Pattern Recognition Neural Network on a Tic-Tac-Toe Game

Sukrit Gupta

Research School of Computer Science, Australian National University,
Canberra, Australia
u5900600@anu.edu.au

Abstract. A Pattern-recognition neural network applying Back propagation algorithm is used to classify the results of data set containing possible board configurations at the end of a tic-tac-toe game into binary classification of "Win for 'x' " or "not", where "x" is assumed to have played the first move. The data was passed through several topologies of the neural network (different number of layers and neurons) as experimentation to find the best suitable topology, 10 K-Fold cross validation was used to keep the results accurate. The evaluation of the model is based on the accuracy, precision, specificity and sensitivity comparing the results with the published research paper applying to the same data set [1]. The best results for the above criteria's were 79.96%, 72.01%, 85.78%, 69% respectively, compared to the research paper[1] using six different algorithms such as NewID, CN2, MBRtalk, IB1, IB3 and IB3-CI, Neural networks were not used in the research paper.

Keywords: Tic-Tac-Toe, Back propagation, K-Fold, Confusion Matrix

1 Introduction

In this paper we will be discussing the results of using a feed-forward single-layered neural network for pattern recognition on a Tic-Tac-Toe Endgame dataset. Various topologies were tested and investigated to find the one with the most desirable results in terms of the miss-classification errors obtained from each of them.

1.1 Background and Motivation

Neural networks are a biologically inspired models consisting of networks generally represented as a system of interconnected neurons comprising of simple processing units that act together over weighted connections. They were primarily inspired by the notions of how a biological brain might do its computations [2].

Despite its apparent simplicity, Tic-tac-toe requires detailed analysis to determine even some elementary combinatory facts, the most interesting of which are the number of possible games and the number of possible positions. A position is merely a state of the board, while a game usually refers to the way a terminal position is obtained [3].

1.2 Problem Modelled and Investigations Outline

The game of Tic-Tac-Toe ends when one symbol either “X” or “O” are obtained in a row of three. The classification in this paper is in a binary format of weather “X” wins or loses considering “X” always begins the game. The dataset encodes the complete set of possible board configurations at the end of Tic-Tac-Toe games [1]. In this experiment, 10-Kfold cross validation method was used for the selected classification algorithm i.e. Back-propagation.

Analysis: This paper concentrates on the performance analysis of the classification algorithm based on the observed Accuracy, Precision, Sensitivity and Specificity for the pattern recognition on the training and testing sets using the aggregated confusion matrix obtained at the end of the neural network run.

2 Method

The paper published by David W. Aha used a number of instance-based learning algorithms (IBL) on the selected Tic-Tac-Toe dataset to perform feature construction [1]. Artificial Neural Networks have been known to be applied for pattern classification, function approximation, optimization, pattern matching successfully [5], [6]. The purpose of the experiment conducted in this paper is to see how a pattern recognition neural network performs when using the Back-propagation algorithm. The target concept for this particular classification problem is to distinguish when the pattern indicates a "win for X" or a "loss for X" and classify the data into two classes.

2.1 Data Set

The problem dataset consists of 958 instances of legal Tic-Tac-Toe endgame boards with 9 attributes each, every attribute correspond to one Tic-Tac-Toe square [4]. The nine available places on the Tic-Tac-Toe board can be occupied by either "X", "O' or in some endgame cases "blank space". The 10th attribute classifies the data into two classes "Win for X" and "Loss for X". This dataset assumes that "X" made the first move in every game. The original dataset acquired from the UCI Machine Learning Repository had real-valued pattern attributes belonging to two output classes [7], for the experimentation conducted in this paper, the dataset has been modified to contain numerical values equivalent to the original real-valued pattern:

Table 1.

Table 1 shows the changes made to the dataset for the neural network experiment

Original Values	New Numerical Values
x	1
o	2
b	0
positive	1
negative	0

Table 2.

Data Set	Size(instances)	Inputs	Outputs	Attributes
Tic-Tac-Toe Endgame Data Set	958	9	2	Integer

The 9 available places on a Tic-Tac-Toe board correspond to the 9 attributes (inputs) and Class (output) in the dataset. They are represented below

— top-left-square:	{0,1,2}
— top-middle-square:	{0,1,2}
— top-right-square:	{0,1,2}
— middle-left-square:	{0,1,2}
— middle-middle-square:	{0,1,2}
— middle-right-square:	{0,1,2}
— bottom-left-square:	{0,1,2}
— bottom-middle-square:	{0,1,2}
— bottom-right-square:	{0,1,2}
— Class:	{0,1}

2.2 Model Design and Investigative Aims

The previous relevant papers working on this data set have used Constructive induction on decision trees [7], domain knowledge for feature construction [8] and incremental constructive induction for feature construction using IB3-CI [1].

The model implemented in this paper is a pattern recognition neural network that uses Back-propagation algorithm to do classification of the data set. Pattern recognition is implemented by using a feed-forward neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern [9]. A variety of neural network architectures were tried out and the final network topology chosen was the one that appeared best suited to the given dataset with the highest accuracy. A single layer with hidden neurons was employed to avoid over fitting and keep the computation time down.

Neural Networks are universal function approximators and with enough time and repetition can lead into a problem called “over-fitting”, to side step this issue “K-Fold” cross validation was used. The training data was divided into k subsets where k is 10. This process repeats 10 times and each time 9/10 of training data is used for training and 1/10 is used for testing. A confusion matrix is generated that describes the performance of this classification problem, the values present in this matrix are used to carry out further performance calculations. The main investigative aim was to find out the values Accuracy, Precision, Sensitivity and Specificity when a neural network is used to classify the dataset and compare them with the results obtained using a decision tree implementing various algorithms [1].

3 Results And Discussion

3.1 Performance Analysis

Performance of the selected dataset after using Back-propagation algorithm with K-fold cross validation on it was observed using Accuracy, Precision, Sensitivity and Specificity which can be defined as follows:

Accuracy: The accuracy of a classifier is the percentage of the test set tuples that are correctly classified by the classifier [7].

$$\frac{true\ positives+true\ negatives}{true\ positives+false\ positives+false\ negatives+true\ negatives}$$

Sensitivity: Sensitivity is also referred as True positive rate i.e. the proportion of positive tuples that are correctly identified [7].

$$\frac{true\ positives}{true\ positives+false\ negatives}$$

Precision: precision is defined as the proportion of the true positives against all the positive results (both true positives and false positives) [7].

$$\frac{true\ positives}{true\ positives+false\ positives}$$

Specificity: Specificity is the True negative rate that is the proportion of negative tuples that are correctly identified [7].

$$\frac{true\ negatives}{true\ negatives+false\ positives}$$

Table 3. Performance of backpropagation algorithm applied to a single-layered neural network

Algorithm	Accuracy	Precision	Sensitivity	Specificity
Back-Propagation	79.96±2%	72.01±2%	85.78±2%	68.98±2%

The above results show the effectiveness of neural network in classifying the dataset, these results are compared with the published papers that use decision tree with instance based learning algorithm to perform feature construction for attaining higher predictive accuracies [1], it used 70% for training, 30% of the instances for testing, and evaluated over 10 trials. Results reported for six algorithms:

–	NewID:	84.0%
–	CN2:	98.1%
–	MBRtalk:	88.4%
–	IB1:	98.1%
–	IB3:	82.0%
–	IB3-CI:	99.1%

As these results compared to the outcomes obtained using neural network demonstrate that using back-propagation algorithm might not be as effective as a decision tree using IB3-CI instance based algorithm but is better than CITRE that performs feature construction using decision trees and simple domain knowledge as constructive biases, whose best result was 76.7% $\underline{+}$ 2.7 [7]

4 Conclusion and Future Work

This paper implemented a Feed-Forward Neural Network using back-propagation algorithm to classify the Tic-Tac-Toe endgame dataset. Generally, Neural Networks are known to be successful in pattern recognition, in this experiment it recorded mediocre to high accuracy and was able to achieve better results than domain knowledge based CITRE [7] but was poor in comparison to instance based algorithm like IB3-CI [1]. However, its current level of performance provides evidence that additional modifications are well worth investigating.

For Future work other algorithms like K-Nearest Neighbor, Support Vector Machines (SVM) algorithm or C4.5 algorithm could be used to investigate and report results after their application to this dataset and compare results to see if any improvements are visible.

5 The References Section

[1] Aha, D. W. (1991). Incremental constructive induction: An instance-based approach. In Proceedings of the Eighth International Workshop on Machine Learning (pp. 117--121). Evanston, ILL: Morgan Kaufmann.

[2] Bishop. C. M. (1995). Neural Networks for Pattern Recognition. Oxford: Oxford University Press

[3] "Tic-Tac-Toe". *Wikipedia*. N.p., 2016.

[4] Aha, David W. *Archive.ics.uci.edu*. N.p., 2016.

[5] J. Dayhoff, Neural-Network Architectures: An Introduction. New York: Van Nostrand Reinhold, 1990.

[6] K. Mehrotra, C. Mohan, and S. Ranka, Elements of Artificial Neural Networks. Cambridge, MA: MIT Press, 1997

[7] Matheus, C.J., & Rendell, L.A. (1989). Constructive induction on decision trees. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. (pp. 645--650). Detroit, MI: Morgan Kaufmann.

[8] Matheus, C.J. (1990). Adding domain knowledge to SBL through feature construction. In Proceedings of the Eighth National Conference on Artificial Intelligence (pp. 803--808). Boston, MA: AAAI Press.

[9] Stergiou, Christos and Siganos, Dimitrios. "Neural Networks". *Doc.ic.ac.uk*. N.p., 2016.