An algorithmic method to build good training sets for neural-network classifiers

Fabio Tamburini **Renzo Davoli**

Technical Report UBLCS-94-18

July 1994

Laboratory for Computer Science
University of Bologna
Piazza di Porta S. Donato, 5
40127 Bologna (Italy)

The University of Bologna Laboratory for Computer Science Research Technical Reports are available via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` in compressed PostScript format. Abstracts are available from the same host in the directory `/pub/TR/UBLCS/ABSTRACTS` in plain text format. All local authors can be reached via e-mail at the address *last-name*`@cs.unibo.it`.

UBLCS Technical Report Series

93-19 *HERMES: an Expert System for the Prognosis of Hepatic Diseases*, I. Bonfà, C. Maioli, F. Sarti, G.L. Milandri, P.R. Dal Monte, September 1993.

93-20 *An Information Flow Security Property for CCS*, R. Focardi, R. Gorrieri, October 1993.

93-21 *A Classification of Security Properties*, R. Focardi, R. Gorrieri, October 1993.

93-22 *Real Time Systems: A Tutorial*, F. Panzieri, R. Davoli, October 1993.

93-23 *A Scalable Architecture for Reliable Distributed Multimedia Applications*, F. Panzieri, M. Roccetti, October 1993.

93-24 *Wide-Area Distribution Issues in Hypertext Systems*, C. Maioli, S. Sola, F. Vitali, October 1993.

93-25 *On Relating Some Models for Concurrency*, P. Degano, R. Gorrieri, S. Vigna, October 1993.

93-26 *Axiomatising ST Bisimulation Equivalence*, N. Busi, R. van Glabbeek, R. Gorrieri, December 1993.

93-27 *A Theory of Processes with Durational Actions*, R. Gorrieri, M. Roccetti, E. Stancampiano, December 1993.

94-1 *Further Modifications to the Dexter Hypertext Reference Model: a Proposal*, W. Penzo, S. Sola, F. Vitali, January 1994.

94-2 *Symbol-Level Requirements for Agent-Level Programming*, M. Gaspari, E. Motta, February 1994.

94-3 *Extending Prolog with Data Driven Rules*, M. Gaspari, February 1994.

94-4 *Schedulability Checking of Data Flow Tasks in Hard-Real-Time Distributed Systems*, R. Davoli, L. A. Giachini, March 1994.

94-5 *A Shared Dataspace Language and its Compilation*, M. Gaspari, March 1994.

94-6 *Exploring the Coordination Space with LO*, S. Castellani, P. Ciancarini, April 1994.

94-7 *Comparative Semantics of LO*, S. Castellani, P. Ciancarini, April 1994.

94-8 *Distributed Conflicts in Communicating Systems*, N. Busi, R. Gorrieri, G. Siliprandi, April 1994.

94-9 *Pseudopolar Array Mask Algorithm for Spherical Coordinate Grid Surfaces*, A. Amoroso, G. Casciola, May 1994.

94-10 *MPA: a Stochastic Process Algebra*, M. Bernardo, L. Donatiello, R. Gorrieri, May 1994.

94-11 *Describing Queueing Systems with MPA*, M. Bernardo, L. Donatiello, R. Gorrieri, May 1994.

94-12 *Operational GSPN Semantics of MPA*, M. Bernardo, L. Donatiello, R. Gorrieri, May 1994.

94-13 *Experiments in Distributing and Coordinating Knowledge*, P. Ciancarini, May 1994.

94-14 *A Comparison of Parallel Search Algorithms Based on Tree Splitting*, P. Ciancarini, May 1994.

94-15 RELACS: *A Communications Infrastructure for Constructing Reliable Applications in Large-Scale Distributed Systems*, Ö. Babaoğlu, M.G. Baker, R. Davoli, L.A. Giachini, June 1994.

94-16 *Replicated File Management in Large-Scale Distributed Systems*, Ö. Babaoğlu, A. Bartoli, G. Dini, June 1994.

94-17 *Parallel Symbolic Computing with the Shared Dataspace Coordination Model*, P. Ciancarini, M. Gaspari, July 1994.

94-18 *An Algorithmic Method to Build Good Training Sets for Neural-Network Classifiers*, F. Tamburini, R. Davoli, July 1994.

94-19 *On Group Communication in Large-Scale Distributed Systems*, Ö. Babaoğlu, A. Schiper, July 1994.

An algorithmic method to build good training sets for neural-network classifiers

Fabio Tamburini [1] **Renzo Davoli** [2]

Technical Report UBLCS-94-18

July 1994

Abstract

To classify complex patterns using a neural network and a supervised training algorithm, one needs a complete training set that represents all possible characteristics of the examined problem. Building good training sets for neural network classifiers can be very difficult, especially if the problem involves complex patterns barely feasible by human perception. This paper proposes an algorithmic method for building small training sets that endow the net with small generalization errors. The effectiveness of the method is shown with two examples of image classification.

1. C.I.L.T.A., University of Bologna, Via Dante, 15, 40125 Bologna, Italy.
2. Department of Mathematics, University of Bologna, Piazza Porta S. Donato, 5, 40127 Bologna, Italy.

1 Introduction

The problem of classifying images, stimuli, bit strings, or, more generally, patterns into well-defined sets of classes is a broad, highly complex problem. Such problems can be successfully solved with multilayer feedforward networks and supervised algorithms.

Many studies have shown that one of the main parameters that affects the generalization ability of a neural network is the information content of the training examples [3, 7, 8]. One of the first problems that a neural network designer encounters when using supervised learning algorithms such as backpropagation algorithm [6] is how to build a training set that permits the net to learn the decision method and generalize it to the classification problem. If the classification problem is very complex, choosing the patterns to build the training set can constitute a difficult obstacle. This situation arises because human perception of the stimuli can differ greatly from those of a neural network. Hand-made training sets may contain patterns that do not help the net to learn and correctly generalize its behaviour for the classification problem.

Each classification-problem category may have several unique characteristics. These can vary greatly class to class and can escape human perception. In these cases the task of building good training sets can be very difficult, especially for problems involving large amounts of data, such image classification. This data that is often analogical and previously subjected to various transformations that can often simplify the classification task, but, at the same time, render the input patterns totally confusing to the human eye. For these reasons is necessary to develop a practical, general method that affords good training sets without requiring human choice about the training-set patterns.

There are some theoretical methods that analytically measure the ability of a neural net to generalize in the average case [7] and worst case [1] as function of the number of patterns making up the training set. These methods require additional knowledge that can be analytically calculated only for some simple problems; they are therefore not a real help when solving practical complex problems. They consider as a parameter only the total number of training set patterns, ignoring the real information content of any given pattern.

Other methods propose the inclusion of hints to help the learning and generalization processes [2, 5]. These methods also require additional knowledge.

2 The proposed method

The proposed practical, step-by-step method builds good, small training sets that give the network with good generalization ability. A poor training set does not include patterns representing all the problem characteristics; thus the network does not correctly generalize, as it does not have all the required information. Vice versa, a good training set has the minimum number of patterns representing all possible problem characteristics; such a training set supplies the network with all necessary information to correctly generalize over unknown patterns. The main problem is to identify those patterns that provide the neural network with the maximum information and include them in the training set. This choice is not made by human operators, but by the network itself.

The idea is quite simple: we suppose that those patterns that presents classification difficulties contain much information about the problem. They presumably are the patterns that best represent the different problem classes, supplying the network with good examples. If the net learns successfully over the training set patterns, we can expect it to perform

well with unknown patterns, since it has already learned most of the characteristics that distinguish the problem classes.

To apply this method, all available patterns are first randomly divided into two disjoint sets: the Working Set, *WS*, containing 15-20% of the total number of patterns and the Verification Set, *VS*, containing the remaining patterns. *WS* isused to build the training set; *VS* to verify its goodness. It is important that *WS* contain patterns belonging to all problem classes.

The algorithm itself starts by creating a small training set, *TS*, containing exactly one pattern for each problem class, randomly chosen from *WS*. Then we train the neural network using *TS* and a supervised learning algorithm, such as classical backpropagation. If the process converges, we obtain a set of weights and thresholds, *W*, that identifies the network. Using, *W*, we can compute the number of patterns belonging to *VS* that the net misclassifies. If the result does not satisfy the objectives, we can try adding new meaningful patterns from *WS*. To do this, we evaluate the least mean square (LMS) error of the network over all *WS* patterns. At this point we take the *STEP* (a small constant e.g. 2, 3 or 4) patterns that show the highest LMS error and add them to *TS*. We thus presume to have added the most significant patterns from *WS*, giving the network more information about the problem classes. We next retrain the network with the new *TS* and repeat the above steps until we are satisfied with the network generalization error.

It is preferable to add only a few patterns at each step; otherwise we might add patterns containing overlapping information about problem characteristics that do not help the network learn new details. For example, if pattern X is very similar to pattern Y and their characteristics are not adequately represented in *TS*, it is likely that the network will show high LMS error when evaluating X and Y. Adding both patterns does not help, because if they are very similar we would expect the network to work properly learning only X or Y and generalizing its behaviour to the other pattern. Adding only a few patterns at a time reduces the risk of introducing redundant patterns to *TS*.

The following scheme details the algorithm:

Input Data:
WS: working set.
VS: verification set.
NCL: number of problem classes.
STEP: number of patterns to add at each step.
ε: desired generalization error.
Output Data:
TS: training set.

• Create *TS* by randomly choosing *NCL* patterns (one for each problem class) from *WS*.
Loop
• Train the network using *TS* as a training set: if the process converges, we obtain a set, *W*, of weights and thresholds.
• Compute the net classification error, *We*, over set *VS* using *W* as the set of weights.
If ($We \leq \varepsilon$) **Then Exit.**
• Compute the LMS error of the *WS* patterns.
• Extract the *STEP* patterns from *WS* that exhibit the highest LMS errors.
• Add these patterns to *TS*.
End Loop
• Use *TS* as the final training set.

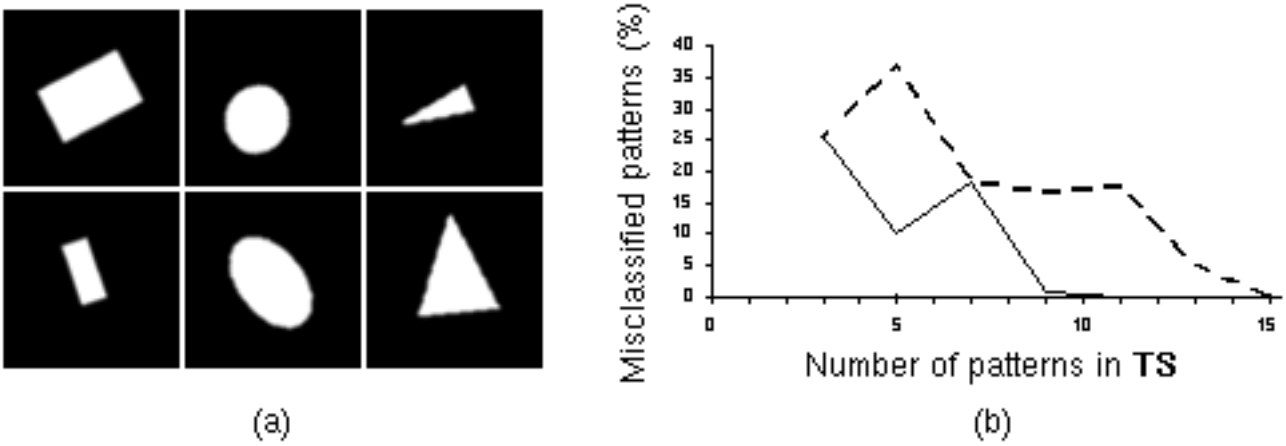

Figure 1. (a) Geometric objects. (b) Percent of images misclassified as a function of the number of patterns in the training set.

We have no guarantee that the process will end, for two reasons. First of all, we cannot be sure that a learning process, using a given training set, reaches a minimum that allows the method to converge. Secondly, the algorithm may not terminate if the desired generalization error, ε, is set too low. Careful selection of ε reduces this risk.

This method constructs the training set algorithmically and can be easily programmed.

3 Experimental results

Some practical applications were chosen to test the effectiveness of the method. These recognition problems concern the classification of objects in digitized images and involve large amounts of information due to image complexity, large numbers of pixels, etc. In such cases, it is difficult to choose which images should make up the training set. Human perception cannot guarantee that hand-made training sets will work in these cases.

Classification of digitized objects often involves mathematical transformations that extract shape information from highly complex schemes [4, 9]. These methods transform digitized images into complex patchworks of colors, making it impossible for human perception to distinguish any shapes or obtain useful information. The very difficult problem of creating efficient training sets is overcome with the proposed method.

The first application is the classification of simple geometric objects (Fig. 1a). These objects (rectangles, ellipses and triangles) were contained in 128×128 pixel images and processed by a three-layer neural network after a shape-extraction transformation [4]. The backpropagation algorithm was used to train the network.

Fig. 1b shows the generalization error (percent of images misclassified) in *VS* as a function of the number of patterns in the training set.

The dashed line represents the network error for a randomly chosen training set; the solid line the results with the proposed method (all points are the average of three learning-evaluation trials with the same training set). As we can see, the training set obtained with the proposed method is smaller then the randomly built one (note that for this type of problem

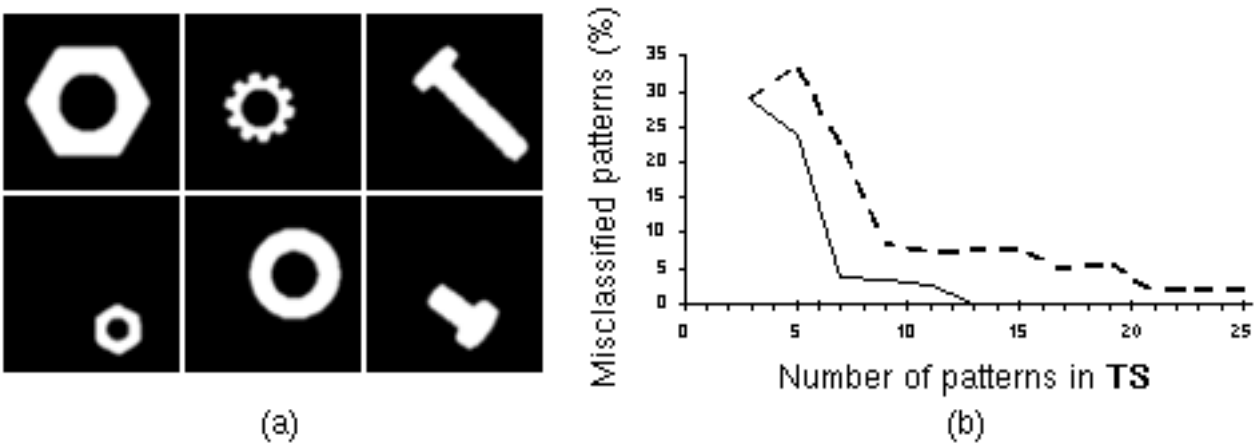

Figure 2. (a) Mechanical parts. (b) Percent of images misclassified as a function of the number of patterns in the training set.

a hand-made training set may be no better or worse than a randomly chosen one).

The second application shows the improvement in efficiency obtained with this method. The problem is to classify mechanical parts, such as nuts, washers and screws (Fig 2a). Fig. 2b shows the results.

In this case we note a significant difference between the generalization error obtained with the proposed method and that for a randomly built training set. The training set built using the proposed algorithm gives a better generalization error and contains fewer images than the random one.

4 Conclusions

The method proposed can help solve the difficult problem of building useful training sets for complex tasks involving elaborate patterns. The training sets obtained are often smaller than random or hand-made ones, resulting in nets that exhibit better generalization errors.

References

[1] Abu-Mostafa, Y.S. 1989. "The Vapnik-Chervonenkis dimension: information versus complexity in learning." *Neural Computation*, **1**, 312-317.

[2] Al-Mashouq, K.A., Reed, I.S. 1991. "Including hints in training neural nets." *Neural Computation*, **3**, 418-427.

[3] Baum, E.B., and Haussler, D. 1989. "What size net gives valid generalization?" *Neural Computation*, **1**, 151-160.

[4] Davoli, R., and Tamburini, F. 1993 "DATA algorithm: a numerical method to extract shape information from gray scale images." *Proc. Image analysis and synthesis*, ed by Pölzleitner, W., and Wenger, E., Vienna, 219-230.

[5] Omlin, C.W., and Giles, C.L. 1992. "Training second-order recurrent neural networks using hints." *Machine learning: Proc. of the ninth international conference (ML92)*, Sleeman, D., and Edwards, P. (Eds), Morgan Kaufmann, San Mateo, CA.

[6] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. "Learning internal representation by error propagation." *Parallel distributed processing: Explorations in microstructures of cognition,* Vol. 1, chap. 8.

[7] Schwartz, D.B., Samalam, V.K., Solla, S.A., and Denker, J.S. 1990. "Exhaustive learning." *Neural Computation,* **2**, 371-382.

[8] Tishby, N., Levin, N., and Solla, S.A. 1989. "Consistent inference of probabilities in layered networks: predictions and generalization." *International Joint Conference on Neural Networks*, Washington, 403-410.

[9] Zwicke, P.E., and Kiss I. Jr. 1983. "A new implementation of the Mellin transform and its application to radar classification of ships." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, n.2, 191-199.