Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition Letters 26 (2005) 1866–1873
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Probabilistic neural network playing and learning Tic–Tac–Toe

Jiří Grim *, Petr Somol, Pavel Pudil

Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, P.O. Box 18, CZ-18208 Prague 8, Czech Republic

Available online 29 April 2005

Abstract

A probabilistic neural network is applied as a tool to approximate the statistical evaluation function for a simple version of the game Tic–Tac–Toe. We solve the problem by a sequential estimation of the underlying discrete distribution mixture of product components.

Keywords: Probabilistic neural networks; Distribution mixtures; Supervised learning; Sequential EM algorithm; Weighting of data; Playing Tic–Tac–Toe

1. Introduction

The design of probabilistic neural networks (PNN) consists in estimating distribution mixtures from training data by means of EM algorithm. In the last years PNN have been applied successfully to various problems of statistical pattern recognition (Grim et al., 2002, 2003b) nevertheless, a weak point in application of PNN to practical problems is a limited size of training data samples. Estimation of mixture parameters especially in case of high dimensional data is a difficult task. If the data set is not large enough we are likely to obtain unreliable (local) solutions accompanied with the problems of over-fitting and bad generalizing properties.

If we want to demonstrate the true capability of PNN we have to avoid the specific small sample conditions. We can extend the training data e.g. by problem invariant transforms (Grim et al., 2002) or we can use artificially generated data (Grim et al., 2003b). One of the few practical problems which can be solved by using potentially infinite data sequences arises in playing games. If we succeed to implement a reliable learning principle then the underlying neural network can improve its own game level simply by learning from simulated games. However, the design of neural networks playing and learning even a simple game is

* Corresponding author. Fax: +420 284683031.
E-mail address: grim@utia.cas.cz (J. Grim).

0167-8655/$ - see front matter
doi:10.1016/j.patrec.2005.03.008

difficult. Let us recall that the meaning of very similar situations may often be completely different and, unlike the standard pattern recognition, the statistical properties of the underlying decision problem may change essentially with the opponent strategy. The statistical decision problem is also influenced by the changing parameters in the course of learning.

One of the few examples of a successful self-learning neural network is the TD-Gammon of Tesauro (1995). It is a computer program based on a standard multilayer perceptron that trains itself to become an evaluation function for the game of backgammon. The training scheme combines the popular back-propagation algorithm with the temporal difference (TD) learning principle. TD-Gammon won the backgammon computer-championship and achieved the level of the best human masters.

The goal of the present paper is to demonstrate that PNN can learn to play a simple version of Tic–Tac–Toe. There are at least three arguments to apply PNN to playing games. (a) An important aspect of games is the possibility to produce nearly arbitrarily large training data sets which are desirable for estimating high-dimensional probability distributions. In view of the potentially infinite training data sequences, we also have a unique possibility to verify sequential learning methods. (b) The design of PNN consists in estimating distribution mixtures from training data by means of the maximum-likelihood criterion. Consequently, the related parameters of neurons may reflect the statistical nature of the underlying decision-making more closely than other approaches like e.g. linear regression or multilayer perceptrons (Furnkranz and Kubat, 2001). (c) Also it may be useful that the application of PNN is possible without using any problem-specific feature extraction procedures since, in case of evaluating game situations, the feature design often becomes an uneasy task.

2. Decision problem of Tic–Tac–Toe

The standard game of "Tic–Tac–Toe" (also called "Noughts and Crosses") is a two player zero-sum game on a square grid. The two players alternately mark one empty cell with a cross "×" or a nought "○" respectively. A player wins by being the first to obtain four matching markers on any row, column or any diagonal. The winning number of markers in a line may be chosen differently. In case of four markers the starting player can always win the game during the first nine moves (4.5 complete moves). Consequently the quality of the starting player can be measured by his ability to learn the winning strategy.

An important question is the game situation coding. Obviously, we can describe each game situation on a grid of size $I \times J$ by a matrix of discrete variables taking three possible values: $y_{ij} = 1$ for the cross and $y_{ij} = 2$ for the nought and $y_{ij} = 0$ for an empty cell:

$$\boldsymbol{y} = [y_{ij}]_{i=1\ j=1}^{I\ J}, \quad \boldsymbol{y} \in \mathscr{Y} = \{0,1,2\}^{I\times J}. \tag{1}$$

In literature the size of the square grid is almost always restricted in order to avoid exceeding dimensionality of the related decision problems. Usually the standard game is played on a 19×19 board but sometimes even a small size of 3×3 cells is considered as a simple benchmarking problem (Furnkranz and Kubat, 2001; Michie, 1961; Blake and Merz, 1998). In our case we have chosen $I = J = 100$ but the board could be virtually infinite as we evaluate the game situations locally.

We assume that the usefulness of placing a marker on a free cell y_{ij} can be sufficiently analyzed and evaluated within a small window centered at the coordinates (i,j). In other words the move to be evaluated is described by the internal cells of the decision window and can be represented by a discrete vector $\boldsymbol{x}$ of a relatively low dimensionality.

In the following we assume a 7×7 decision window which is the smallest one to identify the winning moves. The related dimension of the vector $\boldsymbol{x}$ is $N = 49, (\mathscr{N} = \{1,\ldots,N\})$:

$$\boldsymbol{x}(i,j) = \boldsymbol{x} = (x_1,\ldots,x_N) \in \mathscr{X}, \quad \mathscr{X} = \{0,1,2\}^N.$$

In this notation the discrete variables x_n correspond to the 49 internal cells of the decision window in an arbitrarily chosen fixed order, e.g. left-to-right and top-to-down. Throughout the paper we use the term "move" also in connection