INDUCTIVE LEARNING USING MULTISCALE CLASSIFICATION

Andrew P. Bradley Brian C. Lovell*†

* Dept. of Electrical and Computer Engineering, University of Queensland, Brisbane 4072, AUSTRALIA

† This project is supported by the Cooperative Research Centre for Sensor Signal and Information Processing

Multiscale Classification is a simple *rule-based* inductive learning algorithm. It can be applied to any N-dimensional real or binary classification problem to successively split the feature space in half to correctly classify the training data. The algorithm has several advantages over existing rule-based and neural network approaches: it is very simple, it learns very quickly, there is no network architecture to determine, there is an associated confidence with each classification rule, and noise can be automatically added to the training data to improve generalization.

1 Background

Artificial neural networks trained using back-propagation (backprop) have become a popular solution to many inductive learning problems. However, they have a number of problems, which may be severe, depending on the type of problem they are used to solve. Some of the main problems are:

- Learning from examples can be very slow.
- Local minima in the error surface may lead to convergence to an unacceptable solution.
- The difficulty of choosing a network which is capable of finding an acceptable solution.
- The difficulty of extracting classification rules in a meaningful form from the network.

There have been a number of modifications proposed to the backprop algorithm to reduce these problems such as using second order methods, conjugate gradient or line searches. Cascade-Correlation as proposed by Fahlman and Lebiere (1990) also attempts to overcome the problem of optimal network architecture by starting with a minimal network and then automatically adding and training new hidden units until an acceptable solution is found.

This paper wishes to demonstrate that some problems that are extremely difficult to solve using neural networks can be easily solved by a technique which we call *Multiscale Classification*. The proposed algorithm, though rule-based, has simple parallel and parallel-sequential hardware implementations. It should also be noted that the algorithm does not use any measure of attribute information content normally associated with top down induction of decision trees (*c.f.* CART, ID3).

2 The Basic Algorithm

All N inputs are considered to be real numbers in the range [0,1) which can then be expressed as binary fractions. This means that the entire feature space is mapped to the inside of a unit hypercube. Binary representation of the inputs is convenient because each successive bit position corresponds to a successive halving of feature space. In other words, the most significant bit indicates if an input is greater or less than 0.5; the second most significant bit increases this resolution to 0.25, the third to 0.125, and so on. By performing the classification one bit position at a time, the algorithm uses finer and finer levels of resolution to determine the eventual classification — hence the name multiscale

The multiscale classifier is based on a tree ar-

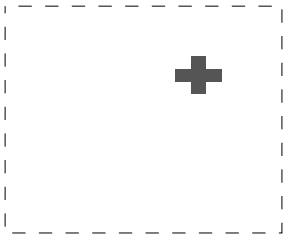

Figure 1: The first training point is learnt as one universal rule

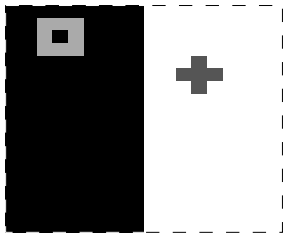

Figure 2: The second point is learnt, the feature space is split in half

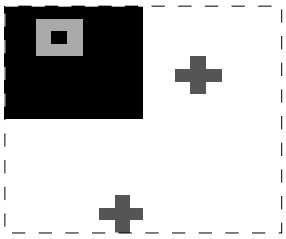

Figure 3: The third training point leads to a further rule split

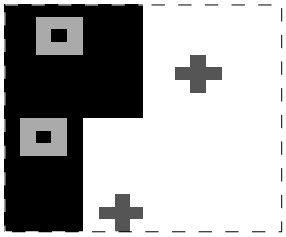

Figure 4: The fourth point requires going to the next level of resolution

chitecture. Each node of the tree may have up to 2^N branches or leaves. Normally there will be less than 2^N branches or leaves at each level of the tree, as rules can have "don't care" terms which can cover large regions of feature space. Here "don't care" terms indicate irrelevant inputs which may be exploited in the same manner as occurs in logic minimization. Classification is performed by simultaneously examining the most significant bit of each of the N inputs. This either yields the output class directly (a leaf of the tree), or tells us that we must examine the next bit (descend down a branch of the tree) to determine the output class. Examination of the next bit either yields the output class directly, or tells us to examine the following bit, and so on. In this way the training data is classified using the minimum level of input resolution required to separate the output classes, using only one bit of each input at each level of the tree .

The evolution of the classification tree as the algorithm learns is best illustrated by the simple example in two dimensions (two inputs) shown in Figures 1 through 4.

When the first training point is learnt the whole classification space is labeled with the training data's class in one universal rule, as shown in Figure 1. When the next training point arrives of a different class, this universal rule matches the input but yields the wrong class and so must be split into two rules, one for each class, as in Figure 2. Note that this splitting of the rules can be done by looking *only* at the most significant bit in the new input data. The third training point now leads to the second rule being split in half as in Figure 3. The fourth training point matches the third rule but is of a different class; in this case the second most significant bit of the input data must now be used to separate the two points. Thus the third rule is converted to a *branch* to the next level down.

Rules are either *leaves* and have a classification associated with them or are *branches* and simply lead down the next level of resolution and eventually a leaf.

3 The Twin Spirals Problem

The "Twin Spirals" benchmark was first proposed by Alexis Wieland of the MITRE Corporation. It consists of two continuous-valued inputs and a single output. The training set consists of 194 X-Y values, half of which are classified as black and the other half as white. These training points are arranged in two inter-

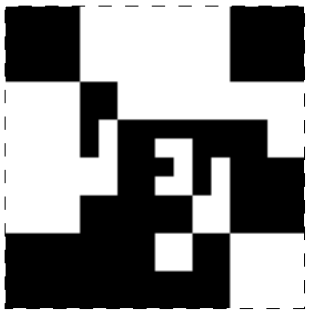

Figure 5: Classification after 1 training epoch

Figure 6: Classification after 4 training epochs

locking spirals that go around the origin three times. This problems has been shown to be extremely difficult for conventional backprop algorithms to solve (Fahlman and Lebiere, 1990).

Figures 5 and 6 show the development of the classification rules during training. At the fourth epoch the algorithm correctly classifies all the training data and has converged to a solution. For comparison, Lang and Witbrock (1988) claim that, with an appropriate choice of network, standard backprop solves this problem in 20,000 epochs, backprop with a modified error function requires 12,000 epochs, and Quickprop requires 8000 epochs. Fahlman and Lebiere (1990) state that the Cascade-Correlation algorithm needs about 1700 epochs.

A measure of generalization performance can be obtained from the twin-spirals benchmark by using another set of 194 test points which fall on the spirals midway between the training set points. The basic multiscale algorithm classifies 95% of this test data correctly — after stochastic generalization (see section 4) it achieves 100%. This should be compared to Cascade-Correlation which yields networks which obtain between 85 and 93% correct classification. Both algorithms obtain 100% correct classification on the training data.

4 Stochastic Generalization

Noise is often added to the training data of neural networks in order to improve their generalization abilities (Tesauro and Sejnowski, 1989). Normally it is very difficult to estimate *a priori* just how much noise to add to get good generalization. In the multiscale classification technique the amount of noise to add can be accurately gauged from the number of levels of resolution required to classify the original training data. Generalization can also be improved by pruning the rule tree generated so that rules with a low confidence (*i.e.,* classify few examples) are merged with adjacent rules that have a higher confidence.

Before stochastic generalization the multiscale classifier achieved 95% correct classification on the twin spiral test set after only 4 epochs of training. Figure 7 shows the classification boundaries after a further 90 epochs of stochastic generalization, which corresponds to 100% correct classification of the test set.

Figure 8 shows the confidence plot of the rule tree generated for the two-spirals problem. Increasing grey level shows increasing confidence of a black classification, while lighter grey levels show increasing confidence of a white classification. Figure 9 shows that after tree pruning, the classifier still obtained 100% but the rule tree was reduced in size by about 25%.

This level of generalization performance is better than either Cascade-Correlation (85–93%) or backprop neural networks with a far lower computational burden.

5 Further Comments

Other features of the multiscale classifier are:

1. **Guaranteed Convergence:** The multiscale classification algorithm will converge to 100% correct classification on any finite training set as long as the output classes are disjoint, *i.e.,* no two examples can have exactly the same feature space co-ordinates and different output classes.

Figure 7: Classification after 90 noisy training epochs

Figure 8: Confidence plot of spiral classification

2. **Localized Forgetting:** Because the algorithm classifies examples by examining one bit position at a time, the exact locations of the examples that originally generated a particular rule leaf are not known at the next level down. When the rule leaf is split, some examples may be "forgotten" until they are relearnt on the next training epoch. Rule splitting can only occur when a new example is misclassified by an existing rule leaf, so only "neighbouring" examples are at risk of being forgotten.

3. **Learning Speed:** The multiscale classifier halves feature space to locate output class clusters (similar to a binary search) and so tends to very rapidly modify class boundaries.

6 Conclusions

Studies of the performance of the multiscale classifier on the Monk's Problems (Thrun, 1991), the XOR problem, and Overlapping Gaussian Output Classes are reported in

Figure 9: Pruned classification after 90 noisy training epochs

(Lovell and Bradley, 1993). In all cases the multiscale classifier offered far faster learning and comparable (if not better) generalization performance than the best reported results for either backprop neural networks or advanced classifiers.

It is envisaged that this technique will be applied in the field of Medical Diagnostics where it is ideally suited because of its parallel implementation, rule-based nature, and confidence level output.

7 References

S. E. Fahlman and C. Lebiere (1990), *The cascade-correlation learning architecture*, in "Advances in Neural Information Processing Systems", D. S. Touretzky, ed., San Mateo: Morgan Kaufmann, 524–532, Volume 2.

K. J. Lang and M. J. Witbrock (1988), *Learning to tell two spirals apart*, in "Proceedings of the 1988 Connectionist Summer School", Morgan Kaufman.

B. C. Lovell and A. P. Bradley (1993), *The Multiscale Classifier*, Center for Sensor Signal and Information Processing, CSSIP Technical Report.

G. Tesauro and T. J. Sejnowski (1989), *A parallel network that learns to play backgammon*, Artificial Intelligence **39**, 357–390.

S. B. Thrun (1991), *The MONK's problem: A performance comparison of different learning algorithms*, Carnegie-Mellon University, Technical Report.