS	150	3	4
QUIN	400	2	12
ZOOBASE	101	7	17

Table 2 -- List of Data Sets Used.

Data-Set Name & Source	Brief Description	Categories
BANKNOTE (Flury & Riedwyl, 1988)	measurements of images on genuine and counterfeit Swiss bank notes	0=forged; 1=genuine
CARDIAC (Afifi & Azen, 1979)	clinical data on patients admitted to a Los Angeles Hospital with heart failure	1=survived; 2=died
DIABETES (Andrews & Herzberg, 1985)	data on diabetic patients obtained by reaven & Miller (1979)	1=overt diabetes, 2=chemical diabetes, 3=healthy
DIGIDAT (Breiman et al., 1984)	quasi-random data simulating a faulty light-emitting diode display, plus four noise variables	numerals: 0 to 9

DOGS (Manly, 1994)	mandible measurements from jaws of five canine species, living and extinct	1=Thai dog; 2=golden jackal; 3=cuon; 4=Indian wolf; 5=prehistoric Thai dog
DRINKERS (Allaway et al., 1988)	blood enzyme measurements of healthy male volunteers obtained by BUPA, plus information about their habitual alcohol consumption	0=light drinker or abstainer; 1=moderate drinker; 2=heavy drinker
FEDS (Mosteller & Wallace, 1984)	frequencies of 22 function words used in sections of essays by Alexander Hamilton and by James Madison	1=Hamilton; 2=Madison
IRIS (James, 1985)	data on petal and sepal sizes of three species of Iris, collected by Anderson (1935) and made famous by Fisher (1936)	1=Iris Setosa; 2=Iris Versicolor; 3=Iris Virginica
QUIN (Quinlan, 1987)	an artificial data set designed to model a task in which only probabilistic classification is possible and which requires a disjunctive concept description	0,1 (assigned by a stochastic rule)
ZOOBASE (Forsyth, 1990)	numeric (mostly binary) attributes describing 101 different animal species grouped into seven zoological classes	1=mammal; 2=bird; 3=reptile; 4=fish; 5=amphibian; 6=insect; 7=other

Essential to the concept of finding the nearest neighbour of a given instance is an operational definition of distance between points in multi-dimensional space. Many different distance measures have been proposed. Two of the the most popular are Euclidean distance (root summed squared deviation) and the `city-block' metric (total absolute deviation). Both these were tried on the above data sets. In addition, performance levels on seen and unseen data sets were compared, to examine the susceptibility of 1-NNC to *overfitting*. This involved splitting each data set into two subsets of roughly equal size. Specifically, nine of

these 10 data sets were divided randomly into two subsets, each case having a 0.5 probability of being allocated to either. After this approximate halving, the file which in fact contained more cases was designated the `training' file and the other the test file. (The single exception was data-set FEDS, where whole essays were randomly assigned to test or training sets, then subdivided into segments of approximately equal length.)

The 1-NNC program was used to classify both training and test sets, producing the results given in Table 3. Note that a form of *jack-knifing* was employed (Mosteller & Tukey, 1977), here and subsequently; that is, when used on a single data set, the program finds the nearest neighbour by computing the distance of each case to all **other** instances, excluding the current case itself.

Table 3 -- Mean Percentage Correct Classifications with Split-Half Testing.

Distance Metric ↓	self-test (jack-knifed)	test on unseen data
Euclidean	72.15	71.54
City-block	73.66	72.57

A 2-way Analysis of Variance of deviations from the mean score on each problem (to eliminate the effect of problem difficulty, essentially a nuisance factor) was performed. There was no significant main effect of Distance Metric ($F_{1,37}$ = 1.06, p = 0.309); nor was testing on useen data significantly different from self-test mode ($F_{1,37}$ = 0.48, p = 0.493). These figures emphasize one of the most desirable properties of the 1-NNC technique, namely the fact that it normally gives what Breiman et al. (1984) call `honest' error estimates. That is: a self-test on a random sample from a population (provided that jack-knifing is used) will tend to give an error-rate estimate that is not systematically biased towards either under- or over-estimation of the error rate to be expected on another random sample of comparable size from the same population. It was thought important to demonstrate these characteristics of the classic 1-NNC algorithm before turning to developments intended to improve it.

3. Some Disadvantages of 1-NNC

As shown above, 1-NNC is easy to implement and relatively robust. Nevertheless, it does suffer from some drawbacks:

(1) it requires all training cases to be stored, thus simulating memorization rather than learning, as that term is usually understood;
(2) in consequence, the classification phase is rather slow;
(3) it uses all features of each feature vector in assessing similarity to memorized cases, thus failing to compensate for, or exploit, the redundancy among variables found in most real data sets;
(4) since the `knowledge base' is just the training data, it does not produce an intelligible **description** of what it has learned.

In an effort to alleviate the weaknesses listed above, a novel program was developed, based on the 1-NNC method but radically modified, as described in the next section.

4. An Instance-Oriented Genetic Algorithm

The need to store all training cases in nearest-neighbour classification has seemed wasteful of both storage space and computing time to previous researchers, and several ways of reducing this wastefulness have been devised. Many authors have proposed ways of selecting only a subset of the training cases (Hart, 1968; Swonger, 1972; Ullman, 1974; Ritter et al., 1974; Gabor, 1975; Tomek, 1976; Hand & Batchelor, 1978; Devijer & Kittler, 1982; Fukunaga & Mantock, 1984; Aha et al., 1991). Some also have proposed methods that involve the creation of `archetypes', such as centroids, along with or instead of actual training instances (e.g. Batchelor, 1974; Chang, 1974; Batchelor, 1978; Geva & Sitte, 1991): this moves the nearest-neighbour technique away somewhat from pure rote memorization towards true learning in that it usually entails the storage of prototypical instances that were never actually encountered during training; thus it becomes rather like the Learning Vector Quantization, or LVQ, technique of Kohonen (1988). In addition, some authors have sought to avoid using spurious &/or redundant variables by methods of feature weighting &/or selection (e.g. Siedlecki & Sklansky, 1989; Kelly & Davis, 1991; Smith et al., 1994; Pei et al., 1995).

However, to the best knowledge of the present author, no previous reseacher has yet proposed a method of economizing on both cases

stored and features used at the same time, the object of the program described in this section.

4.1 The IOGA Program

The selection of a suitable subset of variables and cases for nearest-neighbour classification can be seen as an optimization task. It could perhaps be performed in a sequential manner, as in a stepwise regression, but this approach is well known to be vulnerable to interaction effects among variables (McLachlan, 1992); and in the present case interactions among instances chosen and between variables and instances would also have to be considered. In theory, such an optimization could be performed by exhaustive search, but with a training set such as FEDS, containing 484 instances measured on 23 variables, that would entail looking at 2^{507} subsets -- clearly not a feasible option. Accordingly, since the *genetic algorithm* (GA), has been found to be a robust general-purpose optimization technique (Goldberg, 1989), this problem was tackled here using an evolutionary method.

IOGA (Instance-Oriented Genetic Algorithm) embodies principles common both to the *evolution strategy* of Rechenberg (1973) and the *genetic algorithm* of Holland (1975). These in turn are based on a biological model, namely the Darwinian idea of evolution by natural selection (Darwin & Wallace, 1858).

In any program of this kind there will be a *population* of structures representing potential solutions to the problem in hand which can be scored, or at least ranked, by some kind of *fitness function*. To emulate Darwinian "survival of the fittest", new candidate solutions are generated by a process analogous to *reproduction*. Ordinarily this involves a quasi-random selection, biased somehow in favour of higher-scoring members of the population, of two, or sometimes more, *parents* to which a *crossover* operation is applied (analogous to mating). A *mutation* operator is typically applied to the resultant offspring which is then inserted into the population, displacing a low-scoring individual. Within this general framework there are many variants, differing in details such as how crossover and mutation are implemented. In fact, the two most important attributes of any GA implementation are: (1) the representation scheme; (2) the fitness function used.

In IOGA the representation scheme is quite transparent: each item in the population is a string of R+V bits, where R is the number of records or instances and V is the number of variables in the training data. A 1

anywhere in the first R positions of this bitstring signifies that the corresponding case is to be included among the selected prototypes, a zero means that it is to be excluded. Similarly, 1 anywhere in the last V positions signifies that the corresponding variable in the feature vector is to be used in distance calculations, while a zero means that it is not. This representation is well suited to being chopped up and recombined by the GA operators.

The fitness of an individual bitstring is computed by running the jack-knifed 1-NNC procedure over the whole training set with only the selected instances used as prototypes and only the selected features employed in distance calculations. The number of correct classifications (K) is recorded during this evaluation. The fitness (F) of that gene-string is then given by

$$F = K - B/(R + V)$$

where B is the number of non-zero bits in the string and R+V, as before, is the total number of bits in the string. Essentially the subtraction of B/(R+V), the proportion of bits used, gives this fitness formula a bias towards brevity which acts as a tie-breaker: for bitstrings with equal error rate the one using less information is preferred. This may be seen as a crude operationalization of Occam's Razor. Note that K, the number of correct decisions, is summed over all cases in the training set, whether or not they are included by the bitstring in the prototype subset. Note also that, because jack-knifing is used, no case can be its own nearest neighbour.

The version of the GA used in IOGA is novel, though loosely based on a procedure called Iterative Genetic Search with Uniform Crossover (IGS-U) devised by Ackley (1987). An outline follows.

1. Create an initial population of random gene-strings, and compute their fitness scores.
2. Pick a parental gene-string at random from the population.
3. Pick a second parent by making P random probes in the population and retaining the gene-string with the highest fitness score (of the P strings sampled).
4. Make P random probes in the population and record the location of the gene-string with the lowest fitness score (out of P sampled).
5. Make a new offspring by applying the uniform crossover routine[1] to

[1] Uniform crossover makes an offspring by stepping through each string position in turn and at each position picking a binary digit from one or other parent with equal probability.

the two parental strings.

6. Randomly replace approximately 4% of the bits in the newly created string by random bits (0 or 1 with 0.5 probability). (This will make no difference half the time, by chance, so the effective mutation rate is in fact 2%.)
7. Replace the member of the population selected in step 4 by the newly created gene-string. Also, compute the new string's fitness and if it happens to have the best score seen so far, save a copy (outside the gene pool) for subsequent printout.
8. On a proportion of occasions (currently set at a third) apply the mutation routine to a randomly chosen member of the population (and keep a copy of it for later printout if it happens to be the best so far).
9. Increment counters and stop if enough work has been done; otherwise loop back to step 2.

[In all experiments reported in this paper, P was equal to 4.]

The main point to notice about this particular GA is that it is incremental rather than generational. A generational GA consists of a main cycle in which most or all of the population is replaced, by their `descendants', on each step. Generational GAs are more common than incremental ones (Forsyth, 1989; Goldberg, 1989). However, in performance there is generally little to choose between these two types of GA (Davis, 1991). The procedure used in IOGA sidesteps certain technical problems connected with fitness scaling (Whitley, 1989), and avoids the expense of sorting as well. To asses the amount of work done by a generational GA that it is necessary to multiply the number of generations by the population size to give the number of structures tested, while with an incremental GA it is only necessary to count the number of offspring made.

4.2 Results on Numeric Data Sets

A program, IOGA, implementing the method described above, was written in C, together with some supporting software including a program called NARC. NARC (Nearest Archetype Classifier) applies the 1-NNC procedure to a full datafile but uses only the instances and features selected by IOGA. These programs were applied to the 10 numeric data sets described in section 2, split into training and test sets. Results obtained are given in Table 4. These were obtained using NARC after IOGA had been run with a population size of 42 for 1200 trials (**not**

1200 generations, thus quite a short run as GA experiments go). To smooth out random fluctuations, the median value from three runs is quoted. Euclidean distance was used, as no significant advantage of City-Block distance had been found in section 2. The figures in the second and third columns are percentage success rates. The last two columns give the number of instances kept by IOGA and the number of variables, or features, selected.

Table 4 -- Results of Applying IOGA to Ten Data Sets.

Data Set	jack-knifed self-test (%)	test on unseen data (%)	Euclidean 1-NNC for comparison (%)	cases selected	variables used
BANKNOTE	100	98	99	4	2
CARDIAC	74.19	64.71	56.86	4	1
DIABETES	94.67	91.43	81.43	6	1
DIGIDAT	72.98	69.23	52.23	70	6
DOGS	89.74	84.21	76.32	9	3
DRINKERS	63.49	48.72	39.74	26	2
FEDS	76.03	64.66	61.26	80	8
IRIS	97.40	94.52	93.15	6	1
QUIN	74.15	67.18	67.69	7	2
ZOOBASE	88.46	79.59	87.76	10	4
Mean =	83.11	76.23	71.54	22.2	3.0

These results show firstly that the `honesty' of jack-knifed self-testing has been lost: all 10 data sets show a decrease from self-test to test on unseen data.

Secondly there has been, as intended, a substantial reduction in size from the full data file to the archetype file, as shown by the number of cases and variables needed for nearest-archetype classification. These figures may be compared with Table 2. The best measure of storage required is the product of number of instances times number of features used. On this measure the mean size of the archetype files, as a percentage of the storage needed by the full training sets, was 3.34% -- a compression ratio of approximately 29 to 1. Thus there is indeed an economization of storage, but this would be of little value if it were accompanied by a loss of accuracy. However, the mean success rate of 76.23% obtained here on

unseen data is actually higher than that of the basic 1-NNC using the full data set (71.54%). A paired (2-tailed) t-test shows that this difference is not significant ($t = 2.08$, $p = 0.067$). Nevertheless, it can be asserted that this genetic subset selection process has incurred no loss of accuracy.

In this connection it is interesting to note that Ritter et al. (1974) compared the performance of three different instance-selection algorithms (condensed, reduced and selective nearest-neighbour classification) on mass-spectrum data and found all three gave slightly worse performance than 1-NNC on the full training data. Likewise, Chang (1974) tested his prototype-based algorithm on some liver disease data and found it had a slightly higher error rate than 1-NNC on unseen data. In other words, it is somewhat unusual to find an instance-selecting version of the 1-NNC that gives better results than the standard algorithm.

5. Concluding Comments

IOGA exhibits to a high degree the compression that, as argued by Wolff (1991) among others, is a hallmark of learning. Indeed on this score it is very impressive, surprisingly so in view of the fact that the bias towards brevity in the fitness function was, in essence, only a tie-breaker. (Presumably the presence of spurious variables and rogue examples also creates selection pressure in favour of sparse subsets.) Nor was this thriftiness in storage bought at the cost of a decline in accuracy. Such compression also assists insight into the data (one of the aims of this selection process) by reducing the size of the problem. Details are given by Forsyth (1995).

The fact that IOGA/NARC gave respectable results in finding archetypal subsets in search spaces ranging from 2^{44} to 2^{519} points after testing a mere 1200 (less than 2^{11}) candidates is, in itself, a vindication of the Darwinian approach to optimization. Once again, evolutionary methods have been shown as robust and effective.

In summary, it may be said of this approach that:

(1) despite jack-knifing, there is a systematic difference between self-test mode and testing on unseen data;
(2) a huge reduction in storage requirement has been effected;
(3) there is no significant decrease in accuracy between the full 1-NNC and IOGA/NARC on unseen data (in fact, on seven out of 10 problems the latter does better);
(4) NARC runs faster than the full 1-NNC, at the cost of a slow

training phase, especially with the larger data sets.

In essence, what IOGA does is exchange the fast training and slow classification normally found with 1-NNC algorithms for the reverse situation, though it should be noted that this is due primarily to the $O(N^2)$ nature of the underlying 1-NNC algorithm and not intrinsic to the GA itself. Thus strictly speaking, this is a problem of scalability rather than pure speed. The straightforward implementation described here is acceptably fast with data sets of modest size (roughly: where the product of variables times cases is less than 8,000) but hits a `combinatorial explosion' quite soon after that. IOGA would need major modifications to deal with larger databases: however, the results outlined in this paper give evidence of sufficient promise to warrant further work on such modifications. Also more work needs to be done to assess the effect of preliminary standardization of feature values, e.g. by subtracting the mean then dividing by the standard deviation. Transformations of this sort are quite common in instance-based learning in order to equalize possibly arbitrary differences in scale among variables, but no such transformation was attempted here. Future studies will test whether this practice would lead to even better results than those reported above.

References

Ackley, D.H. (1987). An Empirical Study of Bit Vector Function Optimization. In: L. Davis, ed., *Genetic Algorithms & Simulated Annealing*. Pitman, London.

Afifi, A.A. & Azen, S.P. (1979). *Statistical Analysis: a Computer Oriented Approach*, 2nd. edition, Academic Press, New York.

Aha, D.W., Kibler, D. & Albert, M.K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6, 37-66.

Allaway, S.L., Ritchie, C.D., Robinson, D. & Smolski, O.R. (1988). Detection of Alcohol-Induced Fatty Liver by Computerized Tomography. *J. Royal Soc. Medicine*, 81, 149-151.

Althoff, K-D., Auriol, E., Barletta, R. & Manago, M. (1995). *A Review of Industrial Case-Based Reasoning Tools*. AI Intelligence, Oxford.

Anderson, E. (1935). The Irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59, 2-5.

Andrews, D.F. & Herzberg, A.M. (1985). *Data: a Collection of Problems from Many Fields for the Student and Research Worker*. Springer-Verlag, New York.

Batchelor, B.G. (1974). *Practical Approaches to Pattern Classification*. Plenum Press, London.

Batchelor, B.G. (1978) ed. *Pattern Recognition: Ideas in Practice*. Plenum Press, N.Y.

Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth, Monterey, California.

Chang, C.L. (1974). Finding Prototypes for Nearest Neighbour Classifiers. *IEEE Trans. on Computers*, C-23(11), 1179-1184.

Darwin, C.R. & Wallace, A.R. (1858). On the Tendency of Species to Form Varieties; and on the Perpetuation of Varieties and Species by Natural Means of Selection. *Paper presented to the London Linnean Society, 1st July 1858*. In: D.C. Porter & P.W. Graham (1993). *The Portable Darwin*. Penguin, London, 86-104.

Dasarathy, B.V. (1991) ed. *Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California.

Davis, L. (1991) ed. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, N.Y.

Devijer, P.A. & Kittler, J. (1982). *Pattern Recognition: a Statistical Approach*. Prentice-Hall, New Jersey.

Fisher, R.A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7, 179-188.

Fix, E. & Hodges, J.L. (1951). Discriminatory Analysis -- Nonparametric Discrimination: Consistency Properties. *Project 21-49-004, Report No. 4*, USAF School of Aviation Medicine, Randolf Field, Texas, 261-279.

Flury, B. & Riedwyl, H. (1988). *Multivariate Statistics: a Practical Approach*. Chapman & Hall, London.

Fogarty, T.C. (1992). First Nearest Neighbor Classification on Frey & Slate's Letter Recognition Problem. *Machine Learning*, 9, 387-388.

Forsyth, R.S. (1989) ed. *Machine Learning: Principles & Techniques*. Chapman & Hall, London.

Forsyth, R.S. (1990). Neural Learning Algorithms: Some Empirical Trials. *Proc. 3rd International Conf. on Neural Networks & their Applications, Neuro-Nimes-90*. EC2, Nanterre.

Forsyth, R.S. (1995). *Stylistic Structures: a Computational Approach to Text Classification*. Doctoral Thesis, University of Nottingham.

Fukunaga, K. & Mantock, J.M. (1984). Nonparametric Data Reduction. *IEEE*

Trans. on Pattern Analysis & Machine Intelligence, PAMI-6(1), 115-118.

Gabor, G. (1975). The eta-NN Method: a Sequential Feature Selection for Nearest Neighbour Decision Rule. In: I. Csiszar & P. Elias, eds., *Topics in Information Theory*. North-Holland, Amsterdam.

Geva, S. & Sitte, J. (1991). Adaptive Nearest Neighbor Pattern Classification. *IEEE Trans. on Neural Networks*, NN-2(2), 318-322.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, Mass.

Hand, D.J. & Batchelor, B.G. (1978). An Edited Nearest Neighbour Rule. *Information Sciences*, 14, 171-180.

Hart, P.E. (1968). The Condensed Nearest Neighbour Rule. *IEEE Trans. on Info. Theory*, IT-14(3), 515-516.

Holland, J.H. (1975). *Adaptation in Natural & Artficial Systems*. Univ. Michigan Press, Ann Arbor.

James, M. (1985). *Classification Algorithms*. Collins, London.

Kelly, J.D. & Davis, L. (1991). Hybridizing the Genetic Algorithm and the K Nearest Neighbors Classification Algorithm. In: R.K. Belew & L.B. Booker, eds., *Proc. Fourth Internat. Conf. on Genetic Algorithms*. Morgan-Kaufmann, San Mateo, California, 377-383.

Kohonen, T. (1988). *Self-Organization & Associative Memory*, 2nd. edition. Springer-Verlag, Berlin.

Kolodner, J.L. (1993). *Case-Based Reasoning*. Morgan Kaufmann, California.

Manly, B.F.J. (1994). *Multivariate Statistical Methods: a Primer*. Chapman & Hall, London.

McKenzie, D.P. & Forsyth, R.S. (1995). Classification by Similarity: An Overview of Statistical Methods of Case-Based Reasoning. *Computers in Human Behavior*, 11(2), 273-288.

McLachlan, G. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, New York.

Michie, D., Spiegelhalter, D.J. & Taylor, C.C. (1994) eds. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Chichester.

Mosteller, F. & Tukey, J.W. (1977). *Data Analysis and Regression*. Addison-Wesley, Reading, Mass.

Mosteller, F. & Wallace, D.L. (1984). *Applied Bayesian and Classical Inference: the Case of the Federalist Papers*. Springer-Verlag, New York.

Pei, M., Goodman, E.D., Punch, W.F. & Ding, Y. (1995). *Genetic Algorithms for Classification & Feature Extraction*. Technical Report: Michican State Univeristy, GA Research Group, Engineering Faculty.

Quinlan, J.R. (1987). Simplifying Decision Trees. *Int. J. Man-Machine Studies*, 27, 221-234.

Reaven, G.M. & Miller, R.G. (1979). An Attempt to Define the Nature of Chemical Diabetes using a Multidimensional Analysis. *Diabetologia*, 16, 17-24.

Rechenberg, I. (1973). *Evolutionsstrategie -- Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Halzboog, Stuttgart.

Ritter, G.L., Woodruff, H.B., Lowry, S.R. & Isenhour, T.L. (1974). An Algorithm for a Selective Nearest Neighbour Decision Rule. *IEEE Trans. on Info. Theory*, IT-21(6), 665-669.

Siedlecki, W. & Sklansky, J. (1989). A Note on Genetic Algorithms for Large-scale Feature Selection. *Pattern Recognition Letters*, 10, 335-347.

Smith, J.E., Fogarty, T.C. & Johnson, I.R. (1994). Genetic Selection of Features for Clustering and Classification. *IEE Colloquium on Genetic Algorithms in Image Processing & Vision*. London.

Swonger, C.W. (1972). Sample Set Condensation for a Condensed Nearest Neighbour Decision Rule for Pattern Recognition. In: S. Watanabe, ed., *Frontiers of Pattern Recognition*. Academic Press.

Tomek, I. (1976). An Experiment with the Edited Nearest-Neighbour Rule. *IEEE Trans. on Systems, Man & Cybernetics*, SMC-6(6), 448-452.

Ullman, J.R. (1974). Automatic Selection of Reference Data for Use in a Nearest Neighbour Method of Pattern Classification. *IEEE Trans. on Info. Theory*, IT-20(4), 541-543.

Weiss, S.M. & Kulikowski, C.A. (1991). *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA.

Whitley, D. (1989). The GENITOR Algorithm and Selective Pressure: why Rank-Based Allocation of Reproductive Trials is Best. *Proc. Third Internat. Conf. on GAs*, 116-121, Morgan-Kaufmann, Palo Alto, CA.

Wolff, J.G. (1991). *Towards a Theory of Cognition and Computing*. Ellis Horwood, Chichester.