OPTIMIZING DATA MINING TASKS USING RBF NEURAL NETWORKS

Ch.Srinivasa Rao

G.Y.Raja Kumar

G.Prasada Reddy

R.Bharat Kumar

ABSTRACT

This paper deals with the application of a well-known neural network technique, radial basis neural network for optimizing accuracy in data mining. In many data mining applications that address classification problems, feature and model selection are considered as key tasks. That is, appropriate input features of the classifier must be selected from a given (and often large) set of possible features and structure parameters of the classifier must be adapted with respect to these features and a given data set. This article elaborates on the role of neural networks in data mining, especially classification, and presents various ways of using them in this area. In order to do this the radial basis neural networks are reviewed, and an overview of the classification process and the training of neural networks are given.

Keywords

Data mining, Neural Network, Radial basis Neural Network

1. INTRODUCTION

The past two decades has seen a dramatic increase in the amount of information or data being stored in electronic format. This accumulation of data has taken place at an explosive rate. It has been estimated that the amount of information in the world doubles every 20 months and the size and number of databases are increasing even faster. The increase in use of electronic data gathering devices such as point-of-sale or remote sensing devices has contributed to this explosion of available data. The problem of effectively utilizing these massive volumes of data is becoming a major problem for all enterprises. Data storage became easier as the availability of large amounts of computing power at low cost ie the cost of processing power and storage is falling, made data cheap. There was also the introduction of new machine learning methods for knowledge representation based on logic programming etc. in addition to traditional statistical analysis of data. The new methods tend to be computationally intensive hence a demand for more processing power. Data mining, *the extraction of hidden predictive information from large databases*, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviours, allowing business to make proactive knowledge driven decisions. The automated, prospective analysis offered by data mining move beyond the analysis of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations. The data mining process consists of three basic stages: exploration, model building and pattern

definition. Fig.1 shows a simple data mining structure The data mining process consists of three basic stages: exploration, model building and pattern definition. Fig. 1.1 shows a simple data mining structure.

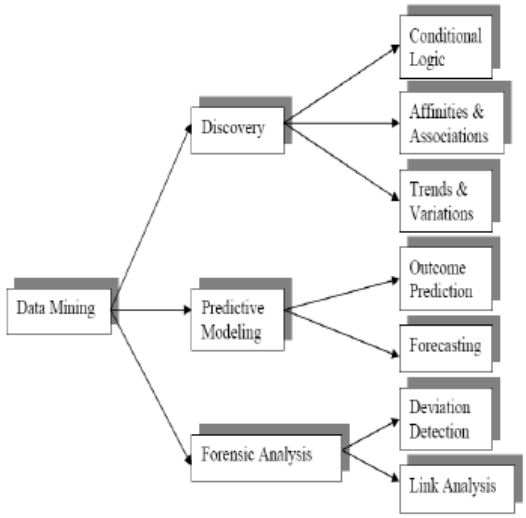

Fig 1 Data Mining Tasks

2.NEURAL NETWORK BASED DATA MINING

There is no general theory that specifies the type of neural network, number of layers, number of nodes (at various layers), or learning algorithm for a given problem. As such, today's network builder must experiment with a large number of neural networks before converging upon the appropriate one for the problem in hand.

2.1 The suitability of Neural Networks for Data Mining

For some problems, neural-networks provide a more suitable inductive bias for data mining than competing algorithms.

Inductive Bias: Given a fixed set of training examples, there are infinitely many models that could account for the data, and every learning algorithm has inductive bias that determines the model that it is likely to return. There are two aspects to the inductive bias of an algorithm: the *restricted hypothesis bias* and the *preference bias.* The hypothesis bias refers to the constraints that the algorithm places on the hypotheses that it is to construct. For example, the hypothesis space of a perceptron is limited to the linear discriminate functions. The preference bias of an algorithm is the preference ordering that it places on the models within its hypothesis space. For example, most algorithms try to fit a simple hypothesis to a given training set and then progressively explore more complex hypotheses until they find an acceptable fit. In some cases, the neural

networks have a more restricted hypothesis space bias than other learning algorithms. For example, for sequential and temporal prediction tasks represent a class of problems, for which, neural networks present a more appropriate hypothesis space. Recurrent networks, which are applied to most of these problems, are able to maintain state information from one time state to the next. This means that recurrent networks use their hidden units to learn derived information to the relevant task at hand, and they can make use of this derived information at one instant to help make prediction for the next instant. Although neural networks have an appropriate inductive bias for a wide class of problems, they are not commonly used for data mining tasks. There are two reasons: trained neural networks are usually not comprehensible and many neural network learning methods are slow, making them impractical for very large data sets.

2.2 Challenges Involved

The hypothesis represented by a trained neural network is defined by:

(a) The topology of the network

(b) The transfer functions used for hidden and output units

(c) The real-valued parameters associated with the network connections (i.e., the weights) and the units (i.e., the biases of sigmoid units).

Such hypotheses are difficult to comprehend for several reasons. First, typical systems have hundreds or thousands of real-valued parameters. These parameters encode the relationships between the input features and target values. Although single-parameter encodings are usually not hard to understand, the sheer number of parameters in a typical network can make the task of understanding them quite difficult. Second, in multi-layer networks, these parameters may represent non-linear, non-monotonic relationships between the input features and the target values. Thus, it is usually not possible to determine, in isolation, the effect of a given feature on the target value, because this effect may be medicated by the values of other features. These non-linear, non-monotonic relationships are represented by hidden units, which combine the inputs of multiple features, thus allowing the model to take advantage of dependencies among the features. Hidden units can be thought of, as representing higher-level, derived features. Understanding of hidden units is usually difficult because they learn distributed representations. In a distributed representation, the individual units do not correspond to well understand features in a problem domain. Instead, features, which are meaningful in the context of the problem domain, are often

encoded by patterns of activation across many hidden units. Similarly, each hidden unit may play a part in representing many derived features.

3 Extraction Methods

One approach to understanding the hypothesis represented by a trained neural network is to translate the hypotheses into a more comprehensible language various strategies using this approach have been investigated under the rubric of *rule extraction. Representation Language:* It is the language used by the extraction methods to describe the neural network's learned model. The languages that have been used include conjunctive inference rules, fuzzy rules, m-of-n rules, decision trees and finite state automata.

3.1 Extraction Strategy: It is the strategy used by the extraction method to map the model represented

by the trained neural network into a model in the new representation language. Specifically how the method explores the candidate descriptions and what level of descriptions it uses to characterize the neural network. That is, do the rules extracted by the method describe the behaviour as a whole (global methods), or the behaviour of individual units in the network (local methods) or something in between these two cases.

3.2 Network Requirements: The architectural and training requirements that the extraction method imposes on neural networks. In other words, the range of networks to which the method is applicable.

3.3 Rule Extraction Task

Consider the following example:

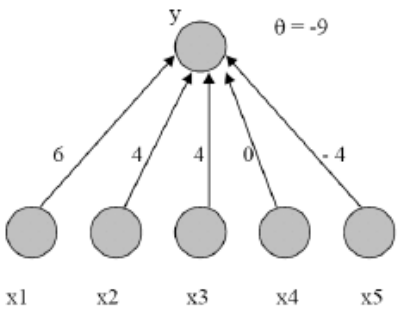

Fig 2. A Simple Network

Fig illustrates the task of rule extraction with a simple network. This one-layer network has five Boolean inputs and one Boolean output. The extracted symbolic rules specify conditions on the input features, that when satisfy, give a guaranteed output state. The output unit specifies a threshold function to compute its activation as follows:

$a_y = 1$, if $(\sum_i w_i a_i + \theta) > 0$

0, otherwise

The extracted rules are:

Y ¬ x1 ^ x2 ^ x3

Y ¬ x1 ^ x2 ^ -x5

Y ¬ x1 ^ x3 ^ -x5

Whenever a neural network is used for a classification problem, there is always an implicit decision procedure that is used to decide which class is predicted by the given network. In the simple example above, the decision procedure is to simply to predict y = true when the activation of the output unit was 1 and y = false when it was 0. In general, an extracted rule gives a set of conditions under which the network, coupled with its decision procedure, predicts a given class. One of the dimensions along which the rule extraction methods can be characterized is their level of description. One approach is to extract a set of global rules that characterize the output classes directly in terms of the inputs. An alternative approach is to extract a set of local rules, by decomposing the multiplayer networks into a collection of single layer networks.

4 The Rule Extraction Algorithm

The rule extraction algorithm is used for extracting comprehensible, symbolic representations from trained neural networks. The algorithm uses queries to induce a decision tree that approximates the concept represented by a given network. Experiments demonstrate that rule extraction algorithm is able to produce decision trees that maintain a high level of fidelity to their respective networks while being comprehensible and accurate. Unlike previous work in this area, the algorithm is general in its applicability and scales well to large networks and problems with high-dimensional input spaces.

4.1 Extracting Decision Trees

Our approach views the task of extracting a comprehensible concept description from a trained network as an inductive learning problem. In this learning task, the target concept is the function

represented by the network, and the concept description produced by our learning algorithm is a decision tree that approximates the network. However, unlike most inductive learning problems, we have available an oracle that is able to answer queries during the learning process. Since the target function is simply the concept represented by the network, the oracle uses the network to answer queries. The advantage of learning with queries, as opposed to ordinary training examples, is that they can be used to garner information precisely where it is needed during the learning process.

4.2 Membership Queries and the Oracle:

The role of the oracle is to determine the class (as predicted by the network) of each instance that is presented as a query. Queries to the oracle, however, do not have to be complete instances, but instead can specify constraints on the values that the features can take. In the latter case, the oracle generates a complete instance by randomly selecting values for each feature, while ensuring that the constraints are satisfied. In order to generate these random values, rule extraction algorithm uses the training data to model each feature's marginal distribution. Rule extraction algorithm uses frequency counts to model the distributions of discrete-valued features, and a kernel density estimation method (Silverman, 1986) to model continuous features. The oracle is used for three different purposes: (i) to determine the class labels for the network's training examples; (ii) to select splits for each of the tree's internal nodes; (iii) and to determine if a node covers instances of only one class.

4.2 Tree Expansion.

Unlike most decision-tree algorithms, which grow trees in a depth-first manner, rule extraction algorithm grows trees using a best-first expansion.

4.3 Split Types.

The role of internal nodes in a decision tree is to partition the input space in order to increase the separation of instances of different classes. This algorithm forms trees that use m-of-n expressions for its splits. An m-of-n expression is a Boolean expression that is specified by an integer threshold, m, and a set of n Boolean conditions. An m-of-n expression is satisfied when at least m of its n conditions are satisfied. For example, suppose we have three Boolean features, a, b, and c; the m-of-n expression 2-of-fa; :b; cg is logically equivalent to (a^ :b) . (a ^ c) . (: b ^ c).

4.4 Split Selection.

Split selection involves deciding how to partition the input space at a given internal node in the tree. A limitation of conventional tree induction algorithms is that the amount of training data used to select splits decreases with the depth of the tree. Thus splits near the bottom of a tree are often

poorly chosen because these decisions are based on few training examples. In contrast, because rule extraction algorithm has an oracle available, it is able to use as many instances as desired to select each split.

4.5 Stopping Criteria.

Rule extraction algorithm uses two separate criteria to decide when to stop growing an extracted decision tree. First, a given node becomes a leaf in the tree if, with high probability, the node covers only instances of a single class. To make this decision, rule extraction algorithm determines the proportion of examples that fall into the most common class at a given node, and then calculates a confidence interval around this proportion. Rule extraction algorithm also accepts a parameter that specifies a limit on the number of internal nodes in an extracted tree. This parameter can be used to control the comprehensibility of extracted trees, since in some domains; it may require very large trees to describe networks to a high level of fidelity.

5 Radial Basis Neural Network (RBN)

For RBN, vector distance between its single row input weight matrix w and the input vector p is multiplied by the bias b. The transfer function for RBN is radbas and the resulting input to this transfer function is output of the multiplier. See Fig 3 for model of a radial basis neuron

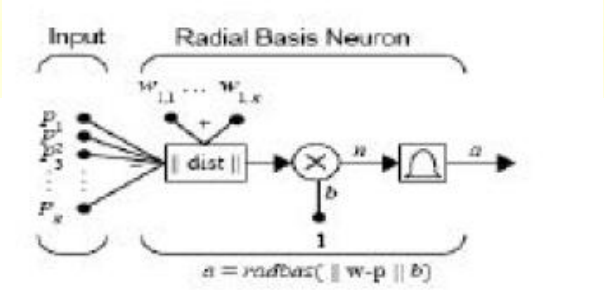

Fig 3 Model of a radial basis neuron

Probabilistic neural networks can be used for classification problems. When an input is presented, the first layer computes distances from the input vector to the training input vectors, and produces a vector whose elements indicate how close the input is to a training input. The second layer sums these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, a complete transfer function on the output of the second layer picks the maximum of these probabilities, and produces a 1 for that class and a 0 for the other classes. The architecture for this system is shown in the Figure 4 below.

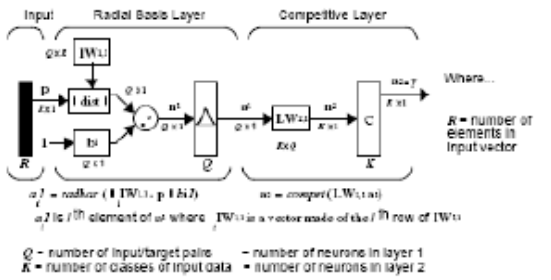

Fig 4 The architecture of RBF Neural Network

It is assumed that there are Q input vector/target vector pairs. Each target vector has K elements. One of these elements is 1 and the rest is 0. Thus, each input vector is associated with one of K classes. The first-layer input weights, IW1, 1 (net.IW {1, 1}) are set to the transpose of the matrix formed from the Q training pairs, P'. When an input is presented the ||dist|| box produces a vector whose elements indicate how close the input is to the vectors of the training set. These elements are multiplied, element by element, by the bias and sent the radbas transfer function. An input vector close to a training vector is represented by a number close to 1 in the output vector a1. If an input is close to several training vectors of a single class, it is represented by several elements of a1 that are close to 1.

The second-layer weights, LW1, 2 (net.LW {2, 1}), are set to the matrix T of target vectors. Each vector has a 1 only in the row associated with that particular class of input, and 0's elsewhere. (A function ind2vec is used to create the proper vectors.) The multiplication Ta1 sums the elements of a1 due to each of the K input classes. Finally, the second-layer transfer function, compete, produces a 1 corresponding to the largest element of n2, and 0's elsewhere. Thus, the network has classified the input vector into a specific one of K classes because that class had the maximum probability of being correct Probabilistic neural networks (PNN) can be used for classification problems. Their design is straightforward and does not depend on training. A PNN is guaranteed

to converge to a Bayesian classifier providing it is given enough training data. These networks generalize well.

6 CONCLUSION

The main objective of our work described in this article was to make evolutionary optimization of RBN networks applicable to a wide range of data mining problems. This paper implements the classification of data sets using radial basis neural networks and the accuracy is optimized using the rule extraction algorithm and the data sets are given as trained sample data sets to the neural networks .This paper implements using oracle, data base tool box ,neural network in mat lab 7.2.

5. REFERENCES

[1] IEEE Transactions on Neural Networks; "Data Mining in a Soft Computing Framework: A Survey", Sushmita Mitra, Sankar K. Pal and Pabitra Mitra. (January 2002, Vol. 13, No. 1)

[2] Using Neural Networks for Data Mining: Mark W. Craven, Jude W. Shavlik.

[3] Data Mining Techniques: Arjun K. Pujari.

[4] Introduction to the theory of Neural Computation: John Hertz, Anders Krogh, Richard G. Palmer

[5] Elements of Artificial Neural Networks: Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka.

[6] Neural Networks based Data Mining and Knowledge Discovery in Inventory Applications: Kanti Bansal, Sanjeev Vadhavkar, Amar Gupta

[7] Data Mining, An Introduction: Ruth Dilly, Parallel Computer Centre, Queen's University Belfast:. Data Mining Techniques: Electronic textbook, Stat soft: