Available online at www.sciencedirect.com
SCIENCE DIRECT®

NEUROCOMPUTING

Neurocomputing 69 (2006) 368–386

www.elsevier.com/locate/neucom

Adaptive conjugate gradient algorithm for perceptron training

G. Nagaraja*, R.P. Jagadeesh Chandra Bose

Department of Computer Science and Engineering, IIT Bombay, Mumbai 400 076, India

Received 6 July 2004; received in revised form 1 March 2005; accepted 1 March 2005
Available online 25 July 2005
Communicated by T. Heskes

Abstract

An adaptive algorithm for function minimization based on conjugate gradients for the problem of finding linear discriminant functions in pattern classification is developed. The algorithm converges to a solution in both consistent and inconsistent cases in a finite number of steps on several datasets. We have applied our algorithm and compared its performance with the adaptive versions of the Ho-Kashyap procedure (AHK). We have also compared the batch version of the algorithm with the batch mode AHK. The results show that the proposed adaptive conjugate gradient algorithm (CGA) gives vastly superior performance in terms of both the number of training cycles required and the classification rate. Also, the batch mode CGA performs much better than the batch mode AHK.

Keywords: Perceptron; Conjugate-gradient; Linear separability; Linear inequalities; Training

Notation

We use capital letters to denote matrices, lower case bold letters to denote vectors and Greek letters to denote scalars. Superscript t denotes the transpose of a matrix or a vector. $\|\cdot\|$ denotes the inner product norm. $|\cdot|$ denotes the absolute value for a scalar and component-wise absolute value for a vector.

*Corresponding author.
E-mail addresses: gn@cse.iitb.ac.in (G. Nagaraja), jcbose@gmail.com (R.P.J.C. Bose).

0925-2312/$ - see front matter
doi:10.1016/j.neucom.2005.03.007

1. Introduction

Perceptrons are among the earliest and most basic models of artificial neural networks, yet they are central to many complex neural net applications. Though perceptrons are limited in their power, they are still of importance because of their inherent simplicity in classifying linearly separable problems. Training a perceptron involves solving a set of linear inequalities. Several training algorithms for perceptrons have been proposed in the literature. The LMS [16] (or Widrow–Hoff rule), the perceptron rule [14], the Ho–Kashyap [10] and its adaptive versions AHK I, AHK II and AHK III [8] are a few of the standard algorithms that are in use.

We follow the convention in [6]. A two class pattern classification problem for a perceptron can be formulated as follows: suppose we have a set of m samples $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m$ some of which are labeled ω_1 and some ω_2 with each $\mathbf{a}_i$ being an n-dimensional vector (a_{i0} is a fixed input set to 1 for the bias). We want to use these samples to determine a weight vector $\mathbf{x}$ in a linear discriminant function $g(\mathbf{x}) = \mathbf{a}^t\mathbf{x}$. A sample $\mathbf{a}_i$ is classified correctly if $\mathbf{a}_i^t\mathbf{x} > 0$ and $\mathbf{a}_i$ is labeled ω_1 or if $\mathbf{a}_i^t\mathbf{x} < 0$ and $\mathbf{a}_i$ is labeled ω_2. We can *normalize* the two-category case by replacing all of the samples labeled ω_2 by their negatives thus making us look for a weight vector $\mathbf{x}$ such that $\mathbf{a}^t\mathbf{x} > 0$ for *all* of the samples. Such a weight vector is called a *separating vector* or a *solution vector*. Thus, it can be formulated as: find an n-vector $\mathbf{x}$ such that

$$\mathbf{a}_i^t\mathbf{x} > 0, \quad i = 1, \ldots, m.$$

Generally $m > n$ in pattern classification problems. The above system of linear inequalities is said to be *consistent* if a solution exists; otherwise, it is said to be *inconsistent*. One can come across many inconsistent systems in the context of pattern classification making them very important [6]. In such a case, we are interested in a "solution" that is "best" in some sense.

Training a perceptron and a neural network, in general, involves adjusting the weights to produce the desired output. Thus training a neural network is, in most cases, an exercise in numerical optimization of a usually nonlinear objective function. Several formulations and algorithms for these systems exist with the prominent ones being the fixed increment rule [5], the linear programming (LP) approach [5] and the Ho-Kashyap algorithm (HKA) [10,5,9].

Gradient descent and conjugate gradient are two widely used techniques for solving a set of linear inequalities. The approach taken here is to define a criterion function that is to be minimized and the vector $\mathbf{x}$ that minimizes the function is a solution vector. Conjugate gradient-based methods are fast as compared to gradient descent techniques and employ a series of line searches in weight or parameter space. Duda et al. [6] summarize the conjugate gradient method in simple terms as:

> In conjugate gradient approach, one picks the first descent direction (for e.g., simple gradient) and moves along that direction until the local minimum in error is reached. The second direction is then computed: This direction, the "conjugate direction", is the one along which the gradient doesn't change its *direction*, but