Combining Feature Selection and Neural Networks for Solving Classification Problems

Paul O' Dea, Josephine Griffith, Colm O' Riordan
Information Technology Department,
National University of Ireland, Galway,
Galway

Abstract

This paper presents an approach to solving classification problems by combining feature selection and neural networks. The main idea is to use techniques from the field of information theory to select a set of important attributes that can be used to classify tuples. A neural network is trained using these attributes; the neural network is then used to classify tuples. In this paper, we discuss data mining, review common approaches and outline our algorithm. We also present preliminary results obtained against a well-known data collection.

1 Introduction

Data mining has been defined as "The nontrivial extraction of implicit, previously unknown, and potentially useful information from data"[3]. The field, also known as knowledge discovery, has parallels in the machine learning community and has recently been afforded much attention in the field of e-commerce.

The task usually involves extracting knowledge or information which is implicit in the data collection. For example, in some database or data warehouse of records, one may be interested in discovering implicit relationships between certain attributes (or sets of attributes).

Techniques used in the field of data mining range from AI based learning algorithms (borrowing ideas from statistics and information theory) to neural network based approaches.

In this paper we discuss, briefly, common problems in the field (Section 2) and well-known approaches with emphasis on ideas from information theory and neural networks (Section 3). In Section 4 we present our algorithm which is in effect a neural network approach combined with feature selection. We discuss our motivations and the design of the algorithm. Section 5 includes initial results and discussion of results obtained to date.

2 Data Mining

Data mining techniques are predominantly applied to the problem of finding association and classification rules, as well as to the problems of item-set recognition and sequential pattern recognition. Classification is discussed in considerably more detail than the others given the focus of this paper.

1. Classification Rules: Classification is the process of finding the common properties among different entities and classifying them into *classes*. The results are often expressed in the form of rules - *classification rules*. By applying the rules, entities represented by tuples can be easily classified into the different classes to which they belong.

Given a set of tuples of attributes $A_1, A_2, \ldots, A_N$, a classification rule may take the form:

$(A_i = condition_i) \wedge \ldots \wedge (A_j = condition_j) \rightarrow Class_i$

We can restate the problem formally defined by Agrawal *et al.*[1] as follows: let A be a set of attributes $(A_1, A_2, ..., A_n)$ and dom(A_i) refer to the set of possible values for attribute A_i. Let C be a set of classes $c_1, c_2, ..., c_m$. We are given a data set, *the training set* whose members are n*+1*-tuples of the form $(a_1, a_2, ..., a_n, c_k)$ where $a_i \in$ dom$(A_i), (1 \leq i \leq n)$, and $c_k \in C_i (1 \leq i \leq m)$.

For example, a retail outlet may wish to classify customers into classes so as to adopt an advertising strategy to maximise profit. So customers may be placed in disjoint classes based on age, salary, previous purchases etc.; based on these classes, different products might then be advertised differently.

2. Associations: Association rules specify associations between sets of items. Given a relation comprising attributes $A_1, A_2, \ldots A_N$, the goal is to extract association rules which involves selecting two subsets of values for attributes S_1 and S_2 such that there is some association between the sets.

3. Itemset recognition is a form of association rule where the goal is to identify sets of items that occur together with a high frequency. Many variations exist on this item-set problem: determining itemsets given the existence of a hierarchical categorisation of items, determining time-dependent sequences of item-sets. The goal with sequential patterns is to derive a pattern of events or actions in the set of tuples. One wishes to detect a form of association rules between tuples with certain temporal constraints.

2.1 Applications

Data mining techniques are applicable to a wide variety of problem areas. Currently, the main application area for data mining is e-commerce where it is used to understand and target each customer's individual needs by highlighting both customer preferences and buying patterns from databases containing customer and transaction information. With this information, companies can better target customers with products and promotional offerings.

A number of financial applications also use data mining techniques. Examples include picking stocks, detecting fraud, commerical lending decisions etc.

3 Techniques and Approaches

Many data mining techniques and approaches have been developed and used. Common approaches are outlined below:

- Decision Trees

  This technique recursively partitions the data set until each partition contains mostly examples from a particular class[4]. In a decision tree, each internal node represents a split point which tests some property where each possible value of that property corresponds to a branch of the tree, leaf nodes representing classifications.

  An object of unknown type may be classified by traversing the tree, testing the object's value for each property at an internal node and taking the appropriate branch. Eventually a leaf node will be reached which represents the object's classification.

  ID3[8] represents concepts as decision trees. The ID3 algorithm constructs decision trees from a set of examples using a top down approach. The examples are tuples where the domain of each attribute of these tuples is limited to a small number of examples, either symbolic or numerical.

  Decision trees are popular because they are easy to understand and results are reasonably accurate. The rules for classifying data are in a form readily understood by humans. However, the performance of the decision tree depends critically upon how the split point is chosen. Often splits between branches are not smooth and the choice of split

is made regardless of the effect such a partition will have on future splits. Additionally, in the presence of noise or missing attribute values in the data set, there can be problems with performance.

- Neural Networks
  Neural networks, a form of subsymbolic computation, are based (simplisitically) on the workings of the brain. A neural network comprises a set of weighted edges and nodes. Learning is achieved by modification of these weights. Most networks contain a number of layers, the first layer being the input layer, the final layer being the output layer. Other internal layers (hidden layers) are often required to ensure sufficient computational power in the network.

  A network can be trained to map input values to corresponding output values by providing a training set. The network is repeatedly tested and modified to produce the correct output.

  The generation of output by a neural network is accomplished via firing values from nodes. An input is passed to the input layer which in turn can activate the internal layers, which in turn activates the output layer, finally resulting in an output.

  Given n links feeding into a node, each link has an input value X_j and a weight W_j. The nodes have an associated threshold, τ. If, according to some activation function, the node has a sufficiently high activation level, the node fires a value onto the next layer. Commonly used activation functions include:

  $$f(a) = \begin{cases} 1 & \text{if } a > \tau \\ 0 & \text{otherwise} \end{cases}$$

  $$f(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

  where a, the activation of a node, is $\sum_{j=1}^{n} X_j W_i$ and τ is a threshold. The initial input vector is fed into the network; sets of nodes are fired which finally results in an output vector.

  (Note: the above description relates to a simple feed-forward network; other more complicated (and computationally expressive) architectures are possible).

  To train the network, any errors in the output are fed back through the network causing a modification of the weights on the nodes. Errors can be calculated at the output layer. For internal nodes, the error is a function of all nodes that use the node's output and the output from that node.

  The error at the output layer is used to re-modify the weights coming to the output layer. This allows the calculation of errors at the last hidden layer etc. The error is *back-propogated* throughout the network.

  An example of a system which uses a neural network approach is NeuroRule[5] where the number of input nodes corresponds to the dimensionality of the input tuples and the number of output nodes is equal to the number of classes to be classified. The first stage, network training, is terminated when a local minimum is reached. Secondly network pruning is carried out, and finally extraction of the classification rules from the pruned network.

  The major criticisms of a neural network approach include the fact that because neural networks learn the classification rules by multiple passes over the training data set, the learning time, or the training time needed for a neural network to obtain a high classification rate, is usually long[1]. In addition, there is difficulty in understanding the rules generated by neural networks as they are buried in the network architecture and the weights assigned to the links between the nodes. Also, there is difficulty in incorporating any available domain knowledge.

3.1 Information Theory

Information theory is widely used in computer science and telecommunications, including such applications as determining the information-carrying capacity of communications channels, developing data compression algorithms, and developing noise-resistant communication strategies. Information theory provides a mathematical basis for measuring the information content of a message[2].We may think of a message as an instance in a universe of possible messages; the act of transmitting a message is the same as selecting one of these messages. From this point of view, it is reasonable to define the information content of a message as depending upon both the size of this universe and the frequency with which each possibe message occurs.

Shannon formally defined the amount of information in a message as a function of the probability of occurrence of each possible message[9]. Given a universe of messages, $M = \{m_1, m_2, ..., ...m_n\}$ and a probability, $p(m_i)$, for the occurrence of each message, the information content of a message in M is given by:

$$I(M) = \sum_{i=1}^{n} -p(m_i)\log_2(p(m_i))$$

The information in a message is measured in bits. This definition formalises many of our intuitions about the information content of messages.

We may think of each property of an instance as contributing a certain amount of information towards its classification. The ID3 family of decision tree induction algorithms use information theory to decide on which attribute, shared by a collection of instances, to next split the data[8]. ID3 measures the information gained by making each attribute the root of the current subtree. It then picks the attribute that provides the greatest information gain. Attributes are chosen repeatedly in this way until a complete decision tree that classifies every input is obtained.

For example, consider the problem of estimating an individual's credit risk on the basis of such properties as credit history, current debt, collateral, and income [6]. Table 5 lists a sample of individuals with known credit risks. We may think of a decision tree as conveying information about the classification of examples in the decision table; the information content of the tree is computed from the probabilities of the different classifications. For example, if we assume that all the examples in Table 5 occur with equal probabilities, then

p(risk is high) = $\frac{6}{14}$
p(risk is moderate) = $\frac{3}{14}$
p(risk is low) = $\frac{5}{14}$

It follows that the information in the table, and consequently any tree that covers those examples, is

$$\begin{aligned} I(\text{Table 5}) & \\ &= -\tfrac{6}{14}\log_2\left(\tfrac{6}{14}\right) - \tfrac{3}{14}\log_2\left(\tfrac{3}{14}\right) - \tfrac{5}{14}\log_2\left(\tfrac{5}{14}\right) \\ &= 1.531 \text{ bits} \end{aligned}$$

For a given test, the information gain provided by making that test at the root of the current tree is equal to the total information in the tree minus the amount of information needed to complete the classification after performing the test. The amount of information needed to complete the tree is defined as the weighted average of the information in all its subtrees. The weighted average is computed by multiplying the information content of each subtree by the percentage of the examples present in that subtree and summing these products.

Assume a set of training instances, C. If we make property P, with n values, the root of the current tree, this will partition C into subsets, $\{C_1, C_2, ..., C_n\}$. The expected information needed to complete the tree after making P the root is:

$$E(P) = \sum_{i=1}^{n} \frac{|C_i|}{|C|} I(C_i)$$

The gain from property P is computed by subtracting the expected information to complete the tree from the total information content of the tree:

$$\text{gain}(P) = I(C) - E(P)$$

4 Algorithm

As discussed earlier, neural networks provide a powerful mechanism for classification obtaining high precision whilst being robust in the presence of noise. Unfortunately the time to train a network can be prohibitive and the classification rules can be buried in the neural network architecture.

The more traditional approaches based on information theory perform well but tend to deal poorly with noise. The main advantages associated with this approach is that the rules are in a more understandable format and the technique is far more computationally efficient.

The motivation behind the algorithm proposed in this paper is to exploit the advantages of the Neural Network approach while maintaining a degree of computational efficiency by utilising ideas present in information theory.

The algorithm comprises two phases:

1. firstly, using ideas from information theory, select *important* attributes, and
2. secondly, apply a neural network to allow classification of tuples based on the attributes selected as being *important*.

More formally, given a set of *n* tuples $t_1 \ldots t_n$ with attributes $a_1 \ldots a_n$, the information content of each of the *n* attributes is calculated according to their ability to classify the tuples $t_1 \ldots t_n$.
The attributes' ability to classify the tuples is calculated by measuring the expected information via:

$$\varepsilon_i = \sum_{k=1}^{N_i} \frac{n_{ik}}{n} \cdot I(S_{ik})$$

where n is the total number of tuples, N_i is the total number of values attribute A_i can take and $I(S_{ik})$ is the entropy of the subset S_{ik}.

This entropy is calculated via:

$$I(S_{ik}) = \sum_{c=1}^{N_c} \frac{n_{ikc}}{n_{ik}} \cdot \log_2 \frac{n_{ikc}}{n_{ik}}$$

where S_{ik} is the subset of tuples in which attribute A_i has value k, n_{ikc} is the number of tuples belonging to class c.

In the second phase, we attempt to classify the tuples using a subset of the attributes. This subset of attributes $s_1 \ldots s_n$ comprises those with an information content above a threshold. We then consider the set as a whole thereby avoiding, to a degree, the attribute independence assumption prevalent in traditional approaches.

A feed-forward network (as described earlier), using back-propagation as a learning algorithm, is used to classify the tuples $t_1 \ldots t_n$ based on the attributes $s_1 \ldots s_n$.

5 Results

5.1 The German Credit Data Set

In order to facilitate testing of the developed approach, experiments were conducted using the german credit data set [1].

The german credit data set contains information on 1000 loan applicants. Each applicant is described by a set of 20 different attributes. Of these 20 attributes, seventeen attributes are discrete while three are continuous. To facilitate feature selection and neural network training in the later phase, the values of the three continuous attributes were discretised. Each of the three attributes was discretised by dividing its range into subintervals.

A classification assigned to each of the applicants determines whether an applicant is a good or bad credit risk. Thus the problem is to classify each pattern as either good or bad. In the data set, there are a total of 700 cases of good applicants and 300 cases of bad applicants. Furthermore, the data set is divided into a training set and a test set. The training set consists of 666 tuples and the test set contains 334 tuples.

Using the feature selection algorithm outlined, of the original 20 attributes describing each pattern in the german dataset, 7 were selected. In Table 1, we list the information gains $\mathcal{G}$, normalised gains $\mathcal{G}'$ and their averages for the german credit problem. Those attributes deemed *selectable* are: *status, duration, credit history, credit amount, savings, housing* and *foreign worker*.

5.2 Learning with Feature Selection

In the second phase, the number of units in the input layer of the neural network was determined. The *thermometer* coding scheme was employed to get the binary representations of the attribute values for inputs to the neural network. Hence for attribute *status*, a value of *less-200DM* was coded as $\{001\}$, a value of *over-200DM* was coded as $\{011\}$, and a value of *no-account* was coded as $\{111\}$. Zero status (*0DM*) was coded by all zero values for the three inputs. The second attribute *duration* was similarly coded. For example, a duration value less than 20 months was coded as $\{0001\}$, a duration value in the interval [20,40] was coded as $\{0011\}$, etc. The coding scheme for the other attributes are given in Table 2.

With this coding scheme, we have a total of 25 binary inputs. Two nodes were used at the output layer. The target output of the network was $\{1,0\}$ if the tuple belonged to class *good*, and $\{0,1\}$ if the tuple belonged to class *bad*. The number of hidden nodes in the network was initially set as three. Thus, there were a total of 81 links in the network. The weights for these links were given initial values that were randomly generated in the interval $[-1, 1]$.

It is useful to look at error profiles as a function of iteration to gain insight into the convergence. Figure 1 illustrates the convergence behaviour during a typical training phase for a network constructed with only selected attributes.

The network was then trained until a local minimum point of the mean squared error function had been reached. In Figure 1, we can observe that the error curve reaches a local minimum point of the mean squared error function in the interval $[0.30, 0.32]$. Through experiment, a network constructed using only selected attributes generally tends to reach a local minimum point of the mean squared error function in the interval $[300, 600]$ epochs.

The end result of the above training phase was a fully connected trained network which achieved an accuracy of 78.53% on the training data where classification accuracy is defined as,

[1] This data set is available publicly from the University of California-Irvine machine learning repository [7] via anonymous ftp to *ics.uci.edu*

No.	Attribute	$\mathcal{G}$	$\mathcal{G}'$
1.	status	0.08166752	0.04533132
2.	duration	0.01565728	0.01175013
3.	credit history	0.03506461	0.02039325
4.	purpose	0.02510743	0.00945620
5.	credit amount	0.01835606	0.02097592
6.	savings	0.04112237	0.02461317
7.	employment duration	0.01262678	0.00581796
8.	installment rate	0.00467093	0.00258875
9.	personal status	0.00621573	0.00404868
10.	debtors	0.00481950	0.00893288
11.	residence	0.00117720	0.00064023
12.	property	0.01892740	0.00971905
13.	age	0.01454505	0.00788522
14.	installment plans	0.00604603	0.00699498
15.	housing	0.01267492	0.01136562
16.	existing credits	0.00131140	0.00119170
17.	job	0.00468588	0.00326961
18.	liable people	0.00030049	0.00049862
19.	telephone	0.00002599	0.00002691
20.	foreign worker	0.00523591	0.02339419
AVERAGE		0.01551192	0.01094472

Table 1. *Attribute gains of the german credit data set.*

$$\text{accuracy} = \frac{\text{number of tuples correctly classified}}{\text{total number of tuples}}$$

The degree of error in this result can be attributed to two characteristics of the training data. In the first case, due to the presence of noise, it is not possible to achieve 100% accuracy on training data for the german credit problem. There are always problems with real-world data. Noise in the data encompasses irrelevant, missing, incorrect, and contradictory data. In general, noise in the data weakens the predictive capability of the features. The other possible factor is due to the imbalance in the training set.

The end result of training is that we now have a network which will act as a classifier for applicants of unknown class. In general, it will be the performance of the classifier on the test set, which does not participate in the training phase, that will be the most salient. The network obtained from the training phase described above achieved a classification accuracy of 74.25% on the test data.

5.3 Description of Results

In order to test the usefulness of the proposed approach, we present empirical analysis comparing the accuracy of networks constructed with and without feature selection over the chosen test collection.

Twenty neural networks were constructed with the full set of twenty attributes and another twenty networks were constructed using only the seven selected attributes. Of the twenty networks constructed in each case, five networks had 1 hidden unit, another five had 2 hidden units etc. Table 3 and Table 4 summarise the classification accuracies achieved by these networks on both the training data and the test data of the german credit problem.

From these tables, we can observe that removing redundant attributes for the german credit problem increased slightly the predictive accuracy of a neural network. For the german credit problem, the accuracy on the training data actually decreased

Attribute	Input Number
status	$\mathcal{I}_1$ - $\mathcal{I}_3$
duration	$\mathcal{I}_4$ - $\mathcal{I}_7$
credit history	$\mathcal{I}_8$ - $\mathcal{I}_{12}$
credit amount	$\mathcal{I}_{13}$ - $\mathcal{I}_{16}$
savings	$\mathcal{I}_{17}$ - $\mathcal{I}_{21}$
housing	$\mathcal{I}_{22}$ - $\mathcal{I}_{24}$
foreign worker	$\mathcal{I}_{25}$

Table 2. *Binarisation of the attribute values*

when 13 attributes were removed from the input data. However, the predictive accuracy was marginally higher with the exclusion of the redundant attributes.

It is also worth noting that the average number of function/gradient evaluations is typically less when only seven attributes are used. A network constructed with only selected attributes as input, tends to reach a minimum of its error function before that of a neural network constructed with all attributes. In addition, the removal of irrelevant data through feature selection results in a simpler network construction with fewer links. A simpler network architecture reduces the computational costs, while a large network with many parameters may over-fit the training data and give a poor predictive accuracy on new data not in the training set. Finally, a simpler network results in a faster training time. Thus a network constructed using only those attributes which provide the most information for classification, can be said to be more computationally efficient than a network which models all attributes.

We can also observe from both tables, that the accuracy of the constructed networks on the test data decreases as the number of hidden nodes in the network increase. This emphasises the importance of limiting the number of hidden units in a network.

6 Conclusion

In this paper we presented an approach to the classification problem which combines feature selection (based on information theoretic approaches to identifying useful attributes) with neural networks. We outlined the motivations behind adopting such an approach—namely the high accuracy to be obtained using neural networks, its robustness to noise and its increased computational tractability given the reduced number of attributes over which the network is trained. We also presented results to show the validity of such an approach.

Given the reduced number of attributes selected, it is feasible that the task of rule extraction from the resulting neural network is simplified. A useful extension would be to investigate pruning algorithms to simplify the network further and then extract classification rules.

Future proposed work includes further experimentation with the number of features selected.

Units	Links	Acc. on train set (%)		Acc. on test set (%)	
		Ave.	Std. Dev.	Ave.	Std. Dev.
1	27	77.83	0.23	75.85	0.35
2	54	77.58	1.02	74.45	0.46
3	81	78.88	1.36	74.45	1.65
4	108	80.38	1.09	73.15	0.46

Table 3. *Results from the german credit problem with 7 selected attributes used as input*

Units	Links	Acc. on train set (%)		Acc. on test set (%)	
		Ave.	Std. Dev.	Ave.	Std. Dev.
1	74	85.99	0.31	72.66	1.13
2	148	86.49	1.48	72.36	2.21
3	222	88.19	2.34	72.36	0.17
4	296	92.69	0.75	71.46	1.75

Table 4. *Results from the german credit problem with all 20 attributes used as input*

References

[1] C. Aggrawal and P. S. Yu. Data mining techniques for associations, clustering and classification. In *Proceedings of the 3rd Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining (PAKDD-99)*, 1999.

[2] T. Cover. *Elements of Information Theory*. Wiley, 1991.

[3] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, pages 213–228, 1992.

[4] M. M. J. Shafer, R. Agrawal. Sprint: A scalable parallel classifier for data mining. In *Proceedings of the 22nd VLDB Conference*, 1996.

[5] H. Lu, R. Setiono, and H. Liu. Neurorule: A connectionist approach to data mining. In *Proceedings of the 21st VLDB Conference*, pages 478–489, 1995.

[6] G. Luger and W. Stubblefield. *Artificial Intelligence*. The Benjamin/Cummings Publishing Co., Inc., 1993.

[7] P. Murphy and D. Aha. *UCI Repository of Machine Learning Databases*. Department of Information and Computer Science, University of California, Irvine, CA, 1994.

[8] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[9] C. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
high	bad	high	none	$0 to $15k
high	unknown	high	none	$15 to $35k
moderate	unknown	low	none	$15 to $35k
high	unknown	low	none	$0 to $15k
low	unknown	low	none	over $35k
low	unknown	low	adequate	over $35k
high	bad	low	none	$0 to $15k
moderate	bad	low	adequate	over $35k
low	good	low	none	over $35k
low	good	high	adequate	over $35k
high	good	high	none	$0 to $15k
moderate	good	high	none	$15 to $35k
low	good	high	none	over $35k
high	bad	high	none	$15 to $35k

Table 5. *Data from credit history of loan applications*

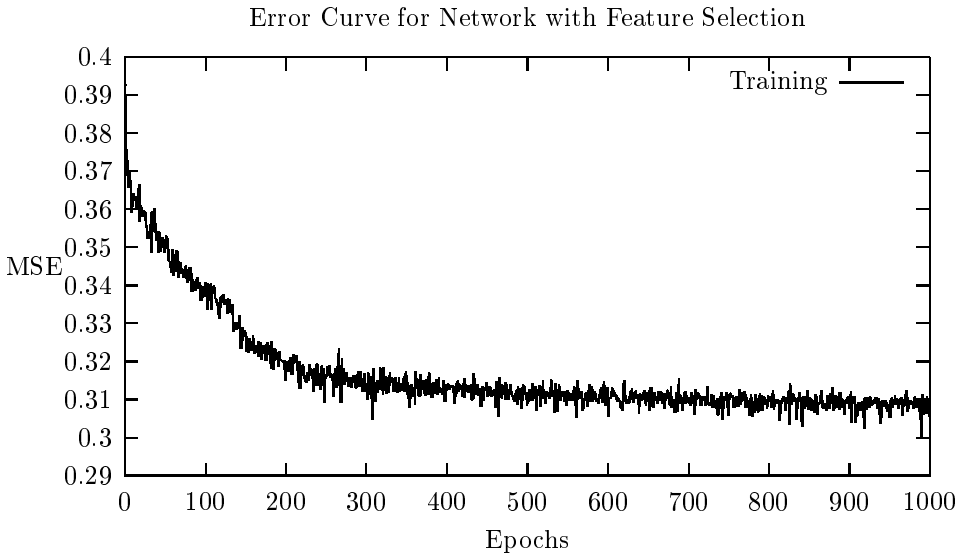

Figure 1. *Back-propagation convergence curve for neural network constructed using only selected attributes.*