A Genetic Algorithms Approach to Feature Subset Selection for Pattern Classification Using Neural Networks

Hasan Doğu TAŞKIRAN
Computer Science Department, Bilkent University
Ankara, Turkey

E-mail: taskiran@cs.bilkent.edu.tr
Phone: +90 533 513 14 26

Abstract

In this paper, we present a genetic algorithms based approach to the selection of a subset of features, which are used to classify the patterns using neural networks. It is important to remember that adding a new feature unnecessarily increases the complexity and the cost of training and classification, so we need to be able to differentiate between those features that contribute new information and not. Many current feature reduction techniques such as PCA and LDA involve linear transformations of the original pattern vectors to new vectors of lower dimensions. So a multi-objective genetic algorithm has been employed to reduce the cost and increase the accuracy of classification. We also show the application of our work on a handwritten digit recognition system. Comprehensive experiments show that the proposed approach is actually effective.

Keywords: Feature Subset Selection, Genetic Algorithms, Neural Networks, Pattern Classification

1. Introduction

In practical pattern recognition problems, a classification function learned through an inductive learning algorithm assigns a given input pattern to one of the existing classes of the system. Usually, the representation of each input pattern consists of features since they can distinguish one class of patterns from another in a more concise and meaningful way than offered by the raw representation. In many applications, it is not unusual to find problems involving hundreds features. However, it has been observed that, beyond a certain point, the inclusion of additional features leads to a

worse rather than better performance. Moreover, the choice of features to represent the patterns affects several aspects of the pattern recognition problem such as accuracy, required learning time and necessary number of samples.

In pattern recognition problems, supervised learning is a very common technique used to classify the patterns in the given dataset. Many of the classification problems can be solved using artificial neural networks (ANNs), which have been shown good classifiers. Given the features, ANNs classify a pattern into one of the predefined classes. For an ANN to be able to classify the patterns the ANN should be trained with some number of patterns and should be taught their correct classes.

However, for a given pattern classification problem, the network may become unbelievably complex if the number of the features used to classify the pattern increases very much. This paper is mainly concerned with reducing the number of features used for classification by ANNs while not allowing the accuracy of the classifier to decrease. This means that we need to decrease the cost of training classification while increasing the accuracy.

The problem of feature subset selection has been the study of a diverse spectrum of fields. In neural network pattern classification, feature selection can be effected using node pruning techniques. After training for a number of epochs, nodes are removed from the network in such a manner that the increase in squared error is minimized. When an input node is pruned, the feature associated with that node is no longer considered by the classifier.

2. Neural Networks

In real-world problems such as medical diagnosis, handwritten letter or digit recognition, stock exchange analyses etc. these numbers may really become very huge and very hard to deal with these ANNs. It is because it increases the size of the ANN and so the time needed to train the network.

For example, in the case of handwritten digit recognition, an image database may be used to train a classifier. Let's say that we are provided with some number of *20 x 20* images of each digit to train an ANN to classify the given images into the bags for the digits *0-9*. If we use the pixels as features in the given images, we now have $f = 400$ features to be analyzed for classification. This means that we will have to have an ANN with 400 input neurons and 10 output neurons for the digits 0-9 that this pattern may belong to. The output neurons will show the probability of the input pattern being the digit that neuron represents. If you also define some number of hidden layers, then there are so many connections between the neurons to deal with. Let's say that we have just one hidden layer with 20 neurons then the number of connections in a fully connected network becomes $(400 \times 20) + (20 \times 10) = 8200$. This means that we somehow need to update the weights of those connections according to the given training patterns.

Experiments show that for an ANN to classify the patterns with high accuracy, it should be that $n/f > 10$ where n is the number of patterns used to train the network, and f is the number of features that are used for classification. For our example for the handwritten digit recognition system, if $f = 400$ then we should have more than or equal to 4000 training samples to be able to make a classification with an acceptable error rate.

3. Feature Subset Selection

The term feature subset selection is applied to the task of selecting those features that are most useful to a particular classification problem from all those available. The main purpose of feature subset selection is to reduce the number of features used in classification while maintaining acceptable classification accuracy. Less discriminatory features are eliminated, leaving a subset of the original features which retains sufficient information to discriminate well among classes.

For classical pattern recognition techniques, the patterns are generally represented as a vector of feature values. The selection of features can have a considerable impact on the effectiveness of the resulting classification algorithm. Consider a feature set, $F = \{f_0; f_1; \ldots; f_N\}$. If f_0 and f_1 are dependent, that is they always move together, then one of these could be discarded and the classifier has no less information to work with. This has the benefit that computational complexity is reduced as there is smaller number of inputs. Often, a secondary benefit found is that the accuracy of the classifier increases. This implies that the removed features were not adding any useful information but they were also actively hindering the recognition process.

The problem of feature selection can be seen as a case of feature weighting, where the numerical weights for each of the features have been replaced by binary values. A value of 1 could mean the inclusion of the corresponding feature into the subset, while a value of 0 could mean its absence.

4. Genetic Algorithms

Genetic Algorithms (GAs) are a family of computational models inspired by evolution. Computational studies of Darwinian evolution and natural selection have led to numerous models for computer optimization. GAs comprise a subset of these evolution-based optimization techniques focusing on the application of selection, mutation, and recombination to a population of competing problem solutions. GAs are parallel iterative optimizers, and have been successfully applied to a broad spectrum of optimization problems, including many pattern recognition and classification tasks.

Being a directed search rather than an exhaustive search, population members cluster near good solutions; however, the GA's stochastic component does not rule out wildly different solutions, which may turn out to be better. This has the benefit that, given enough time and a well bounded problem, the algorithm can and a global optimum. This makes

them well suited to feature selection problems {they can and near optimum solutions using little or no a priori knowledge.

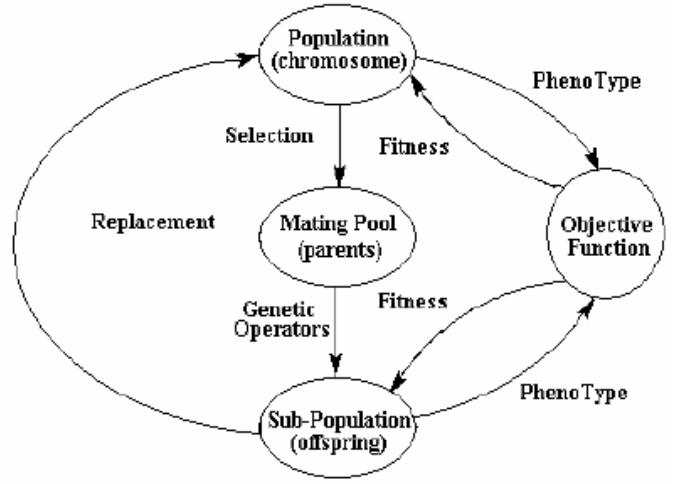

Figure 1-Simple Genetic Algorithm Flow

There are three major design decisions to consider when implementing a GA to solve a particular problem. A representation for candidate solutions must be chosen and encoded on the GA chromosome, an objective (fitness) function must be specified to evaluate the quality of each candidate solution, and finally the GA run parameters must be specified, including which genetic operators to use, such as crossover, mutation, selection, and their possibilities of occurrence.

The process of fitness-dependent selection and application of genetic operators to generate successive generations of individuals is repeated many times until a satisfactory solution is found. In practice, the performance of genetic algorithm depends on a number of factors including: the choice of genetic representation and operators, the fitness function, the details of the fitness-dependent selection procedure, and the various user-determined parameters such as population size, probability of application of different genetic operators, etc.

5. Methodology

In this section we present our application of genetic algorithms to the process of feature subset selection where they will be used to train an ANNs. We will also show our results on a handwritten digit recognition problem.

As the problem of feature selection can be seen as a case of feature weighting, we will represent our feature subsets as binary strings where a value of 1 will represent the inclusion of a particular feature in the training process and a value of 1 will represent its absence.

Since we are representing a chromosome through a binary string, our genetic algorithm will operate on a pool of binary strings. The mutation and crossover operators operate in the following way: Mutation operates on a single string and generally changes a bit at random. Thus, a string 10010 may, as a consequence of random mutation gets changed to 10110. Crossover on two parent strings produces two offsprings. With a randomly chosen crossover position 2, the two strings 01101 and 11000 yield the offspring 01000 and 11100 as a result of crossover.

With given training dataset we will create an ANN to be able to evaluate the fitness of the resulting binary set which represents the feature subset to our problem. For each binary string we train a new network with the selected features as input nodes. So the network will have the same number of input nodes as the number of 1s in the binary string.

After training the network we then test it with some number of test inputs. As a result of the training we obtain an error value e(x) for classification where x represents the binary string and $0 \leq e(x) \leq 1$ and also we have to consider the cost of the training needed to obtain this error value.

If we assume that the cost of using full feature set as 1 and the cost of training the network is linear in the number of its number of input neurons then we can find the cost, s(x), of the binary string by dividing the number of 1s in the string to the number of features in the full feature set. So s(x) becomes $0 \leq s(x) \leq 1$ also. An analysis of the s(x) and e(x) values then gives us the feature subset fitness function as

$$f(x) = \big(2 - e(x)\big) - \left(\frac{s(x)}{2 - e(x)}\right).$$

This is a multi-objective optimization function which both tries to reduce the error and the cost of the network. This may seem an ad-hoc function at first but it does discourage trivial solutions, i.e. a zero cost solution with low accuracy, from being selected as the fittest over reasonable solutions which give high accuracy at a moderate and acceptable cost.

6. Experiments and Results

We implemented our methodology using the Matlab Neural Network Toolbox and Genetic algorithm toolbox. Although it takes very much time to evaluate fitness values for each binary string, it was respectively easy to implement those issues in Matlab.

The database we used in our experiments was the UCI database for handwritten digits. This database includes 200 samples for each digit. So there are totally 2000 digits. We have randomly chosen 100 digits from each digit for the training set and used the remaining 100 digits for testing our networks to obtain the necessary e(x) values. The digits are represented as 15 x 16 images each. We decided to use the pixels as our features and so we have 240 features to evaluate. So we will have a huge network if we use to decide to use the full feature set. And we will need to have more than 2400 training samples to obtain good results.

As we decided to reduce features we will not need so much. So we create a pool of feature subsets represented as 240-bit bit-strings where 1s represent the inclusion of the associated pixel value or the absence of it if it is 0 while training our network.

Generation	f(x)	Generation	f(x)
1	*-1.690*	51	*-1.802*
2	*-1.697*	52	*-1.799*
3	*-1.713*	53	*-1.803*
4	*-1.709*	54	*-1.800*
5	*-1.705*	55	*-1.804*
6	*-1.719*	56	*-1.802*
7	*-1.735*	57	*-1.805*
8	*-1.731*	58	*-1.798*
9	*-1.747*	59	*-1.811*
10	*-1.731*	60	*-1.797*
11	*-1.752*	61	*-1.803*
12	*-1.750*	62	*-1.799*
13	*-1.759*	63	*-1.803*
14	*-1.755*	64	*-1.803*
15	*-1.757*	65	*-1.804*
16	*-1.762*	66	*-1.808*
17	*-1.759*	67	*-1.809*
18	*-1.775*	68	*-1.797*
19	*-1.770*	69	*-1.814*
20	*-1.771*	70	*-1.807*
21	*-1.771*	71	*-1.805*
22	*-1.775*	72	*-1.803*
23	*-1.779*	73	*-1.812*
24	*-1.781*	74	*-1.806*
25	*-1.776*	75	*-1.808*
26	*-1.793*	76	*-1.813*
27	*-1.796*	77	*-1.814*
28	*-1.796*	78	*-1.802*
29	*-1.788*	79	*-1.815*
30	*-1.802*	80	*-1.807*
31	*-1.789*	81	*-1.805*
32	*-1.793*	82	*-1.808*
33	*-1.786*	83	*-1.817*
34	*-1.799*	84	*-1.810*
35	*-1.807*	85	*-1.805*
36	*-1.791*	86	*-1.807*
37	*-1.792*	87	*-1.804*
38	*-1.794*	88	*-1.808*
39	*-1.790*	89	*-1.805*
40	*-1.792*	90	*-1.807*
41	*-1.794*	91	*-1.810*
42	*-1.810*	92	*-1.812*
43	*-1.795*	93	*-1.807*
44	*-1.802*	94	*-1.811*
45	*-1.795*	95	*-1.810*
46	*-1.802*	96	*-1.805*
47	*-1.796*	97	*-1.808*
48	*-1.796*	98	*-1.819*
49	*-1.809*	99	*-1.814*
50	*-1.794*	100	*-1.816*

Table 1 - Best Fitness Values from the Generations of GA

Errors	Training Dataset	Test Dataset
Full Feature Set (240, s(x) = 1.00)	3 / 1000	101 / 1000
Optimal Subset (53, s(x) = 0.221)	6 / 1000	96 / 1000

Table 2- Classification Errors Obtained from the Datasets

Accuracy	Training Dataset	Test Dataset
Full Feature Set (240, s(x) = 1.00)	99.7%	89.9%
Optimal Subset (53, s(x) = 0.221)	99.4%	90.4%

Table 3 - Accuracies of the Classifiers for the Datasets

For each generation of GA we decided to have a pool of 50 bit-strings to operate on. For each binary string in the pool we create a new Feed-Forward back-propagation ANN with one hidden layer composed of 10 neurons. We decided on 10 because experiments show that adding more neurons inside the hidden layer does not increase the accuracy as expected for this particular problem. We used logarithmic sigmoid functions a gradient descent with momentum and adaptive learning rate back-propagation training function, namely 'traingdx'. This is the slowest training function for the ANNs in the Matlab Neural Network Toolbox; however it gives comprising results for pattern recognition problems.

We ran our genetic algorithms code for 100 generations and we have chosen to move on the two best binary-strings from our evaluation to the next generation. Best Fitness Values from our Generations are shown in Table 1. Also the mutation type is uniform where we could use a Gaussian based

mutation strategy but we wanted to keep the things simpler. We also used a rank based strategy for selection of the bit-strings for crossover. The parameters for the Genetic Algorithm for our task are:

- Population Size: 50
- Number of Generations: 100
- Probability of Crossover: 0.6
- Probability of Mutation: 0.001
- Elite Count: 2
- Type of Mutation: Uniform
- Type of Selection: Rank-based
- Stall Generations Limit: 10
- Stall Time Limit: Infinite

As the results in Tables 2 and 3 show that we obtained 53 features selected by our GA approach. This means we reduced the cost to $s(x) = 53 / 240 = 0.221$ from 1. That means we obtained an improvement on the training and classification by a factor of 4.525.

7. Conclusion

The results presented in this paper indicate that genetic algorithms offer an attractive approach to solving the features subset selection problem (under a different cost and performance constraints) in inductive learning of pattern classifiers in general, and neural network pattern classifiers in particular. This methodology finds application areas in cost sensitive design of classifiers for tasks such as medical diagnosis and computer vision. Other applications of interest include automated data mining and knowledge discovery from datasets with an abundance of irrelevant or redundant features. In such cases, identifying a relevant subset that adequately captures the regularities in the data can be particularly useful. The GA-based approach to feature subset selection does not rely on monotonicity assumptions that are used in traditional approaches to feature subset selection which often limits their applicability to real-world classification and knowledge acquisition tasks.

In this paper we presented a Genetic Algorithms approach to feature subset selection, where these features will be used to train and classify patterns using Artificial Neural Networks.

We have demonstrated that the proposed methodology succeed in reducing the complexity of the feature set used by the classifier and also that such a classifier even using less features achieved recognition rates at the same level than reached by the original classifier.

An analysis is still needed to improve the results obtained using GAs. Performance improvement and trials for the other datasets may be included in this perspective. Another analysis could be based on the fitness evaluation function where there may be used other fitness functions to be used in this approach to be able to find better results.

References

[1] Y. Le Cun, Comparison of learning algorithms for handwritten digit recognition, in International Conference on Artificial Neural Networks, Paris (F. Fogelman and P. Gallinari, eds.), pp. 53{60),1995.

[2] Oliveira L.S., Sabourin R., Bortolozzi F., and Suen C.Y. *A Methodology for Feature Selection using Multi-Objective Genetic Algorithms for Handwritten Digit String Recognition,* International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), 17(6):903-930, 2003.

[3] Oliveira L.S., Benahmed N., Sabourin R., Bortolozzi F. and Suen C.Y., *Feature Subset Selection Using Genetic Algorithms for Handwritten Digit Recognition*, 14th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2001), pages 362-369, Florianópolis, Brazil, IEEE CS Press, October 15-18, 2001.

[4] Jihoon Yang, Vasant Honavar, *Feature Subset Selection Using a Genetic Algorithm*, Genetic Programming 1997: Proceedings of the Second Annual Conference,13-16,1997.

[5] Brill, F., Brown, D., and Martin, W. *Fast genetic selection of features for neural network classifiers*. IEEE Transactions on Neural Networks, 3(2):324-328., 1992.

[6] Jarmulak, J., and Craw, S. *Genetic algorithms for feature selection and weighting.* In Proceedings of the IJCAI'99 workshop on Automating the Construction of Case Based Reasoners. Cambridge, England, 1999.

[7] Kira, K. and Rendell, L.. *A practical approach to feature selection.* In Proceedings of the Ninth International Conference on Machine Learning, pages 249-256, 1992.

[8] Almuallim, H. and Dietterich, T. *Learning boolean concepts in the presence of many irrelevant features.* Artificial Intelligence, 69(1-2):279-305, 1992

[9] John, G., Kohavi, R. & Pfleger, K. *Irrelevant features and the subset selection problem*, in W. W. Cohen & H. Hirsh (eds), Machine Learning: Proceedings of the 11th International Conference, Morgan Kaufmann, San Francisco, CA., pp. 121-129, 1994.

[10] Narendra, P. and Fukunaga, K. *A branch and bound algorithm for feature subset selection.* IEEE Transactions on Computers, 26:917-922, 1977.

[11] Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, *Handwritten digit recognition with a back-propagation network*, in Advances in Neural Information Processing Systems (D. Touretzky, ed.), vol. 2, (Denver 1989), pp. 396{404, Morgan Kaufmann, San Mateo,1990.