# Application of Genetic Algorithms to Data Mining

**Robert E. Marmelstein**
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433-7765

**Abstract**

Data Mining is the automatic search for interesting and useful relationships between attributes in databases. One major obstacle to effective Data Mining is the size and complexity of the target database, both in terms of the feature set and the sample set. While many Machine Learning algorithms have been applied to Data Mining applications. There has been particular interest in the use of Genetic Algorithms (GAs) for this purpose due to their success in large scale search and optimization problems. This paper explores how GAs are being used to improve the performance of Data Mining clustering and classification algorithms and examines strategies for improving these approaches.

## What is Data Mining?

Data Mining is an umbrella term used to describe the search for useful information in large databases that cannot readily be obtained by standard query mechanisms. In many cases, data is mined to learn information that is unknown or unexpected (and therefore interesting). To this end, Data Mining applications typically have as their goals one or more of the following:

**Finding Patterns:** Patterns are clusters of samples that are related in some way relative to the database attributes. Discovered patterns can be unconstrained ("Identify all clusters consist of at least 10 percent of the population") or directed ("Find population clusters that have contracted cancer").

**Deriving Rules:** Rules describe associations between attributes. Accordingly, clusters provide a good source for rules. Rules are typically defined in terms of IF/THEN statements. An example rule might be:

IF ((Age > 60) AND (Smoker = TRUE))
THEN Risk(Cancer) = 0.7;

**Classifying Unknown Samples:** In many cases, users seek to apply Data Mining algorithms to categorize new data samples into existing classes (i.e., "What kind of car is a 35 year old, white, suburban, single, female teacher most likely to drive?").

## The Curse of Dimensionality

While Data Mining is a relatively new branch of Artificial Intelligence, it involves many of the same problems as more established disciplines like Machine Learning (ML) and Statistical Pattern Recognition (SPR). One of the most vexing of these problems is that of high dimensionality in data. High dimensionality refers to the situation where the number of attributes in a database is large (nominally 20 or more). A highly dimensional data set creates problems in terms of increasing the size of the search space in a combinatorially explosive manner (Fayyd, Piatetsky-Shapiro, & Smyth 1996). In addition, the growth in the search space as more features are added increases the number of training samples required to generate reliable results. In order to develop effective Data Mining algorithms, the curse of dimensionality must be overcome.

One way to remove the curse of dimensionality is to reduce the list of attributes to those that are really important for distinguishing between distinct classes within the data. This process (called feature selection) attempts to find the set of features that results in the best classifier performance (i.e., the set that has the fewest mis-classifications). In addition, it is possible to further improve classifier performance by assigning relative weights to each feature vector. Each feature can then be multiplied by its assigned weight in order to optimize the overall classifier performance. Thus, the contribution of those features that help separate classes will be enhanced while the contribution of those that don't will be diminished. The result of this transform is illustrated in Figure 1. Note that the classes in (a) are better separated after the transform is applied in (b). The process of computing such a

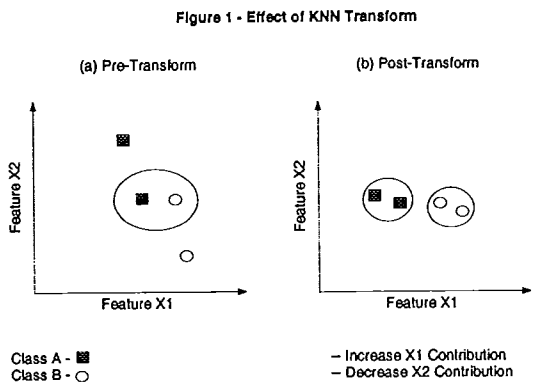


Figure 1: Effect of KNN Transform

weighting vector for a given attribute is called feature extraction. While there are a number of conventional feature selection algorithms, such as forward selection (Lippman 1993), these techniques are greedy in nature and do not accomplish feature extraction.

## Why use Genetic Algorithms?

GAs are global search algorithms that work by using the principles of evolution. Traditionally, GAs have used binary strings to encode the features that compose an individual in the population; the binary segments of an individual that represent a specific feature are known as chromosomes. Binary strings are convenient to use because they can be easily manipulated by GA operators like crossover and mutate. Binary chromosomes can also be used to represent non-binary numbers that have integer and floating point types. Given a problem and a population of individuals, a GA will evaluate each individual as a potential solution according to a predefined evaluation function. The evaluation function assigns a value of goodness to each individual based on how well the individual solves a given problem. This metric is then used by a fitness function to determine which individuals will breed to produce the next generation. Breeding is done by crossing existing solutions with each other in order to produce a better solution. In addition, a mutation factor is present which will randomly modify existing solutions. Mutation helps the GA break out of local minima, thus aiding the search for a globally optimum solution. While there is no guarantee that GAs will find an optimal solution, their method of selection and breeding candidate solutions means a pool of "good" solutions can be developed given enough generations.

GAs are of interest because they provide an alternative to traditional ML algorithms, which begin to show combinatorial explosion (and therefore poor computational performance) for large search spaces. GAs have been shown to solve problems that were considered intractable due to an excessively large search space (Punch *et al.* 1996). In short, GAs can help lift the curse of dimensionality.

## A Hybrid Approach - GAs and the K-Nearest Neighbor Classifier

One approach that combines both feature selection and extraction was developed by the Genetic Algorithms and Research Application Group (GARAGe) at Michigan State University (MSU) (Punch *et al.* 1993). Based on the earlier work of (Sklansky & Siedlecki 1989), this technique uses a hybrid approach that combines a GA with a K-Nearest Neighbor (KNN) classifier. The KNN algorithm works by assigning a vector to the class most frequently represented by the K nearest samples of known classification (Duda & Hart 1973). For example, if K=5 then we assign class based on the five nearest neighbors of a given vector. Of the five closest neighbors, if three belong to class A and two to class B, then the vector will be assigned to class A (the majority class). The KNN technique is non-parametric is nature because it is not necessary to assume a form for the probability density function (PDF) in advance. While KNN is a powerful classification technique, it is suboptimal in that it will usually lead to an error rate greater than the minimum possible (Bayes error rate) (Therrien 1989). The accuracy of the KNN algorithm can be improved, however, by using feature extraction to tune the attribute weights used to compute distance between samples. In this case, the distance (D) between distinct feature vectors is computed as:

$$D = \sqrt{(X-Y)^t W (X-Y)}$$

where,
X and Y are distinct feature vectors and
W is the weighting vector

A GA can be utilized to derive the optimal weighting vector (W) for the feature set. The weighting vector is modeled by encoding each feature as a distinct chromosome in the GA's schema. The evaluation function can then measure the performance of each candidate weighting vector based on the error achieved with the KNN classifier. The KNN error is measured by multiplying each known feature vector by the weighting represented by the GA candidate. If the KNN classifier maps the feature vector to the incorrect class, then this counts as an error. When all known vec-

tors are processed, the total KNN error for the candidate weighting vector is computed. Only the most promising weighting vectors are selecting for breeding by the GA. Using this process, the weights for features that help distinguish between classes will be increased; conversely, the weights for irrelevant features will decrease (toward zero). The goal is to produce a weighting vector that minimizes the KNN classifier error for the given training data set.

## Algorithm Improvements

While the above approach works, it can be very expensive computationally. According to (Punch *et al.* 1993), processing a large feature set (96 features) took 14 days on a SPARC workstation. Clearly, this type of performance is unacceptable when compared to existing greedy techniques. While results were obtained faster using a parallel configuration, it is possible to modify the algorithm to produce results more quickly.

The primary modification is to break the GA search up into two phases. The first phase conducts a binary (0/1) search for the most promising features. The GA structure for this phase also contains a chromosome that encodes the value of K. After the first phases is complete, the features that reduce the KNN classifier error are assigned a 1; those that are irrelevant or detrimental to classification are assigned a 0. The second phases uses the results of the first phase to evaluate each individual in its gene pool. In this phase, each feature is encoded in a chromosome with multiple bits (in this case four bits are used). The binary value of each chromosome is then scaled by a factor based on the results of the previous phase. The basic concept is to scale good features to enhance their contribution (i.e., greater than one) and scale bad features to diminish their contribution (i.e., less than one). In this phase, the optimal K value derived from the first phase in utilized. Since the search space of the first phase is much smaller than that of the second phase, fewer generations are needed to search the first phase. The result can then be used to better focus the search for the optimal feature weights in the second phase. By using the results of the shorter binary search in the first phase, we can speed up the overall progress of the GA search.

## Mining for Edible Mushrooms

To test my proposed improvements, I chose an often referenced SPR benchmark, the Mushroom database. This database (constructed by the Audobon Society) identifies 22 features that can help determine if a mushroom is poisonous or edible; in all, the database contains 8124 samples. Both the MSU GA/KNN hybrid and my modified version were run against this data. While both algorithms eventually achieved a 0% error rate against the training set (508 samples), the MSU hybrid took fifteen generations to achieve this result, while my version took a maximum of three generations (see Table 1). Even though my algorithm reached its goal quickly, I let it run for 5 generations in order to get a diverse set of candidate weighting vectors. During this period, a total of 8 distinct feature sets were produced that achieved a training error of 0.0%.

After the feature extraction process was finished, the Link Net classifier system was used to process the selected features on both the training and data sets. The results of this experiment are shown in Table 2. The first three entries in Table 2 are the results achieved with the reduced feature set derived from the hybrid GA/KNN classifier. Obviously, not all the feature sets performed equally well. The best feature set had a test error rate of 0.39% while the worst had a test error rate of 1.65%. The lesson to be learned from this is not to settle for the first set of promising results. The GA should be run long enough to get a pool of candidates; from these, the best set can be derived by evaluating it against the test data.

Table 2 illustrates how the results on this dataset compared to other types of classifiers. A Gaussian classifier was also applied to the dataset. This classifier was likewise run using Link Net with a full covariance matrix for each training class. Good results were achieved using the full feature set and a 50/50 division of samples between the training and test data sets. The results with the derived feature set were much less promising. The best overall performance on this data set were achieved with the REGAL classifier system (Neri & Saitta 1996). REGAL is a genetic-based, multi-modal concept learner that produces a set of first order predicate logic rules from a given data set. These rules are, in turn, used to classify subsequent data samples. While REGAL was able to completely eliminate test error, it did so with a much larger training set (4000 samples). It must also be noted that REGAL took 164 minutes (using 64 processors) to derive the rule set that optimized its performance. In addition, when REGAL processed a comparable training set (500 samples), its performance was much closer to that of the modified GA/KNN classifier. Lastly, a neural network classifier (Yeung 1991) achieved a 0.9% error rate with a significantly reduced training set (300 samples).

## Conclusion

Several important points can be inferred from the above results. First, the modified GA/KNN algorithm

Table 1: GA Parameters for Mushroom Data Set

| Approach | GA Parameters | Description |
|---|---|---|
| Original (MSU) GA/KNN Hybrid | Population Size = 20<br>Probability of Mutation = 0.01<br>Probability of Cross Over = 0.9 | Took up to 15 generations to converge to 0.0% training error |
| Modified GA/KNN Hybrid | Population Size = 20<br>Probability of Mutation = 0.01<br>Probability of Cross Over = 0.9 | Took up to 3 generations to converge to 0.0% training error |

Table 2: Mushroom Data Set Results for KNN Classifier

| K Value | Feature Set Size | Training/Test Set Size | Training Error (%) | Test Error (%) |
|---|---|---|---|---|
| K=1 | 12 | 508/7616 | 0.0 | 0.39 |
| K=1 | 12 | 508/7616 | 0.2 | 1.65 |
| K=1 | 10 | 508/7616 | 0.59 | 0.93 |
| K=1 | 22 | 508/7616 | 0.0 | 0.47 |
| K=3 | 22 | 508/7616 | 0.0 | 0.54 |

Table 3: Comparison to Other Classifiers

| Classifier Type | Feature Set Size | Training/Test Set Size | Training Error (%) | Test Error (%) |
|---|---|---|---|---|
| Gaussian | g22 | 4062/4062 | 0.17 | 0.221 |
| Gaussian | 12 | 508/7616 | 2.76 | 2.991 |
| REGAL (Neri & Saitta 1996) | 22 | 500/7624 | N/A | 0.42 |
| REGAL | 22 | 4000/4124 | N/A | 0.00 |
| Neural Network (Yeung 1991) | 22 | 300/7824 | N/A | 0.92 |

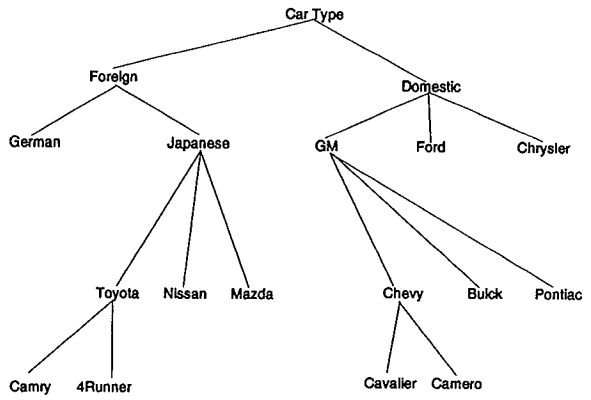


Figure 2: Auto Manufacturer Taxonomy

is able to successfully reduce the dimensionality of the data set while actually improving classification accuracy. Second, the classification accuracy was improved using a smaller data set for training. This is important because training (and testing) a classifier with a smaller data set is much more efficient using a larger one. Third, because the initial phase of the algorithm had a smaller search space (as compared to the second phase), this approach was able to minimize training error faster than the MSU algorithm. In addition, incorporation of the K parameter into the GA schema helped to further improve classification accuracy.

## Future Directions

While the above results look promising, the basic approach has weaknesses for data sets that are not homogeneous and/or unimodal within a given class. By using a single weighting distribution for all the data, we are making the underlying assumption that all class distributions have the same basic shape for a given feature set; this is obviously not the case in most real world data sets. In addition, the distribution within a given class is often multi-modal. For example, there may exist different subclasses of poisonous mushrooms that are characterized by mutually exclusive features. Under these circumstances, using a single weighting vector as a transform will not produce optimal results. Instead, an approach for evolving weighting vectors tailored to each data class and/or subclass needs to be explored.

Using GAs to perform domain taxonomy substitutions as part of the search process may also improve feature selection and classification. In any domain, a taxonomy can be constructed that describes hierarchical relationships between objects in the domain. Figure 2 contains an example of such a taxonomy. This figure shows the hierarchical relationship between car brand, company, and brand origin (foreign vs domestic). When searching for the best feature set, the GA functionality could be modified to substitute the GM label for its related brands. An alternative substitution would be the domestic label for all car brands produced in North America. Performing such substitutions may reveal clusters that go undetected when viewing the data through the lens of a label at the bottom of the taxonomy hierarchy. For example, substituting brand origin for the brand itself may reveal a tendency for twentysomethings to buy Japanese cars. As a result, performing these substitutions can find a feature set that yields rules with better support. Because multiple hierarchies may describe a given data set, its important that the search take these into account as well. Since examining these relationships substantially increases the size of the search space, continuing to use GAs as the search mechanisms is the logical choice.

## References

Duda, R., and Hart, P. 1973. *Pattern Classification and Scene Analysis.* John Wiley & Sons. pages 95-99.

Fayyd, U.; Piatetsky-Shapiro, G.; and Smyth, P. 1996. *Advances in Knowledge Discovery and Data Mining.* The MIT Press.

Lippman, R. 1993. *Link Net Users Guide.* MIT Lincoln laboratories.

Neri, F., and Saitta, L. 1996. Exploring the power of genetic search in learning symbolic classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(11):1135–1141.

Punch, W.; Goodman, E.; Pei, M.; Lai, C.-S.; Hovland, P.; and Enbody, R. 1993. Furthe research on feature selection and classification using genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms*, 557–564.

Punch, W.; Goodman, E.; Raymer, M.; and Kuhn, L. 1996. Genetic programming for improved data mining–application to the biochemistry of protein interactions. Downloaded from MSU CS Dept WWW Site.

Sklansky, J., and Siedlecki, W. 1989. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters* 10:335–347.

Therrien, C. W. 1989. *Decision Estimation and Classification.* John Wiley & Sons. page 130.

Yeung, D. 1991. A neural network approach to constructive induction. In *Proceedings of the Eighth International Conference on Machine Learning*, 228–232.