EVOLUTIONARY NEAREST NEIGHBOUR CLASSIFICATION FRAMEWORK

Amal Shehan Perera
Department of Computer Science & Engineering
University of Moratuwa.
Sri Lanka.
shehan@uom.lk
(701) 866-6433

Topic Area Data Engineering: Classification, Data Integration, Data Mining
Topic Area Software Engineering: Software Components

D.G.N. Dayaratne
Computer Science & Eng.
University of Moratuwa.
Moratuwa, Sri Lanka.

William Perrizo
Computer Science Department
North Dakota State University
Fargo, ND 58105 USA
William.perrizo@ndsu.edu
(701) 231-7248
EVOLUTIONARY NEAREST NEIGHBOUR CLASSIFICATION FRAMEWORK

Amal S. Perera, D.G.N. Dayaratne  
Computer Science & Eng.  
University of Moratuwa.  
Moratuwa, Sri Lanka.  
shehan@uom.lk

William Perrizo  
Computer Science  
North Dakota State University  
Fargo, ND 58105  
william.perrizo@ndsu.edu

Abstract

Data classification attempts to assign a category or a class label to an unknown data object based on an available similar data set with class labels already assigned. K nearest neighbor (KNN) is a widely used classification technique in data mining. KNN assigns the majority class label of its closest neighbours to an unknown object, when classifying an unknown object. The computational efficiency and accuracy of KNN depends largely on the techniques used to identify its K nearest neighbours. The selection of a similarity metric to identify the neighbours and the selection of the optimum K as the number of neighbours can be considered as an optimization problem. The optimizing parameters for KNN are value for K, weight vector, voting power of neighbours, attribute selection and instance selection. Finding these values is a search problem with a large search space. Genetic Algorithms (GA) are considered to provide optimum solutions for search problems with a large search space. The search space is defined by the application domain. There are multiple real world classification applications that can utilize a parameter optimized KNN. Due to this, there is various research work carried out on using Genetic Algorithms for optimizing KNN classification.

Even though multiple instances of research had been carried out on using GAs to optimize KNN there is no software framework available, which could be easily adapted to various application domains. This work is aimed towards building a framework to carry out the optimization of KNN classification with the help of a Genetic Algorithm. The developed framework provides a basic backbone for GA optimization of KNN while providing sufficient flexibility for the user, to extend it to specific application domains.

1 INTRODUCTION

Data mining is extracting knowledge from large amounts of data. In a broad sense, data mining is the process of discovering interesting knowledge, from large amounts of data stored in databases, data warehouses, or other information repositories [1][2]. Classification can be defined as the process of finding a mechanism to describe and distinguish classes or concepts, with the objective of predicting the class labels of objects, where class labels are not known. A set of training data (i.e. a set of objects whose class label is known) is analyzed for this process. In this work the focus is on k-nearest neighbor classification. KNN also known as instance-based classification and it is a widely used technique for pattern recognition and machine learning. It is considered to be a simple, yet efficient technique for supervised learning problems. Though this has a history of more than thirty five years, it is still widely studied and used. When trying to solve a new problem people tend to look at solutions to similar problems they have already solved. K-nearest neighbor uses a similar technique to solve the problem of identifying the class of an unknown instance. It decides on which class to place a new instance by examining k number of most of the similar instances. The new class is assigned to the majority class in the neighborhood.

In order to apply KNN it is necessary to find a measure of the distance between attributes in data and then calculate it. This is easy for numerical data, but categorical variables will need special handling, for example how to calculate the distance between blue and green. Once it is possible to calculate the distance between cases, a set of already classified instances will be selected as the basis for classifying unknown instances. The size of the neighborhood (value of k), importance of each neighbor (weight on each neighbor) and importance of each attribute (weight of each attribute) are some of the parameter considerations when implementing a KNN classification application.

Furthermore, KNN requires a large computational load because the calculation time increases along with the factorial of the total number of instances in the training data set. While it is a rapid process to apply a decision tree or neural network to a new case, KNN requires a new calculation to be made for each new case. Performance of KNN classification depends largely on efficient and accurate selection of k nearest neighbours.

Genetic algorithms (GAs) are a set of randomized search and optimization techniques guided by the principle of evolution and natural selection. GAs solve optimization problems by manipulating a population of chromosomes. Chromosome is the genetic representation of a solution. Appropriateness of the solution is measured by defining a fitness function. GAs are considered as an adaptive and robust global search method, suitable in situations where the search space is large. Much research effort has been put into selecting or scaling features to improve KNN classification using Genetic Algorithms.
This work focuses on developing a framework for KNN classification using a Genetic Algorithm. It is expected that this framework will benefit a new practitioner to easily adapt to a new application domain with a lower adaptation cost.

The Search for an effective weight vector, selection of features, instance selection, voting granularity of neighbours for KNN are hard optimization problems with a very large search space. In general genetic algorithms have been shown to be good at solving hard optimization problems.

A Genetic algorithm had been used by Siedlecki and Sklanski [3] for feature selection. In their work a genetic algorithm was used to find a vector with ones and zeros, which was used by the KNN classifier. Each one in the vector corresponded to a feature used. The obtained vector was required to give a performance level above a threshold value and at the same time it had to try to minimize the number of ones. Kelly and Davis [4] in their implementation uses a genetic algorithm to find the weight vector used by KNN. This weight vector consists of real values to be multiplied with each feature resulting in a new feature vector that is linearly related to the original one. Punch et al [5] had taken a similar approach as Kelly and Davis. In their approach weights were allowed to vary over a scale of 0 to 10. If a weight had gone below 1 it was approximated to 0. Though this is quite similar to work carried out by Kelly and Davis, they had incorporated parallel processing to reduce the compute time. Raymer et al [6] had further extended this approach by introducing a masking vector for feature selection. In their approach they had used both a feature weight vector and a masking or feature selection vector. Feature vector values were real valued while mask vector values were either 0 or 1. Each feature was multiplied by both the feature weight value and the masking value, prior to classification by the KNN classifier. In this approach GA could test the effect of removing the feature by setting the masking value to be 0. This allowed the reintroduction of the feature without losing the previously learnt weight completely. As this method introduced a significant importance on a single bit of the chromosome a second method was devised. Here each feature was associated with m-mask bits. If majority of m mask bits were 1 then the feature was selected for the classification. In both methods feature could be removed from classifier if the weight approached zero.

Kucheva [7] had experimented with three fitness functions for use in feature editing. Genetic Algorithm had been used for simultaneous feature selection and editing for nearest neighbor classification [8]. In their experiment each chromosome S was a binary string consisting of N + n bits where N is the number of instances in the training set and n is the number of features. The bit had value 1 if the respective element was included in the reference set. According to their experiments GA had a good compromise with both high accuracy and a moderate number of parameters. Chen et al.[9] have proposed a multi-objective approach for nearest neighbor classification. They had utilized an intelligent multi objective evolutionary algorithm to simultaneously edit and compact the instances and the feature sets. In this method they had used an intelligent crossover operation for the GA. The crossover operation is carried out after economically estimating the contribution of individual genes to a fitness function and intelligently selecting better genes for child chromosomes.

Frederic [10] has proposed a hybrid approach. The first phase of the procedure is a pure GA. The objective was to promote diversity among the chromosome pool in order to remove unused features and prepare for the second step. The second step was to find a set of candidate solutions. In the second step instead of diversity elitism, was promoted.

A successful application for a real world application is presented by Perrizo and Perera [11]. Using a GA pulmonary embolisms are detected in a medical data base as part of the winning entry for the 2006 KDD Cup data mining competition. In this application the GA optimization is done for training set without the knowledge of the actual test set as it happens in a real world scenario. A genetic algorithm is used to select features and the k to arrive at the optimal solution.

There are some generic GA frameworks available for a data mining practitioner to use. N-genes is a Genetic Algorithms and Programming toolkit written in java. It has Stack-based Genetic Programming The Genetic Programming implemented in n-genes relies on linear postfix programs. Also all parts of evolutionary computing have been made into components, through Design Patterns. The code is written with high performance computing considerations.Another framework is the Watchmaker. It is an extensible, high-performance, object-oriented framework for implementing platform-independent GAs in Java. The framework provides type-safe, non-invasive evolution for arbitrary representations. It has Multi-Threaded Evolution Engine. And it takes advantage of parallelism to improve performance on multi-core and multi-processor machines.

JGAP is a Genetic Algorithms and Genetic Programming component provided as a Java framework. It provides basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions. The code base is stable and easy to maintain. It is also optimized for performance. As with most of the other frameworks it uses abstract classes and interface classes where necessary to provide Customizable Genetic Operators and Genes. JGAP has been cited on various publications, applications and research projects.
2 FRAMEWORK DESIGN

A good framework should provide a basic conceptual structure or a skeletal, openwork, or structural frame on which a user can construct their own application[13]. A good application framework provides proven and reliable design patterns and code architecture to an application, which leads to increasing development productivity. Frameworks grew out of a realization that many common design patterns could have a single implementation, and that implementation could be used on multiple projects. Often, a framework is built as the result of an iterative process: As projects that have some common elements are developed, reusable components are created to reduce the need to continually re-create typical functionality. As these components are re-used and made more generic and flexible, they become the backbone of a framework.

This work focuses on developing a framework for optimizing KNN with a Genetic Algorithm as a solution for the following problems.

- Extensive research has been carried out on using GAs for optimizing KNN algorithms. All these work had been carried out targeting a specific application. Although this is extensively used there is no framework available, which could be easily adapted for various application domains. The framework is expected to provide basic mechanisms required for optimizing KNN and provide easy adaptability for various application domains.

- There are data mining frameworks like Weka[15]. But these are too generalized for all aspects of data mining. Therefore it is not easy to obtain an optimum solution implementation for genetic algorithm optimization of KNN.

As a solution to the above problems the new framework is developed to provide adaptability in the following manner.

- The framework would provide basic mechanisms for data representation, optimized data manipulation and components for KNN implementation.

- Also the framework will provide easy extensibility for a specific domain by the user. That is a generalized implementation will be provided for a user to carry out his own implementation by extending the functionality.

The framework tries to provide a solid implementation for general data mining tasks while providing easy adaptability for application specific implementations. In other words framework is intended to be specific for GA optimization of KNN and general enough to extend to suite a specific application domain. By making the framework available for software development community as Open Source product it is expected that this framework would be used for new applications of KNN optimization.

The objective of the framework is to carry out optimization of KNN classification. Optimization parameters for KNN are, Finding optimum value for K, Finding the weight vector for attributes, Finding voting power of neighbours, Carrying out attribute selection, and Carrying out instance selection. For the current implementation of the framework optimization with the first two parameters would be made possible, but the framework is designed in such a way that it can be easily extended to optimize other parameters. The framework was designed to provide implementations for basic features required for KNN optimization with GA. Framework encompasses data representation, data preprocessing, GA implementation, solution evaluation, and similarity calculation.

Data mining deals with large data sets. Data could be available in various forms. They could be stored in an RDBMS (e.g. Data Warehouse, Operational databases), XML data stores or flat files. This implementation framework will only support flat files. It is possible for it to be extended to support other data forms later.

The value of an attribute of an instance, of a data set is a measurement of the quantity that the attribute refers to. When representing data it is important to identify how to represent different data types. The data types of attributes could be Integer, Real, Date or Nominal. Integer, Real or Date type attributes can be readily converted to numeric attributes though it might be necessary to have some kind of special handling. Therefore in a broad sense the two types of attributes are Numeric and Nominal [14]. Nominal attributes take on values in a pre-specified finite set of possibilities. They are also called categorical data.

The interpretation of attributes is based on the learning scheme used. In the KNN algorithm the attributes should be used for the distance calculation. With numeric attributes it is possible to apply a straightforward technique. For nominal attribute representation this would be application specific. It is possible to have a two-level distance i.e. 1 if the two values are different, or 0 if the two values are the same. If the attributes were Boolean, this gives only two possible values and would be a good technique. Also if the nominal type is ordinal it may be possible to assign numeric values depending on the order. Ordinal quantities are those, that can be ranked or ordered by categories. For example weather temperature conditions could be HOT, MILD or COOL. This could be ordered as HOT > MILD > COOL. Sometimes it is possible to do a genuine mapping between nominal quantities and numeric scales. For example postal zip codes could indicate areas that could be represented by geographical coordinates. Also it is possible to represent nominal quantities as several more numeric attributes. For example how to represent color. Color could be...
Therefore the attribute representation is very much dependent on the application domain. Thus the framework should have enough flexibility to extend the representation to application’s requirement. In this implementation the data representation is limited to ARFF [15]. A description ARFF format is given in Appendix A. This is a standard way of representing data sets that consist of independent, unordered instances that do not involve relationships between instances. But the framework provides the extensibility to support other file formats.

Incomplete, inconsistent and noisy data are common properties of large real world data sets [1][14]. Incomplete data can occur due to several reasons. Attributes of interest may not always be available. Sometimes data may not be included simply because it was considered not important at the time of entry. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions. Data that were inconsistent with other recorded data may have been deleted. Furthermore, the recording of the history, or modifications to the data may have been overlooked. Missing data, particularly for tuples, with missing values for some attributes, is a problem that should be addressed in a data mining framework.

In this framework implementation we only support treatment of incomplete data. That is filling out missing values. The design is done making it possible to add other data cleaning tasks later. It is important to have data in ratio scales[14][1], which in turn require data normalizing. Attributes are often normalized to lie in a fixed range like 0 and 1. There are several methods of employing normalization. This implementation will only provide normalization by dividing the complete column by the maximum value. It is also possible to extend it to implement other types of normalization. The following diagram shows the overall component level design of the framework.

KNN is a computationally intensive algorithm, as to find the nearest neighbors of an instance it has to calculate the similarity between each and every other instance. Therefore it is very important to represent data internally in a very efficient structure.

One option is to go for a data structure like Vector where all the types of data can be held. For example a similar approach to the Weka data mining framework. The “Instances” class in Weka holds data in a Vector structure common to all data types. But when we consider KNN specifically, data representation for KNN should lead to a numeric representation optimum performance. Internal data representation is required to calculate similarity, that is the distance between instances. Also, even for nominal type data, the data can be converted to a numeric form. For example if the data is of Boolean type it can represent as 1 and 0. If the nominal values are of Ordinal type it can be converted to representative numeric values. This would be application domain specific and framework should have flexibility to extend, it if necessary. The attribute information that is Meta data of attributes is stored in a Attribute class. Attribute class will have information like type, name, the possible values for Nominal type, Date format for Date type attributes, etc. For nominal values the attribute instance will hold the values in a Hashtable. The data is represented using a two-dimensional numeric array. This data will be held in an Instances class. In the Instances class data is to be held in a 2D array of type double. Compared to java vectors, arrays have better performance. Though it can be considered marginal in smaller data sets, with larger data sets the increase in performance is significant. KNN is based on distance calculations. To calculate similarity it is imperative that the data should be represented as numeric quantities. Therefore having an array to represent data is justified, as arrays can be considered as the simplest and the most efficient data structure. Also after reading the data set the size of the data set doesn’t vary. Therefore having data structure of fixed size is justified if it gives better performance.

Instance class will have information of each attribute stored in a Vector of Attributes. This Vector is implemented as a FastVector without synchronized methods. This is done as to improve the performance of the Vector. The values to be stored in the array for nominal type data is the hash index of the value.

3 RESULTS

The overall objective of the work is to build a framework for GA optimized KNN that is adaptable, extensible, accurate and scalable. In this section we show the adaptability of the framework by describing classification results for three publicly available data sets. Two data sets extracted from UCI Machine Learning Repository and the other one from the data made available for the Physiological Data Modelling Contest at ICML 2004. These data sets were selected to
highlight the diversity of the applications in order to show the adaptability of the framework. Testing was carried out with a Pentium® 4 CPU 2.8 GHz with 224 MB of memory. The memory was kept at a minimum intentionally to evaluate the efficiency of the implementation. The following table gives a summary of the datasets and the results obtained using the suggested framework in this paper.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Adult</th>
<th>Abalone</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>48842</td>
<td>4177</td>
<td>189961</td>
</tr>
<tr>
<td>Attributes</td>
<td>14</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Properties</td>
<td>Large number of Nominal attributes</td>
<td>Multi class classification</td>
<td>Large number of instances</td>
</tr>
<tr>
<td>Missing Data</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Classification</td>
<td>Salary &gt; 50K</td>
<td>Age</td>
<td>Gender</td>
</tr>
<tr>
<td>Source</td>
<td>UCI</td>
<td>UCI</td>
<td>ACML</td>
</tr>
<tr>
<td>Training Time req.</td>
<td>&lt; 1h</td>
<td>38.5h</td>
<td>12.9h</td>
</tr>
<tr>
<td>Accuracy</td>
<td>100%</td>
<td>84%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 1 Datasets Used for Evaluation

For all three data sets the following similarity metric was used. As indicated earlier the framework provides a very easy mechanism to define your own similarity metric that suites the application making the framework adaptable to various application domains. The implementation of the following functionality requires changing few lines of java code.

For all three data sets the following similarity metric was used. As indicated earlier the framework provides a very easy mechanism to define your own similarity metric that suites the application making the framework adaptable to various application domains. The implementation of the following functionality requires changing few lines of java code.

\[
\text{Nominal variables, } d_i = \begin{cases} 
W_i X_i = Y_i \\ 0 \quad X_i \neq Y_i 
\end{cases}
\]

\[
\text{Numeric attributes } d_i = W_i (X_i - Y_i)
\]

\[
\text{Similarity } \text{Sim} = \frac{1}{n-1} \sum_{i=0}^{n-1} d_i
\]

Similarly, the fitness function can also be defined very easily for a given application. For example the following fitness calculation was used for the Abalone data set.

\[
\text{Fit} = \left( \left( \sum_{i=0}^{n-1} (1 - \text{Conf}_i)^2 \right) \right) \times 100
\]

The above fitness is computed assuming an appropriate confidence measure for the respective application.

The following figure shows the optimized weights for each of the weight attributes for the Abalone dataset. By observing the weights it can be clearly seen that the most important predictive attributes in predicting the actual age of an abalone sample are getting the higher weight values. This clearly indicates the learning capability of the model.

![Figure 2 Optimized Weights for Abalone Data](image)

The following figure shows the optimized weights for the adult data set. Once again the weight values show how the similarity metric of generic application framework is adapted towards the application of predicting salary.

![Figure 3 Optimized Weights for Adult data set](image)

The following figure shows the optimization progress for the Gender data.

![Figure 4 Optimization Progress for Gender Data](image)
training time with the data set size. As expected it is clear that there is a gradual increase in the time with the increase of the data set size.

![Figure 5 Training Time Vs Data File Size](image)

4 CONCLUSION

K nearest neighbor (KNN) is a widely used classification technique in data mining. The computational efficiency and accuracy of KNN depends largely on the techniques used to identify its K nearest neighbours. The selection of a similarity metric to identify the neighbours and the selection of the optimum K as the number of neighbours can be considered as an optimization problem. Genetic Algorithms (GA) are considered to provide optimum solutions for search problems with a large search space. The search space is defined by the application domain. There are multiple real world classification applications that can utilize a parameter optimized KNN. Even though multiple instances of research had been carried out on using GAs to optimize KNN there is no software framework available, which could be easily adapted to various application domains. This work was an attempt aimed towards building a framework to carry out the optimization of KNN classification with the help of a Genetic Algorithm.

The main objective of the GA optimized KNN framework was to make it extensible, adaptable, accurate and scalable. As indicated in the results section with results from a diverse array of applications it is clear that the framework is extensible and adaptable. It is important to note that the applications were implemented within a very short period of time. But it needs to be properly evaluated through a larger user sample to make any concrete statements with respect to the adaptability and the extensibility of the framework.

As shown in the results section it is clear that the framework provides very accuracy levels. The weight values also indicate the learning capability of the framework. With respect to scalability the framework provides acceptable levels of compute times. But it is obvious that further improvements can be made to the framework. The framework is publicly available on SourceForge.net as “gaKnn”.

5 REFERENCES