THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Improving Performance of k-NN Classifier using Prototype Generation: Survey

Ansari Mariyam
Student, Department of Computer Engineering,
K. K. Wagh Institute of Engineering and Research, Maharashtra, India
Bodke Priya
Student, Department of Computer Engineering,
K. K. Wagh Institute of Engineering and Research, Maharashtra, India
Desai Aishwarya
Student, Department of Computer Engineering,
K. K. Wagh Institute of Engineering and Research, Maharashtra, India
Dusane Aishwarya
Student, Department of Computer Engineering,
K. K. Wagh Institute of Engineering and Research, Maharashtra, India

Abstract:
Databases contain huge amount of hidden information that can be used for intelligent decision making. Classification is one of the forms of data analysis that can be used to extract models that represents classes of data. Although k-NN is well known pattern classification algorithm used in different applications, it has some loopholes for large dataset. First limitation is, it requires whole training set to be stored in memory. Also for classifying a test pattern it has to be compared with all other training instances. So in order to improve the performance of k-NN classifier prototype generation are used. Prototype reduction techniques can be divided into two different approaches, known as prototype selection and prototype generation or abstraction. By building new artificial prototypes, prototype generation increase accuracy of NN (Nearest Neighbor) classification. EMOPG provides better results in terms of accuracy and reduction than other Prototype Generation methods.

Keywords: *Nearest Neighbor Classifier, Prototype Generation, Multi-objective Optimization, Evolutionary Algorithm*

1. Introduction

A vast number of pattern classification methods have been proposed so far, out of which the k-nearest neighbor (k-NN) classifier is one of the well-known.k-nearest neighbours algorithm plays a significant role in the processing time of many applications. These applications covers various fields such as pattern recognition, data mining and machine learning, text mining k-NN classifier belongs to lazy learning family, meaning that no training phase is needed for this method. Lazy learners classify the sample on the basis of the labels assigned to their closest training samples. In this sense the standard k-NN requires an entire training set to be stored in memory and it performs as many distance computations as samples available in the training set for classifying a single test pattern.
However k-NN classifier has two main drawbacks in case of large dataset as

1. The large training set has to be stored in memory
2. For classifying a new instance it has to be compared to all other training instances

So to overcome these limitations of k-NN classifier the prototype generation methods can be introduced. These prototype generation methods can used to reduce the size of training set for maintaining or increasing the performance of classification.
In prototype generation, we build new artificial prototypes to increase accuracy of classification. Prototype Generation method is treated as one of optimization, where the goal isto find (to optimize) the best set of prototypes for pattern classification with *k*-NN. This technical paper will provide the survey about the different prototype generation methods.

2. Survey of Prototype Generation

Formal specification of prototype generation problem is:
x_p is an instance, where $x_p = (x_{p1}, x_{p2},\ldots, x_{pm}, x_{pw})$, with x_p belonging to a class w as given by x_{pw} and a m-dimensional space in which x_{pi} is the value of i-th feature of the p-th sample, then as assumed that there is a training set TR consisting of n instances x_p and also a test set TS which is composed of s instances $x_{t,}$ with x_{tw}) unknown. Prototype generation obtains a prototype generate set TG. TG consists of r, $r < n$, prototypes, they may be selected or generated from the examples of TR. Prototypes of the generated set are

determined to represent efficiently the distributions of the classes and to discriminate well when used to classify the training objects. Cardinality of this should be sufficiently small so that to reduce both the storage and evaluation time spent by a k-NN classifier.

These different approaches for prototype generation:

2.1. Particle Swarm Optimization (PSO):
This method defines the particles of the swarm, as sets of a fixed number of prototypes, which are modified as the particle is moved in the search space.PSO tries to minimize the classification error in the training set. The method is run several times in order to obtain varied solutions (set of prototypes). When classifying a new object the outputs of all the sets of prototypes are combined via voting that is an instance is assigned to the most voted class.

2.2. Adaptive Michigan Particle Swarm Optimization (AMPSO):
Adaptive Michigan PSO (AMPSO), another variant of PSO has been used for prototype generation. Its particles encode a prototype, each one being the generated train data represented as the whole particle swarm. The PSO rules optimize the positioning. Depending on the neighborhood based the prototypes are positioned by movement equations. In AMPSO each particle of the swarm is associated to a prototype in a manner that the whole population is the set of prototype that are optimized

2.3. Prototype Selection Clonal Selection Algorithm (PSCSA):
This is based on an artificial immune system, it uses the clonal selection algorithm to find the most appropriate position for a prototype set. This algorithm is initialized by representing the training instances as antigens and choosing one antigen from each class to fill the immune memory. It consists of phases such as Proliferation I (Hyper-mutation), Resource Allocation, While resources left to do:

i) Proliferation II (Mutation).
ii) Resource Allocation.
And at last insert best antigen into immune memory.

2.4. Prototype nearest Neighbor (PNN):
The algorithm of finding Prototypes for Nearest Neighbor classifiers (PNN) can be stated as follows: given a training set TR, the algorithm starts with every point in TR as a prototype. Initially, set A is empty and set B is equal to TR. The algorithm selects an arbitrary point in B and initially assigns it to A. After this, the two closest prototypes x_p in A and x_q in B of the same class are merged, successively, into a new prototype, x_p, if the merging will not degrade the classification of the patterns in TR, where x_p is the weighted average of x_p and x_q. Initially, every prototype has an associated weight of unity.

3. Proposed System

3.1. Evolutionary Multi-objective Approach for Prototype Generation (EMOPG):
This is an evolutionary multi-objective approach for the prototype generation. This aims to adjust the positioning of prototypes in the input feature space by using PAES (Pareto Archived Evolution Strategy). A solution is initialized with a set of potentially good instances. Then, PAES optimizes the positioning of these instances and aims to explicitly maximize classification and reduction performance of k-NN classifier. PAES is proposed for generating a single solution from the set of non-dominated solutions obtained. This proposed method achieves the best trade-off in terms of reduction and accuracy.
We are generating the prototypes by weighting the instances in the training set based on the Euclidean distance formula; the initial set of prototype will consist of higher weighted instances for the evolutionary algorithm. The second step of our proposal is to generate an initial set of prototypes through the selection of samples from the training set. After construction of initial set of prototypes, these prototypes are positioned in the feature space by PAES algorithm. After proper positioning of prototypes we are selecting a single solution from the non-dominate set of solution. And finally this set of prototypes to be used as a reduced data set for the 1NN classifier. We will be extending EMOPG for dealing with the reduction of both the number of samples and the number of features. We will be evaluating EMOPG for different values of *k* of *k*-NN, also like to study the impact of the evolutionary parameters on the quality of the prototypes found by EMOPG. Finally, we will test EMOPG on large scale data sets.

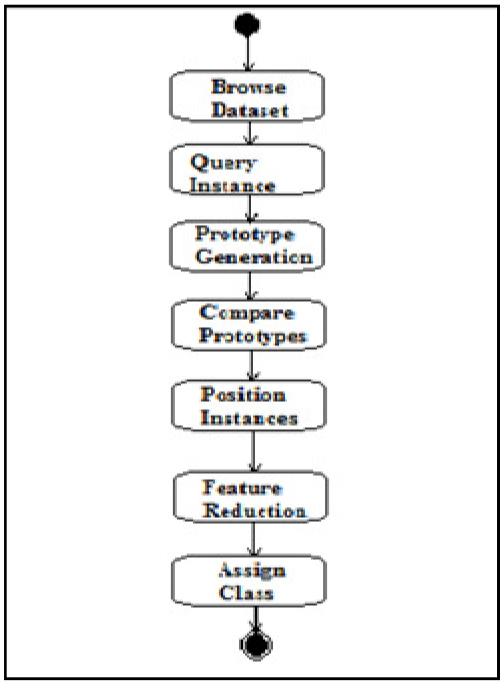

Figure 1: Flow Diagram

4. Conclusion

Our approach explicitly aims to optimize the two main objectives that are directly concerned with the two main drawbacks of the *k*-NN classifier that affect its use on large data sets. EMOPG obtains prototypes which do not degrade the performance of *k*- NN. This makes the approach applicable to problems from different domains, especially on large data sets. Our algorithm outperformed over several PG methods and it was not significantly worse compared to the best ones. With reference to EMOPG we are working for dealing with the reduction of both the number of samples and the number of features. Including preferences during the optimization is also another interesting path for future research.

5. References

i. Alejandro Rosales-P´erez, Hugo Jair Escalante, Carlos A. Coello Coello, Jesus A. Gonzalez, and Carlos A. Reyes-Garcia, "An Evolutionary Multi-Objective Approach for Prototype Generation" ,IEEE Congress on Evolutionary Computation (CEC),July 6-11, 2014.

ii. X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," Knowl. Inf. Sys., vol. 14, no. 1, pp. 1–37, 2007.

iii. I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearestneighbor classification," IEEE Trans. Syst. Man Cy. C, vol. 42, no. 1, pp. 86–100, Jan. 2012.

iv. L. Nanni and A. Lumini, "Particle swarm optimization for prototype reduction," Neurocomputing, vol. 72, o. 4–6, pp. 1092–1097, 2008.

v. U. Garain, "Prototype reduction using an artificial immune system," Pattern Anal. Appl., vol. 11, no. 3–4, p. 353–363, 2008.

vi. A. Cervantes, I. M. Galvan, and P. Isasi, "AMPSO: a new particle swarm method for nearest neighborhood classification," IEEE Trans. Sys. Man Cy. B, vol. 39, no. 5, pp. 1082–1091, 2009