APPROXIMATION OF CONDITIONAL PROBABILITY FUNCTION USING ARTIFICIAL NEURAL NETWORKS

Alexey Vasilyev and Andrei Kapishnikov

Riga Technical University, 1 Kalku Street, LV-1658 Riga, Latvia,
http://dssg.cs.rtu.lv
servent@apollo.lv, sv70019@lanet.lv

This work shows that a multilayer perceptron can successfully be applied for approximation of conditional probability function. Such an approach makes it possible to examine the training sample with contradictory vectors and to interpret the outputs of the network in classification tasks as the evaluation of the probability of belonging to the specific class. This method was used for creating the adaptive agent for RoboCup simulation environment.

1 Introduction

Many tasks of pattern recognition require the use and manipulation of probabilistic data. Therefore, it is very often necessary not only to classify patterns, but also to find probability of pattern belonging to a specific class, i.e. it is necessary to obtain the distribution function of conditional probability. To solve this problem, a neural network can be used, typical representative of which is multilayer perceptron [1]. Because of the fact that the perceptron training is based on the minimization of the error, its output can be viewed as (1) estimation of probability, which is (2) approximated by the perceptron as a result of training. A good result can be achieved due to the artificial neural network (ANN) approximation abilities.

2 Probability estimation

The way of interpretation of ANN outputs as probability estimation can be demonstrated by the following example. Let us assume that there are training vectors in a form: $v = \{x_1,\ x_2,\ y\}$, where x_1, x_2 – input parameters, y – class to which the vector belongs.

$v_1 = \{0,\ 1,\ \mathrm{A}\}$, $v_2 = \{0,\ 1,\ \mathrm{A}\}$, $v_3 = \{0,\ 1,\ \mathrm{B}\}$, $v_4 = \{1,\ 0,\ \mathrm{A}\}$, $v_5 = \{1,\ 0,\ \mathrm{B}\}$

It can be seen, that vectors v_1, v_2 and v_3 have the same input parameters, however, vectors v_1, and v_2 belong to class A, while vector v_3 belongs to B. As well vectors v_4 and v_5, having the equal input parameters, belong to different classes. It is obvious, that the probability of belonging to class A, when $x_1 = 0$, $x_2 = 1$, is equal to 2/3. Let's denote it as $p_{01}(\mathrm{A}) = 2/3$. $p_{01}(\mathrm{B}) = 1/3$, $p_{10}(\mathrm{A}) = 1/2$, $p_{10}(\mathrm{B}) = 1/2$ accordingly.

Let us consider now an ANN with the continuous activation function, whose normalized output is in the range [0..1]. Let values '1' and '0' be assigned, respectively to classes A and B during the training phase. After the training process based on the above mentioned vectors, the network will generate value 0.66 if input $x_1 = 0$, $x_2 = 1$ is fed. This value corresponds to the probability that the input vector belongs to class A. It is obvious that input $x_1 = 1$, $x_2 = 0$ will give output value equal to 0.5. This is true thanks to ANN training principle of minimizing square of error e between desired inputs and obtained ones. In the case if more general training set were given, the ANN outputs will not be equal to conditional probability precisely; however, the precision can be acceptable in many practical applications. Moreover, an ANN can successfully approximate probability function and therefore to give probability estimation even if inputs are given, which were not included into training set.

3 Approximation of probability function

It is well known, that a feedforward neural network, for example a multilayer perceptron, with one hidden layer of neurons can represent any continuous function exactly ("Kolmogorov mapping neural network existence theorem") [2]. This property of multilayer perceptron allows considering it as an approximator of conditional probability function. That can be demonstrated by the following example:

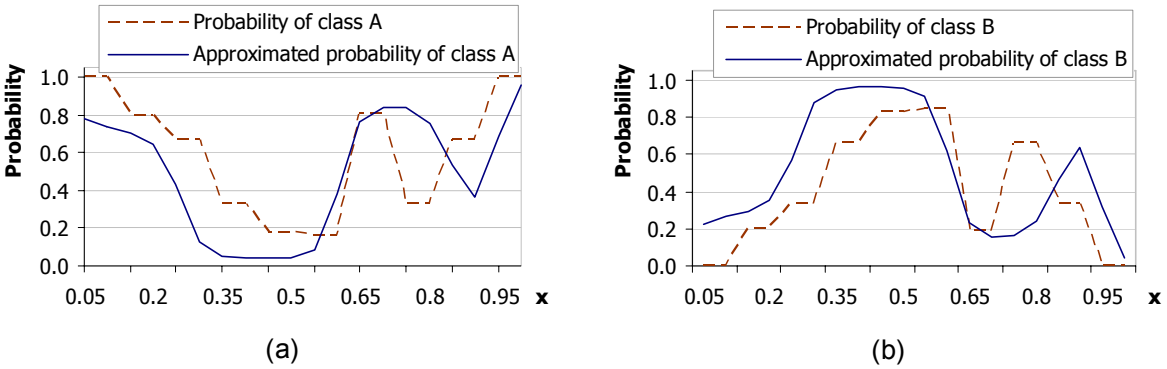

Fig. 1. Probability function of class A (a) and class B (b) and their approximations with the neural network

Let us assume that there are vectors $v = \{x_1, y\}$, where y – class A or B. The function of conditional probability, which depends on x_1, is shown in Fig. 1. *a* and *b* using the dotted line for class A and B respectively. The ANN with 1-50-1 architecture (*i.e.* 1 input, 50 neurons in the hidden layer and 1 output) was used for the approximation. The result obtained after the training process is also shown in Fig. 1. *a* and *b* by the solid line. It is obvious that with an increase of neurons in the hidden layer, the approximated function will approach the real probability function. In order to show a practical application of the described approach, we applied it to the RoboCup testbed.

4 Approximation of probability function solving a real problem

A huge number of states $(108\times50)^{23}$ and possible policies 300^{22} within one step makes the RoboCup simulation environment [3] extremely popular for different heuristic learning systems. RoboCup simulation server is a multi-agent environment that supports 24 independent agents (22 players and 2 online coaches) interacting in a dynamic, real-time environment. The server implements many real-time complexities, such as noisy, limited sensing; noisy action and object movement; limited agent stamina; limited inter-agent communication bandwidth; different types of agents with different parameters. The goal of the agents is to win a football match.

The method of probability estimation on the basis of ANN described above was employed to create adaptive program agents for RoboCup simulation environment. The behaviour of the agents can be divided into sub-assignments and formulated in the form of rules. For example, the rule of the simplified behaviour of the agent when it has the ball can look as follows:

IF a probability to score a goal is high THEN kick the ball into the goal
ELSE IF a probability to make a leading pass is high THEN make the leading pass
ELSE dribble with the ball towards the goal

It is obvious that in this case the creation of modules, which would evaluate the probability of scoring the goal as well as probability of leading pass success, is necessary. Then tuning the thresholds for decision-making is necessary. For these purposes, the multilayer perceptron was used, which approximated the function of conditional probability.

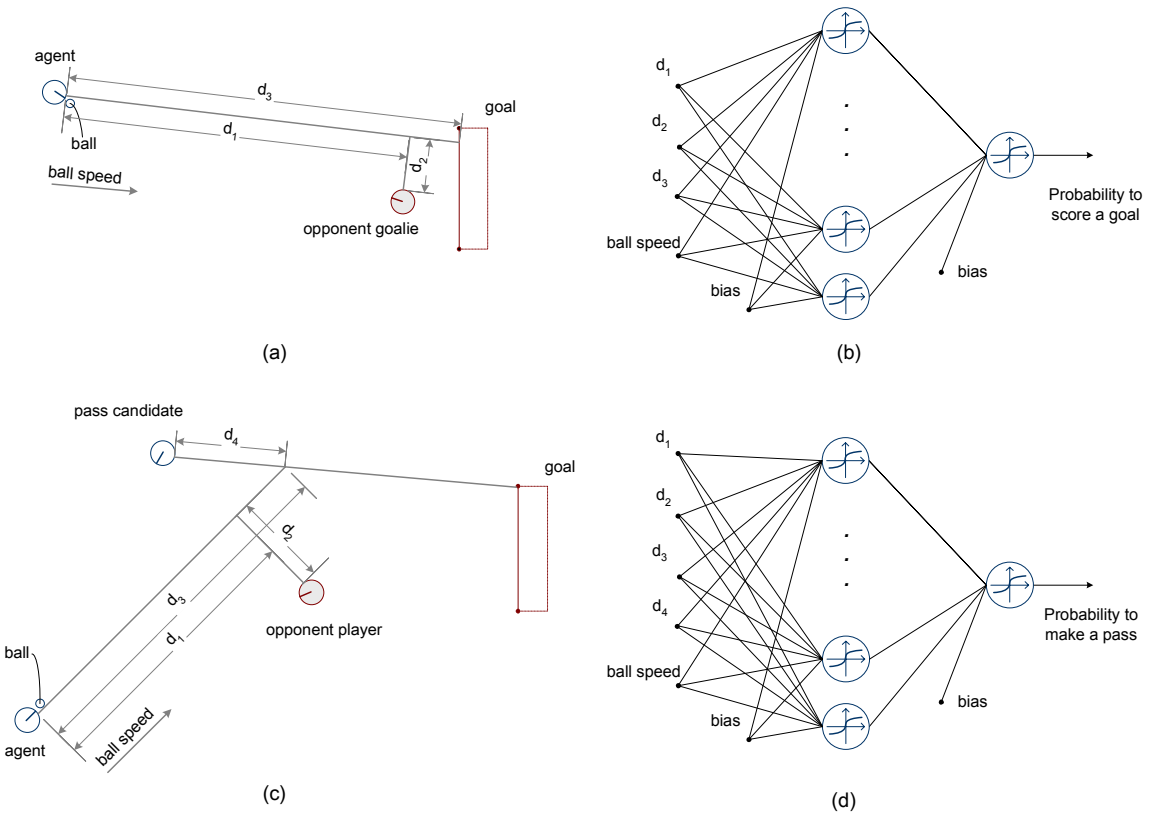

Fig. 2. Probability calculation for (a) goal shooting, (b) neural network for goal shooting, (c) leading pass, (d) neural network for leading pass

For calculating the probability of scoring the ball (Fig. 2.b), a multilayer perceptron of form 4-15-1 (4 inputs, 15 hidden neurons and 1 output) was used. The input parameters used by the ANN are depicted in Fig. 2.*a*. The error back-propagation training algorithm was used during the training phase with training coefficient c = 0.01 and training momentum m = 0.9. For acceleration the convergence of the training algorithm, the procedure of Nguyen-Widrow was used. This procedure helps to select initial weights of synapses for the hidden layer (see for example [4, p.297]). The training was carried out based on 3500 examples. The sum-squared error was equal to 0.116 within the training set after the 500th iteration. The testing set contained 500 examples. The selected level of threshold for decision-making equal to 0.8 has given the error level equal to 0.73%, *i.e.* only 1 of 100 of balls kicked into the goal were intercepted by the opponent's goalie.

The probability of success of a pass was estimated using a 5-20-1 ANN. (Fig. 1. *c*, *d*), c = 0.01, m = 0.9. The whole set was divided into 2000 training examples and 500 testing examples. The sum-squared error was equal to 0.28 within the training set after the training phase was completed. When the decision-making threshold was set to 0.8, the number or erroneous decisions were 3.35% within the testing set. Setting the threshold to 0.9 decreases the error up to 0.91%.

The above-described approach enables achieving highly reliable results. In addition, changing the threshold for decision-making allows tuning the agent behaviour depending on opponents' actions during the game [5].

5 Conclusion

In this work, it has been shown that multilayer perceptron can be successfully applied for estimating conditional probability when source data contains contradictory examples. The approach has shown a high efficiency level even if the data, which were not within inputs of the training set, are fed. The high level of successful decisions, made by agents of RoboCup simulation environment, demonstrates that fact. Thus, the error level was less than one percent when the agent made a decision to strike the ball into the goal, and about 3% when decision to make a leading pass was carried out.

The similar results for conditional probability estimation and approximation can be achieved by using probabilistic neural network [6], however its size depends on the number of the samples in training set, which is

not true in the case of multilayer perceptron. In many cases the use of multilayer perceptron will give a more compact representation of conditional probability function.

References

[1] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. Learning internal representations by error propagation. In Rumelhart, D.E. and McClelland, J. L., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1, pp. 318-362, Cambridge, MA: The MIT Press, 1986.

[2] Hecht-Nielsen, R. Theory of the Backpropagation Neural Network, *International Joint Conference on Neural Networks*, *Washington*, *DC*, 1-593:605.

[3] Mao Chen, Klaus Dorer, Ehsan Foroughi, Fredrik Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Jan Murray, Itsuki Noda, Oliver Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. RoboCup Soccer Server User Manual: for Soccer Server version 7.07 and later, 2003. At http://sourceforge.net/projects/sserver.

[4] Fausett L. Fundamentals of Neural Networks Architectures Algorithms and Applications. Prentice Hall International Inc., 1994. ISBN: 0-13-334186-0.

[5] Sebastian Buck, Martin A. Riedmiller: Learning Situation Dependent Success Rates of Actions in a RoboCup Scenario. PRICAI 2000: 809.

[6] Specht D.F., Probabilistic Neural Networks. Neural Networks, 3, p.109-118, 1990.