Available online at www.sciencedirect.com

SCIENCE DIRECT®

European Journal of Operational Research 169 (2006) 508–519

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Path relinking and GRG for artificial neural networks ☆

Abdellah El-Fallahi [a], Rafael Martí [a,*], Leon Lasdon [b]

[a] *Departamento de Estadística e Investigación Operativa, Universitat de València, 46100 Burjassot (Valencia), Spain*

[b] *Department of Management Science and Information Systems, The University of Texas at Austin, Austin, TX 78712, USA*

Received 30 June 2003; accepted 20 February 2004
Available online 29 September 2004

Abstract

Artificial neural networks (ANN) have been widely used for both classification and prediction. This paper is focused on the prediction problem in which an unknown function is approximated. ANNs can be viewed as models of real systems, built by tuning parameters known as *weights*. In training the net, the problem is to find the weights that optimize its performance (i.e., to minimize the error over the training set). Although the most popular method for training these networks is back propagation, other optimization methods such as tabu search or scatter search have been successfully applied to solve this problem. In this paper we propose a path relinking implementation to solve the neural network training problem. Our method uses GRG, a gradient-based local NLP solver, as an improvement phase, while previous approaches used simpler local optimizers. The experimentation shows that the proposed procedure can compete with the best-known algorithms in terms of solution quality, consuming a reasonable computational effort.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Neural networks; Metaheuristics; Evolutionary computation

1. Introduction

Artificial neural networks offer a general framework for representing non-linear mappings from several input variables to several output variables. They are built by tuning a set of parameters known as *weights*, and can be considered as an extension of the many conventional mapping techniques. In classification or recognition problems the net's outputs are categories, while in prediction or approximation problems they are continuous variables. Although this paper is focused on the prediction problem, most of the key issues in the net functionality are common to both.

In the process of training the net (*supervised learning*), the problem is to find the values of the weights w that minimize the error across a set of input/output pairs (patterns) called the training

☆ Research partially supported by the Ministerio de Ciencia y Tecnología of Spain: TIC2003-C05-01/TIC2002-10886E and by the AVCiT of the Generalitat Valenciana: Grupos03/189.

* Corresponding author. Tel.: +34 96 386 4362; fax: +34 96 386 4735.
E-mail address: rafael.marti@uv.es (R. Martí).

set E. For a single output and input vector $\mathbf{x}$, the error measure is typically the root mean squared difference between the predicted output $p(x,w)$ and the actual output value $f(x)$ for all the elements x in E (RMSE); therefore, the training is an unconstrained non-linear optimization problem, where the decision variables are the weights and the objective is to reduce the training error. Ideally, the set E is a representative sample of points in the domain of the function f that we are approximating; however, in practice it is usually a set of points for which we know the f-value.

$$\min_{w} \operatorname{error}(E,w) = \sqrt{\frac{\sum_{x\in E}(f(x)-p(x,w))^2}{|E|}}.$$

The main goal in the design of an ANN is to obtain a model which makes good predictions for new inputs (i.e., to provide good generalization). Therefore the net must represent the systematic aspects of the training data rather than their specific details. The standard way to measure the generalization provided by the net consists of introducing a second set of points in the domain of f called the testing set T. We assume that no point in T belongs to E and $f(x)$ is known for all x in T. Once the optimization has been performed and the weights have been set to minimize the error in $E(w = w^*)$, the error across the testing set T is computed ($\operatorname{error}(T,w^*)$). The net must exhibit a good fit between the target f-values and the output (prediction) in the training set and also in the testing set. If the RMSE in T is significantly higher than that one in E, we will say that the net has *memorized* the data, instead of *learning* them (i.e., the net has over-fitted the training data).

If we consider a net with too few weights, it will not be able to fit the training data, while too many weights may provide a perfect fit to the training data, but a poor fit over the test set T. There is no direct rule to compute the optimum number of parameters when designing a neural net. The tradeoff between the flexibility and generalization properties of the system must be empirically determined in each particular case.

Several models inspired by biological neural networks have been proposed throughout the years, beginning with the *perceptron* introduced by Rosemblatt (1962). He studied a simple architecture where the output of the net is a transformation of a linear combination of the input variables and the weights. The transformation g is a threshold activation function where $g(x) = -1$ for $x < 0$ and $g(x) = 1$ for $x > 0$. This model has been the subject of extensive study in the statistics literature under the generic name of "logistic discrimination" where the activation function is given by the sigmoid function $g(x) = 1/(1 + e^{-x})$, which produces a smooth approximation to the original step function.

Minsky and Papert (1969) showed that the perceptron can only solve linearly separable classification problems and is therefore of limited interest. A natural extension to overcome its limitations is given by the so called *multilayer-perceptron* or simply multilayer neural networks, consisting of a set of nodes N and a set of arcs A. In two-layered networks, N is partitioned into three subsets: N_{I}, input nodes, N_{H}, hidden nodes and N_{O}, output nodes. The arcs go from N_{I} to N_{H} or from N_{H} to N_{O}; in this sense we say that the net is a layered graph. We assume that there are n input variables, so $|N_{\mathrm{I}}| = n$, and a single output. The neural network has m hidden neurons ($|N_{\mathrm{H}}| = m$) with a bias term in each hidden neuron.

Kolmogorov (1957) proved that every continuous function of several variables can be represented as the superposition of a small number of one-variable functions with a certain accuracy level. This result explains, in a theoretical sense, why neural networks work, but in practice it is of limited interest. If we put Kolmogorov's theorem in neural network terms, we will obtain a net with non-smooth activation functions that depend on the mapping that we are trying to approximate and provide poor generalization. Recent studies have proved that multilayer networks with one hidden layer and specific "squashing" functions are able to approximate any function. Hornik et al. (1989) show that neural networks are universal approximators and they conclude that any lack of success in applications must arise from inadequate learning or an insufficient number of hidden units. However, practical results show that in some "difficult" functions, neural networks provide approximations of low quality.