World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:2, 2015

A Combined Neural Network Approach to Soccer Player Prediction

Wenbin Zhang, Hantian Wu, Jian Tang

Abstract—An artificial neural network is a mathematical model inspired by biological neural networks. There are several kinds of neural networks and they are widely used in many areas, such as: prediction, detection, and classification. Meanwhile, in day to day life, people always have to make many difficult decisions. For example, the coach of a soccer club has to decide which offensive player to be selected to play in a certain game. This work describes a novel Neural Network using a combination of the General Regression Neural Network and the Probabilistic Neural Networks to help a soccer coach make an informed decision.

Keywords—General Regression Neural Network, Probabilistic Neural Networks, Neural function.

I. Introduction

A Neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to do computation. Neural networks are designed for modeling complex relationships between inputs and outputs[1], and they have been a fertile field of research and application. Rani and Mahip give an excellent review and many examples of neural networks[2]. It has become commonplace for a soccer team with more and more staffs to acquire all kinds of players' information to help coach determines the best lineup. The coach has to consider many factors to make this decision, and it is not always easy to make a proper decision that the selected player has the highest probability to perform better in all candidate offensive players. However, to my best knowledge, there is no work for player selection in real soccer teams to date, although in the certain soccer areas, neural networks are given a wide range of applications, including robot soccer[3], soccer match result prediction[4], soccer videos classification[5] and so on. Inspired by this practical need and the function of neural network, we use the idea of neural network to build a predictor for football player selection. More specifically, we combine the General Regression Neural Network and the Probabilistic Neural Networks to build this predictor. The player selected should have the largest probability to perform on a higher level in a certain position. It is anticipated that, given a particular game, it will be helpful to select players for different positions according to a new approach.

Wenbin Zhang is with the Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, A1B 3X5 Canada (e-mail: wenbin.zhang@mun.ca).

Hantian Wu is with the Department of Computer Science, Illinois Institute of Technology

Jian Tang is with the Department of Computer Science, Memorial University of Newfoundland

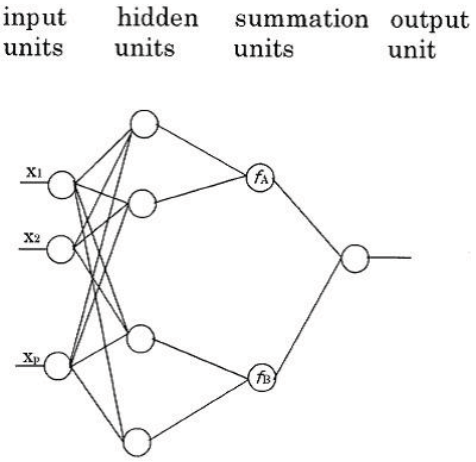

Fig. 1. The structure of Probabilistic Neural Network

II. Related Work

A. Probabilistic Neural Network

The Probabilistic Neural Network(PNN) is a feed forward neural network, which was firstly introduced by D.F. Specht in the early 1990s. The PNN is a multilayer feed-forward network with four layers. The structure of PNN is shown in Fig. 1.

Each neuron in the input layer represents an input variable, and it is connected with every neuron in the hidden layer. The hidden layer contains one neuron for each sample in the training data set. It stores the values of the input variables for the case along with the sample value[1], [6]. A hidden neuron computes how similar (the Euclidean distance) the input is to the corresponding training sample. The closer it is, the more likely it is to return a higher value which represents that this input is more likely belonging to the same class with the training sample. Each node in this layer only connects with the node representing the corresponding class in the summation layer. Each node in summation layer represents one class in the classification, and adds the values received from the hidden layer. The output layer compares the probabilities for each target class accumulated in the pattern layer and uses the largest probability to predict the target class[7], [8].

If the probability density function(PDF) of each of the populations(classes) is known, then an unknown input, X, belongs to class i if: $h_i c_i f_i(x) > h_j c_j f_j(x), j \neq i$. Here the $f(x)$ is the probability density function, h is the prior probability, which is the probability of an unknown sample being drawn from a particular population, and c is the misclassification cost, indicating the cost of incorrectly

classifying an unknown input[9]. In this work, we do not take into account the prior probability or the misclassification cost. Whats more, it is not practical to find out the exact probability density function, so the reasonable solution is estimating it from the training sample. For a single sample in the training set, the PDF can be estimated as $\frac{1}{\sigma}W(\frac{x-x_k}{\sigma})$, where the x is the input, W is the weighting function, x_k is the k-th sample and σ is the smoothing parameter. The PDF for a single class becomes $\frac{1}{n\sigma}\Sigma_{k=1}^{n}W(\frac{x-x_k}{\sigma})$[10]. The estimated PDF approaches the true PDF as the training set size increases, as long as the true PDF is smooth. It is important to select a proper weighting function, which should have the following properties: it should return a large value when the distance between unknown input and the training sample is small; and the returned value rapidly decreases to zero as the distance increases[6], [11]. Commonly the Gaussian function is a good choice because it behaves well, is easily computed and is not related to any assumption about a normal distribution[12]. Then the estimated PDF function for each class becomes:

$$g_i(x) = \frac{1}{(2\pi)^{p/2}\sigma^p n_i}\Sigma_{k=1}^{n_i} e^{-\frac{\|x-x_{ik}\|^2}{2\sigma^2}} \quad (1)$$

For the PNN, the training procedure is equal to build the network and store the information of samples in the hidden layer. In other words, after building the neural network, there is no training procedure. Therefore, it is quite fast. Besides this, the training samples can be added or removed easily without extensive retraining. Another important fact is that a proper smoothing parameter σ is required. It can be determined by an educated guess based on knowledge of the data or estimated by a heuristic technique.

B. General Regression Neural Network

The General Regression Neural Network(GRNN) was also introduced by D.F. Specht in the early 1990s and it is based on probability as well. GRNN is very similar to PNN, especially for the structure. However, the GRNN is designed for regression, while PNN is designed for classification. Given the training data, the GRNN can reconstruct the underlying function, $f(x)$. The architecture of GRNN is very similar to the architecture of PNN, except the neuron in the hidden layer is connected with every neuron in the summation layer and the edges that connect the hidden layer and summation layer are assigned weights[13], [14], [15], as shown in Fig. 2.

As a reasonable estimate of an unknown regression function $f(x)$, relying on a prior belief of its smoothness, one may take a mean of observations from a neighborhood of the point x. This approach is successful if the local average is confined to observations in a small neighborhood(i.e. a receptive field) of the point x as observations corresponding to points away from x will generally have different mean values[16]. More precisely, $f(x)$ is equal to the conditional mean of z given x. Using the formula for the expectation of a random variable, it can be written: $f(x) = E[z|x] = \int_{-\infty}^{+\infty} zp_z(z|x)dz$ where $p(z|x)$ is the conditional probability density function of z given x. From probability theory, it is known that: $p_z(z|x) = \frac{p_{x,z}(x,z)}{p_x(x)}$, then the formula comes to:

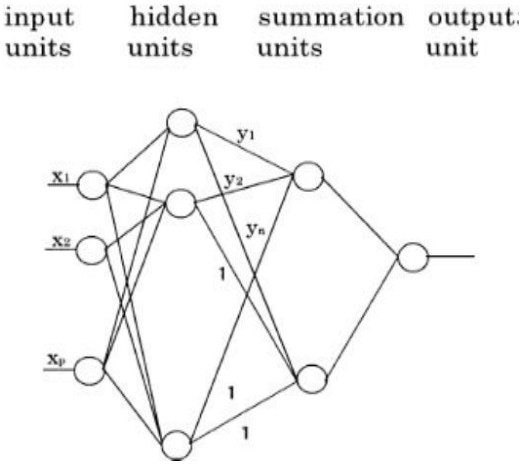

Fig. 2. The structure of General Regression Neural Network

$f(x) = E[z|x] = \frac{\int_{-\infty}^{+\infty} z(x)p(x|z)dz}{\int_{-\infty}^{+\infty} p(x|z)dz}$, where $p(x,z)$ is the joint distribution function[13].

We can use either parametric or nonparametric models to find the joint distribution function from the training data. The latter one is always a preferred choice because it estimates the density directly from the data without making any parametric assumptions about the underlying distribution. The Parzen-Rosenblatt density estimator of joint PDF is adopted, where $p(x,z)$ in the joint input-output space is defined as: $p_{x,z}(x,z) = \frac{1}{\sigma^{p+1}N}\sum_{i=1}^{N} K(\frac{x-x_i}{\sigma})K(\frac{z-z_i}{\sigma})$. Here the K is kernel function, always, Gaussian function is a good choice for the kernel function[14]. After applying the Gaussian function, we can get (2) and (3). Then assessing the two indicated integrations and using $\int_{+\infty}^{-\infty} ze^{-z^2}dz = 0$, we yield(4).

$$p(x,z) = \frac{1}{(2\pi)^{3/2}}\frac{1}{N}\Sigma_{n=1}^{N}\frac{1}{\sigma_{x,n}^2\sigma_z}\exp\{-\frac{D_{x,n}^2}{2\sigma_{x,n}^2} - \frac{D_{z,n}^2}{2\sigma_z^2}\}, \quad (2)$$

$$f(x) = \frac{\Sigma_{i=1}^{n}\exp[-\frac{(x-x_i)^T(x-x_i)}{2\sigma^2}]\int_{-\infty}^{+\infty} z\exp[-\frac{(z-z_i)^2}{2\sigma^2}]dz}{\Sigma_{i=1}^{n}\exp[-\frac{(x-x_i)^T(x-x_i)}{2\sigma^2}]\int_{-\infty}^{+\infty}\exp[-\frac{(z-z_i)^2}{2\sigma^2}]dz}, \quad (3)$$

$$f(x) = \frac{\Sigma_{i=1}^{n} z_i\exp[-\frac{(x-x_i)^T(x-x_i)}{2\sigma^2}]}{\Sigma_{i=1}^{n}\exp[-\frac{(x-x_i)^T(x-x_i)}{2\sigma^2}]}. \quad (4)$$

Formula (4) is a weighted sum over all the training patterns. Each training pattern is weighted exponentially according to its Euclidean distance to the unknown pattern x and also according to the smoothing factors σ. Note that the use of the normalized Gaussian function does not imply any normal assumption about the distribution of the data in the feature space[15]. The Gaussian function is used just for the reason that it satisfies the requirements about the properties of the kernel function. The σ is the smoothing factor, which leads to different estimation results: if σ is too small, the result will become over-fitting; if too large, the estimation will be over smoothing.

The final formula is mapped into the GRNN. Each node in the hidden layer represents a training pattern. The summation

layer includes two nodes: the first node sums all the outputs of the hidden layer and evaluates the numerator of the final formula, and the second unit evaluates the denominator of the final formula. Each unit in the hidden layer is connected to each of the two nodes in the summation layer. The weights of the connection between the node i in the hidden layer and the first unit of the summation layer is equal to y_i, the target value. The i weight of the connection between any node i in the hidden layer and the second node in the summation layer is equal to unity. The output node merely divides the two outputs of the summation layer to yield the predicted value of the dependent feature[17], [18]. The advantages of GRNN are fast training and the ability to easily add or delete training samples.

III. A Combined Approach

A. Methodology

Considering the similar architecture of PNN and GRNN as well as fast training and re-training features of both neural networks, we are going to construct a neural network by combining PNN and GRNN to predict soccer players' performance value. Applications using the combination of neural networks have been discussed in some literatures[19], [20], [21].

In reality, many factors will affect a game. We choose some key and useful factors: time, opponent, the opponent's playing style, city, weather, difficulty and player form[22]. The target value is the players performance level. Other factors can be easily included by adding corresponding input nodes. As there are several candidate players, we need to calculate the most probable performance for every player, and choose the maximum one as the final choice. This results in one more layer to compare the estimated performances of players. The GRNN part will use the time information and performance information (target value) in historical data to estimate the underlying function between time input and performance value. When an input game comes, the GRNN will use the input time factor and estimate the target performance value for each candidate player. Note that each candidate player has a different underlying function. Other factors will be used by the PNN. The target value–performance will be discretized into several performance classes. In the PNN, the Gaussian function is still used but not enough because there are several factors that cannot be digitized and applied in Gaussian function, such as: city, opponent. How to solve this problem? The idea of PNN is if input is closer, namely more similar to one training sample, it has higher probability of belonging to the corresponding class. For example, if there are two samples for player1: $(ATM, offensive, sunny, home, 9.0)$ and $(Betis, defensive, cloudy, away, 7.0)$, where the last value is the target performance level, assuming each variable has the same weight, then when an input $(Betis, defensive, sunny, away)$ comes, using the idea of PNN, the node representing the second sample will return a higher value than the first one. So we design our own neural function for our PNN which ensures that return a large value when the distance between unknown input and the training

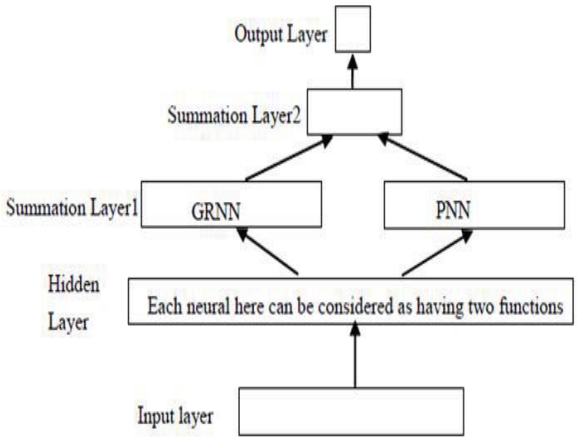

Fig. 3. The structure of combined Neural Network

sample is small; the return value rapidly decreases to zero as the distance increases. Using this idea, in our neural function, if one variable in the city is the same, it will add a value to its output, if not, it will not. For the numeric variables (game difficulty and player form), the Gaussian function still applies. In this way, our neural function formula becomes:

$$\begin{aligned} SUM &= 0.15 * (opponent) + 0.15 * (style) \\ &+ 0.1 * (city) + 0.05 * (weather) \\ &+ 0.55 * Gaussian(difficulty, playerform). \end{aligned} \tag{5}$$

Here, if the input of opponent is same as the sample, the opponent value will be 1, otherwise the value will become 0. Other factors work in the same way. Then, for one input, the GRNN and PNN will return estimated performance values for each candidate offensive player. The added layer will add these two estimated values for each candidate and send to the output. The output layer will do comparison between players, then output the selection and estimated performance for this player.

B. Structure

Fig. 3 is a schematic diagram. The first layer is the input layer. The number of the nodes in this layer is equal to the number of factors. For the hidden layer, each node stores the information of the sample and it can be considered as there are two neural functions: one uses time and historic target value to do regression, another uses other factors to do classification. In the summation layer1, there are two parts: the node in the left part sums the values calculated by regression function in hidden layer for each candidate; the node in the right part sums the values calculated by classification function in hidden layer for each candidate. Then the summation 2 layer does division for regression and comparison for the classification. This layer calculates the estimated performance and the most probable performance for each player. Finally, the output layer can make the selection and output the average of the sum of

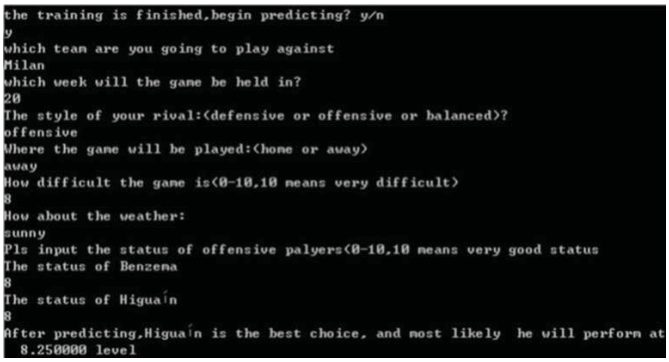

Fig. 4. Selecting player for Real Madrid when against AC Milan

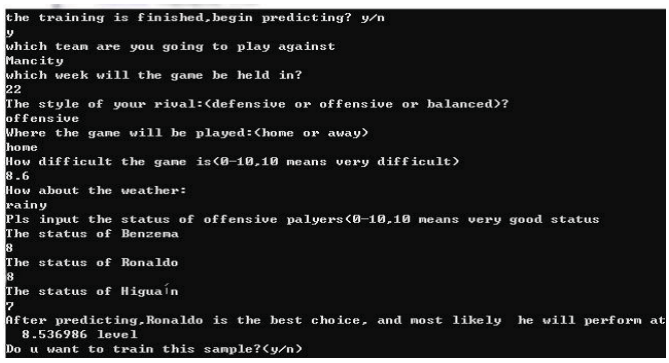

Fig. 5. Selecting player for Real Madrid when against Mancity

estimated performance and the most probable performance as the final performance. The smoothing factors for PNN and GRNN we use are 1.1 and 1.85 respectively.

IV. Empirical Study

One predicting process based on an UEFA Champions League for Real Madrid to select player when against AC Milan on November, 4^{th}, 2011 is presented in figure 4. Time factor uses week as unit time; the name of the opponent along with the style of this opponent are required; the place factor is either home or away; difficulty factor range from 0 to 10 with higher value means more difficulty; then are the weather and the form of the player which also range from 0-10, and higher values reflects better status. The performance is the target value, from 0-10, and still higher value means better performance.

After entering the factors for a certain match, it will return the recommended player and estimated performance. The selected player Higuain scored the first goal for Real Madrid and was rated 8 out of 10 for this game by UEFA.com[22]. Figure 5 is another test result for a different Real Madrid's game against Mancity on September, 18^{th}, 2012. The selected player Ronaldo striked in the final seconds earned Real Madrid 3: 2 against Mancity and scored 9 out of 10 for this game by UEFA.com[23]. After the predicting process, the user can either add or delete the training samples.

The above experimental results suggest that predictor is practical for helping soccer coach to determine the most suitable player after taking different factors into consideration.

V. Conclusion and Discussion

In this paper, we propose a novel Neural Network which combines the General Regression Neural Network and the Probabilistic Neural Networks to help a soccer coach select players who have the largest probability to perform on a higher level. The novel Neural Network is implemented and achieves simplicity and promising finite sample performance in our empirical studies. Our future work will pay more attention to the selection of smoothing factors for neural functions in a more technical method for this problem. Also, we will extend this Neural Network to be applicable to other matters in daily life.

References

[1] D.F.Specht: Probabilistic Neural Networks. In: Neural Network, Vol. 3, pp. 109-118 (1990)
[2] Rani Pagariya, Mahip Bartere: Review Paper on Artificial Neural Networks. In: International Journal of Advanced Research in Computer Science, Volume 4 (2013)
[3] Jolly, K.G, Ravindran, K.P, Vijayakumar, R.: Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks. In: Robotics and Autonomous Systems, Volume 55, Issue 7 (2007)
[4] Heuer Andreas, Rubner Oliver: Towards the perfect prediction of soccer matches. In: Physics.data-an (2012)
[5] Ballan. L, Bazzica. A, Bertini, M: Deep networks for audio event classification in soccer videos. In: IEEE International Conference on Multimedia and Expo (2009)
[6] Shaffer, R. E.; Rose-Pehrsson, S. L.: Improved Probabilistic Neural Network Algorithm for Chemical Sensor Array. In: Pattern Recognition. Anal. Chem., 71, 4263-4271 (1999)
[7] D.F.Specht: Probabilistic neural networks for classification, or associative memory. In: IEEE international conference on neural networks, San Diego, vol 1, pp 525535 (1988)
[8] D.F.Specht, Shapiro PD: Generalization accuracy of probabilistic neural networks compared with back-propagation networks. In: International joint conference on neural networks Seattle, vol 1, pp 887892 (1991)
[9] Cacoullos, T.: Estimation of a multivariate density. In: Annuals of the Institute of Statistical Mathematics, 18(2): 179-189 (1966)
[10] D.F.Specht: Enhancements to probabilistic neural networks. In: International joint conference on neural networks Baltimore, vol 1, pp 761768 (1992)
[11] Adeli H, Panakkat A: A probabilistic neural network for earthquake magnitude prediction. In: Neural Network 22(7): 10181024 (2009)
[12] J. Tetteh, A. Beezer, J. Orchard, C. Tortoe: Application of Radial Basis Function Network with a Gaussian Function of Artificial Neural Networks in Osmo-dehydration of Plant Materials. In: Journal of Artificial Intelligence, Volume 4 (2011)
[13] D.F.Specht: A general Regression Neural Network. In: IEEE Transactions on Neural Networks, Vol 2, pp 568576 (1991)
[14] Marchctte, D., Priebe, C.: An application of neural networks to a data fusion problem. In: TriServiceData Fusion Symposium 1, 23tl-235 (1987)
[15] Cichocki A and Zdunek R: Multilayer nonnegative matrix factorization using projected gradient approaches. In: International Journal of Neural Systems 17(6): 431446 (2007)
[16] Mikhail Kanevski: Machine Learning Algorithms: Theory, Applications and Software Tools (2009)
[17] Oscar TP.: Predictive model for survival and growth of Salmonella Typhimurium DT104 on chicken skin during temperature abuse. In: Poultry science, 72:30414 (2009)
[18] I-Cheng Yeh, Kuan-Cheng Lin: Supervised Learning Probabilistic Neural Networks. In: Neural Processing Letters, Volume 34, Issue 2 (2011)
[19] Yangpo Song, Xiaoqi Peng: Modeling method using combined artificial neural network. In: International Journal of Computational Intelligence and Applications, Volume 10, Issue 2 (2011)
[20] Raghu P.P, Poongodi R, Yegnanarayana B: A combined neural network approach for texture classification. In: Neural Networks, Volume 8, Issue 6 (1995)
[21] Chang Wei-Der: Recurrent neural network modeling combined with bilinear model structure. In: Neural Computing and Applications, Volume 24, Issue 3 (2014)

[22] Juergen Perl: Neural Network-Based Process Analysis in Sport Gaming and Simulations: Concepts, Methodologies, Tools and Applications (2011)
[23] The official website for European football http://www.uefa.com/uefachampionsleague/history/index.html

Wenbin Zhang is currently doing his graduate study at the Memorial University of Newfoundland with degrees in Computer Science. His interests lie in the fields of machine learning with specialization in designing variable selection algorithm in mixture cure model.

Hantian Wu is currently pursuing his master degree of Computer Science at Illinois Institute of Technology. His interests are in the fields of machine learning and mobile application developments.

Jian Tang is currently a professor of Department of Computer Science at the Memorial University in Canada. His research interests cover various topics in the broad area of data mining, database systems, distributed computing, workflow management, E-commerce, XML