A Two-phase Feature Selection Method using both Filter and Wrapper

Huang Yuan* **Shian-Shyong Tseng****
Wu Gangshan* **Zhang Fuyan***
***State Key Laboratory for Novell Software Technology**
Department of Computer Science and Technology, Nanjing Univ.
Nanjing, Jiangsu 210093, China
hy@graphics.nju.edu.cn,{gswu|fyzhang}@netra.nju.edu.cn
****Department of Computer and Information Science**
National Chiao Tung Univ., Hsinchu,Taiwan
sstseng@cis.nctu.edu.tw

ABSRACT

Feature selection is an integral step of data mining process to find an optimal subset of features. After examine the problems with both the filter and wrapper approach to feature selection, we propose a two-phase feature selection algorithm of filter and wrapper that can take advantage of both approaches. It begins by running GFSIC(filter approach) to remove irrelevant features, then it runs SBFCV(wrapper approach) to remove redundant or useless features. Analysis and experimental studies show that the effectiveness and scalability of the proposed algorithm. The generalization of neural network is improved when the algorithm is used to preprocess the training data by eliminating the irrelevant and useless features from neural network's consideration.

Keyword: feature selection, feedforward neural network, filter, wrapper, genetic algorithms

1. INTRODUCTION

An abundance of unnecessary features can increase the size of search space and the time needed for inductive algorithms. On the other side, most practical inductive algorithms in data mining generalize worse given too many attributes than if given a good subset of those attributes. For example, if neural network has fewer input neurons than necessary, the inductive algorithm will fail to find the desired classification function. If neural network has far more input neurons than necessary, it can result in overfitting of the data leading to poor generalization, wasting resources by measuring irrelevant variables and a model which is difficult to understand. In a classification task, features can be redundant or irrelevant. Irrelevant features does not affect the underlying structure of the data in any way while redundant features does not provide anything new in describing the underlying structure. Feature selection is a process to select an optimal subset of features from a large set of mutually redundant, possibly irrelevant original features. After feature selection, induction algorithm can run on data only containing features relevant for classification with maximal accuracy. The generalization of induction algorithm is improved, resources are saved and the resulting architecture is easier to interpret. Many publications have reported performance improvements when feature selection algorithms are used [1,2,3].

2. PREVIOUS APPROACHES AND THEIR PROBLEMS

In general, two categories of algorithms have been proposed to solve feature selection problem. The difference of these algorithms is whether or not the feature selection is done independently of the induction algorithm. The first category is filter approach that is independent of an induction algorithm and serves as a filter to sieve the irrelevant features. The second category is wrapper approach that uses the induction algorithm itself as part of the function evaluating feature subsets. The shortcoming of filter approach is that it totally ignores the biases of the induction algorithms and the effect of selected feature subset on the performance of the induction algorithm [4]. So it can not efficiently remove the redundant features or features

0-7803-5731-0/99$10.00 ©1999 IEEE

useless or even harmful for generalization. In most cases, the optimal selection of features may not be independent of the inductive and representational biases of the inductive algorithm. The major drawback of wrapper approach is time consuming and costly especially for computationally intensive induction algorithms such as neural networks. If the size of dataset and the minimal feature set is even moderately large, wrapper will take a long time. This paper explores a two-phase algorithm mixing both filter and wrapper approaches for neural network feature selection to reduce time complexity and improve classification accuracy.

3. TWO PHASE FEATURE SELECTION METHOD

3.1 The GFSIC algorithm

The first phase GFSIC (Genetic Feature Selection with Inconsistency Criterion) uses genetic algorithm to search optimal subset of features with low inconsistency. There are several search algorithms used to search the space of feature subsets. The simplest search algorithm, called gradient-descent search, starts at an arbitrary point in the fitness landscape, and attempts to make small changes that improve the solution, but doesn't necessarily find the optimal solution. If the landscape is complicated, it may end up in a local minimal. An alternative is stochastic gradient-descent, which makes some small random changes in search process. Initially the technique can escape from local minimal and find globally good areas of the fitness landscape. It then gradually settles into a good solution. Genetic algorithms are based on an analogy with biology in which a group solution evolves via natural selection [5]. They sometimes also behave like a stochastic gradient-descent algorithm, although they have the distinction of exploring a large number of possible solutions simultaneously. Genetic algorithms also can't always find the global optimum, but they would be more robust than gradient-descent algorithms when there are strong interdependencies among features. They make relatively few assumptions about the shape of the search space, and are generally quite effective for rapid global search of large search spaces in optimization problems. Genetic algorithms have demonstrated substantial improvement over a variety of random and local search methods [6].

We use a binary string to represent the presence or absence of each possible feature. Each individual chromosome in the population represents a candidate solution to the feature subset selection problem. Let N be the total number of features available to be chosen to represent the patterns (Note that there exist 2^N possible feature subsets. Thus, exhaustive search is impractical unless N is very small). The chromosome is represented by a binary string of n bits (where n is the total number of features). If a bit is a 1, it means that the corresponding feature is selected. A value of 0 indicates that the corresponding feature is not selected. The fitness of an individual is determined by evaluating the inconsistency of a training set whose pattern are represented using only the selected subset of features. If an individual chromosome has m bits turned on, the corresponding feature set has m input features. The individual chromosomes in the population are then evaluated via a fitness function, and then the less fit individuals are eliminated. Combining two different criteria – the inconsistency of the selected feature subset and the cost of perform classification, we define the fitness function as follows to find reasonable solutions that yield low inconsistency at a moderate size of feature subset:

$$fitness(x) = consistency(x) - \frac{\lambda \cdot cost(x)}{(consistency(x)+1)\cdot cost_{\max}} \quad (1)$$

where fitness (x) is the fitness of the feature subset represented by chromosome x, and consistency(x) is the consistency rate of the selected feature subset. In our experiment, to keep things simple, cost(x) is simply represented by the number of selected features. $cost_{\max}$ is the total number of features under consideration. A nonnegative complexity penalty factor λ is added to the evaluation function, penalizing feature subsets with many features. λ also represents the crossing point of feature subsets where SBFCV takes over from GFSIC. In our experiment, we set $\lambda = 0.08$. If there are still too many features found after the first phase, λ can be adjusted optimally for the specific datasets. But if λ is too high, the small sized subsets generated by GFSIC might not contain any minimal size subset.

The inconsistency criterion suggests using feature good for discrimination with compact descriptions and maximally distinct [1]. That is, feature selection is formalized as finding the smallest set of features that is "consistent" in describing class with the full set. Given a set of training example X and a set of features Q , let c_1 and c_2 denote two class labels, for a pair of

examples $< X_1; c_1 >$ and $< X_2; c_2 >$, an inconsistency is generated if X_1 and X_2 have the same values for all the features in Q. The inconsistency rate of a dataset can be calculated as follows: (1) two instances are considered inconsistent if they match except for their class labels: (2) for all the matched instances, the inconsistency count is the number of the instances minus the largest number of instances of class labels. (3) The inconsistency rate is the sum of all the inconsistency counts divided by the total number of instance [7]. To describe inconsistency criterion simply, we use consistency rate in fitness function. The consistency rate is defined as follows:

$$consistency(x) = 1 - inconsistency(x) \qquad (2)$$

Algorithm: GFSIC(Sample datasets)
1. Create the initial pool with random population of feature subsets
2. Repeat until stopCriteria()
2.1. Apply genetic operators such as selection, crossover and mutation generates new population of feature subsets.
2.2. for every chromosome in pool
Evaluate the chromosome according to two criteria: the inconsistency and the cost of the selected feature subset.
2.3. Rank the population in pool by fitness function
end;
3. S = bestOf(pool);
4. Output the selected feature subset S to next phase.

Figure 1: A framework of GFSIC

Figure 1 describes the framework of GFSIC, which starts from the pool of random feature subset candidates. An initial population is generated at random to be the basis of the next generation. Good feature subsets are more likely to be chosen than bad ones. Applying standard genetic operators such as selection, crossover and mutation generates new pool of feature subset. The crossover operator works by taking two chromosomes and combing them somehow to produce one or two offspring. This is done by randomly selecting a point in the coded chromosome and then appending the part of the second chromosome after that point to chromosome one up to that point and vice versa. The mutation operator is only applied to one chromosome at a time and involves the random variation of one particular feature subset. This adds a limited random element into the search and may reintroduce potentially useful material that has been lost earlier in the search. Then we rank feature subsets according to the consistency and cost criterion. The process of creating new generations can be terminated when a predefined number of generations is achieved or when the overall fitness value of the population is not increased during the last generations. After the last generation of GA, the feature subsets of highest ranked individual are extracted for the next phase.

Although the first phase dramatically reduces the feature number and the search complexity, there are maybe still many redundant features after this phase. Practical induction algorithms that generate classifiers may benefit from the omission of these features, including some strongly relevant features. Relevance of a feature does not imply that it must be in the optimal feature subset [3].

Algorithm: SBFCV(Sample datasets)
1. let S be the feature subset after the first phase. Let N is the number of features in S. Divide the data set into training set D_1 *and cross-validation test set* D_2.
2. Train network to minimize the classification error function of D_1 *with the feature subset S as input nodes, let* R *be the classification accuracy of* D_2.
3. for (i=1, i < N, i++)
replacement the value of input node S_i *by its average value, calculate sensitivity measure* E_i *with dataset* D_1.
4. According to E_i *rank all input nodes. Let* S_k *be the head of queue, delete the input node* S_k *and get the network* N_k.
5. Retrain the Network N_k, *let* R_k *be the classification accuracy of* D_2 *with network* N_k.
6. If $R - R_k \le \delta$ *, then {*
$S := S - \{S_k\}$, $N := N-1$, $R := \max\{R, R_k\}$
go to step 3
}
7. output selected feature subset S.

Figure 2: A framework of SBFCV

3.2 The SBFCV algorithm

The second phase SBFCV (Sensitivity-based Feature selection with v-fold Cross Validation) starts with a feedforward neural network whose input nodes are features of optimal feature subset in the first phase. Key steps in SBFCV are shown in Figure 2. The networks used consist of 3 layers trained using back-propagation. We use sensitivity of the network to estimate the relationship of input features with network performance [8]. The

Table 1: Results of runs of GFSIC and SBFCV on the datasets with one example of the selected feature subset

Dataset	Size	#Att	#Selected Att after GFSIC	#Selected Att after SBFCV	Selected Features
Monk1	432	6	3	3	A1,A2,A5
Monk3	432	6	3	3	A2,A4,A5
DNA	106	57	29	26	-
Soybean-large	683	35	21	19	-
House votes	435	16	11	8	A1-A4,A9,A11,A15,A16

Table 2: Comparison of neural network classification accuracy using all original features against those using the optimal feature subset selected by two-phase algorithm

	All Attributes		After two-phase Feature Selection	
Dataset	Features	Classification Accuracy	Features	Classification Accuracy
Monk1	6	99.45	3	100.00
Monk3	6	93.10	3	96.22
DNA	57	86.25	27	98.43
Soybean-large	35	82.21	19	91.85
House votes	16	95.72	8	97.15
Average	24	91.35	12	96.73

sensitivity of the network to feature S_i is defined as:

$$E_i = \frac{1}{N}\sum_j E_{ij} \quad E_{ij} = SE(\bar{S}_i, \omega_\lambda) - SE(S_{ij}, \omega_\lambda) \text{ with } \bar{S}_i = \frac{1}{N}\sum_{j=1}^{N} S_{ij} \quad (3).$$

Sensitivity measure assumes that replacement of a variable by its average value removes its influence on the network output. So if we want to delete feature S_i, the influence to classification accurate is estimated by computing E_i which replaces the value of feature S_i by its average $\bar{S}_i$ for all training exemplars. It is not necessary retrain the network in evaluating E_i. Network removes the least relevant features one at a time. Then we use neural network performance on the cross-validation dataset, as the criterion to determine whether the input node should be excluded from the network. V-fold cross validation is used to check performance of the resulting neural networks on an independent test set that was not used during the search. In our experiment, we use 5-fold cross-validation. It means the resampling method removes 20% of the available data for testing a network generated with the remaining 80%. With the difference in performance between two networks with different sets of input features, SBFCV also decides whether to continue or to stop removing more features. SBFCV stops if the performance of network drops below a given threshold δ by removal the least relevant feature. We assume that the performance on a classification task is measured by the classification accuracy on the unseen test set. On a regression task, the performance can be measured by the mean squared error of the test set:

$$E = \frac{1}{2m}\sum_{\mu=1}^{m}(T^\mu - O^\mu)^2 \quad (4)$$

Where T^μ and O^μ is the target and output of example μ in test set, m is the number of exemplars in test set. In our experiment, we set the threshold $\delta = 0$. Then the best feature subset is obtained by the highest cross-validation average network performances. SBFCV guarantees good generalization, but its time performance can deteriorate if the selected feature number M is not large with respect to total feature number N. It is due to every time we removing a node, the neural network should be retrained. Because GFSIC dramatically reduce the number of features during the first phase, so N-M is always small. We can see this in following experiments.

4. Experiments

In this section we report our experimental results to select features for classification problems. The classification problems reported here include real-world problems as well as artificial problems. Artificial problems include Monks1 and Monks3 where the relevant features are known before feature subset selection is conducted. Three real-world problems were also tested, including DNA Promoters Gene Sequences, Michalski's Soybean Dataset and Congressional Voting Records. All datasets were obtained via anonymous ftp from the University of California at Irvine repository [9], from which full documentation for all datasets can be obtained. Table 1 summarizes the characteristics of the data sets. In this experiment we report all averages for cross-validation classification

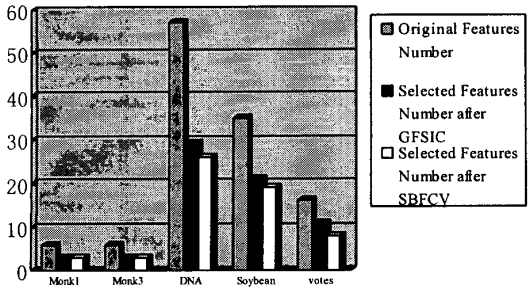

Figure 3: Number of Features in original dataset, Selected after GFSIC and selected after SBFCV.

accuracy before and after feature selection, size of selected feature subset after each phase and number of datasets evaluated.

The number of those features selected after first phase and second phase are reported in Table 1. For the Monk1 and Monk3 datasets, the relevant features are always selected. As shown in Figure 3, the numbers of selected features are small compared to the original datasets. The effectiveness the feature selection algorithm is shown by the small number of features selected. Also the numbers of features removed during the second phase are always rather small. So the time complexity of wrapper approach is dramatically reduced after the first filter phase. In some simple artificial datasets such as Monk1 and Monk3, GFSIC is enough to get the optimal feature subsets. There is not always necessary for SBFCV to remove redundant features. Table 1 also shows some samples of small selected feature subsets in the last column. As can be seen in Table 2 the performance of the networks has improved in general after our feature selection process. The average accuracy significantly increases from 91.35 to 96.73, while the average features number decrease from 24 to 12. This indicates that our algorithm has successfully removed superfluous features, which are very noisy or contain only small amounts of information from these datasets. Feature subset selection resulted in significant improvement in generalization.

5. CONCLUSION

Neural network feature selection is a fairly hard problem. Because of the time complexity of retraining the network, wrapper methods are infeasible. On the other hand, filter approach is not always enough to get good classification accuracy. In most cases, humans currently do the feature selections to neural network classifiers. Because we still do not have a good understanding of how neural networks work, it is unlikely that the optimal feature subsets available are actually selected. In this paper we have proposed a novel approach for neural network feature selection by mixing filter and wrapper approach that attempts to solve their problems. The algorithm is effective in eliminating unimportant and redundant features while reducing cross-validation error. Because most of irrelevant features are deleted after the first phase of filter approach, it avoids the exponential computation problem of wrapper approach in the second phase. Our experimental results suggest that the proposed algorithm work well on a wide variety of problems.

REFERENCES

[1] Almuallim, H., and Dietterich, T.G., "Efficient algorithms for identifying relevant features" In Proceedings of the Ninth Canadian Conference on Artificial Intelligence, Vancouver, BC: Morgan Kaufmann ,1992,pp. 38-45.

[2] Aha, D.W., and Bankert, R. L., "A comparative evaluation of sequential feature selection algorithms. " In D. Fisher & J.-H. Lenz (Eds.), Artificial Intelligence and Statistics V. New York: Springer-Verlag. 1996

[3] W Siedlecki and J. Skalansky, "On automatic feature selection," Int. J. Pattern Recog. Art. Intell. vol. 2, no.2.pp.197-220. 1988.

[4] Kohavi, R., and John, G. "Wrappers for feature subset selection. Technical Report." Computer Science Department, Stanford University. 1995

[5] D.E.Goldberg. "Genetic Algorithm as Search Optimization and Machine Learning." Reading, MA. Addison-Wesley,1989

[6] Davis L. "Handbook of Genetic Algorithms" Van Nostrand Reinhold. 1991

[7] Liu and R.Setiono. "Feature selection and classification – a probabilistic wrapper approach", in proceedings of the 9th International Conferences on Industrial and Engineering Applications of AI and ES, 1996.

[8] Moody,J.E. and Utans,J. "Principled architecture selection for neural networks.".Morgan Kaufmann Publishers, San Mateo, CA, pp.683-690.

[9] UCI Machine Learning Repository http://www.ics.uci.edu/~mlearn/MLRepository.html