PREDICTING POKER HANDS WITH ARTIFICIAL NEURAL NETWORKS

Gökay DİŞKEN, Çukurova Üniversitesi,Adana

Project goal: Aim of this project is to classify five cards drawn from a standard deck of 52 by suits and ranks of each card. After determining each cards value, system measures strength of the hand according to the poker game. Data for training and testing the network is from the UCI archive [1]. Training data has over 25000 samples and 11 attributes. 5 attributes are for suits of 5 cards, 5 attributes are for ranks of 5 cards and 1 attribute for the strength of poker hand. Sample size may be reduced to get rid of long computation time. With proper adjustment of the neural network system (e.g. correct number of inputs and outputs, sufficient number of nodes in hidden layer, transfer functions), the goal is to get high efficient classifying results.

The strength of the poker hand, which will be the output of the system, is defined by numbers between 0-9 in the original data set. In order to train the system properly, the output will be redefined by 10 elements, one for each strength level. For an input, the desired output will be 1 and other 9 outputs will be all zero. The details of the poker hands are described below. According to the data set, we have 12493 zeros, 10599 ones, 1206 twos, 512 threes, 93 fours, 54 fives, 35 sixes, 6 sevens, 5 eights and 5 nines.

I. Basics of Poker Game

Poker is a family of card games involving betting and individual play, whereby the winner is determined by the ranks and combinations of their cards, some of which remain hidden until the end of the game [2]. There are various types of the game, each of having different procedures. Some popular types are Texas Hold'em, Omaha, 7-card stud, 5-card draw. Since this project is not interested in actions like betting, folding and calling, we will only deal with the names of poker hands which, in fact, will be the outputs of the network system. To put it simply, the strength of hand is increased by including multiple cards of the same rank or by all five cards being from the same suit or by the five cards forming a consecutive series. The Ace can be the lowest or the highest ranked card in series (the lowest if the series is A-2-3-4-5, the highest if the series is 10-J-Q-K-A). With this information, we can form ten different levels of strength. From the weakest to most powerful hand names are "high card", "one pair", "two pairs", "three of a kind", "straight", "flush", "full house", "four of a kind", "straight flus", "royal flush". High card means there is no multiple cards in the hand, value of the hand is the highest ranked card. One pair means, there is one pair of equal ranked cards. Two pairs mean we have two pairs of equal ranks. Three of a kind is three equal ranks. Straight means five cards are sequentially ranked without a gap. Flush is the name of a hand consisting five cards with the same suit. Full house is a pair with three of a kind (ranks of the pair and other three are different). Four of a kind is four equal ranks. Straight flush is combination of straight and flush. Royal flush is the always winning hand with Ace, King, Queen, Jack, Ten with flush. The neural network system is going to classify the hand according to the data, then give one of the classes above as a result, depending on the cards ranks and suits.

II. Overview of Methods

Cattral et. al., used a similar dataset for data mining [3]. They used a data mining system (RAGA-Rule Acquisition with a Genetic Algorithm) that uses a hybrid genetic algorithm and genetic programming based engine. It was designed for both supervised and unsupervised learning. Poker data set is chosen because of its solution space is bounded. The poker hand scoring system model is the same as the video poker, which has more ranks than standard poker. They test the system and compared with another data mining system See-5 [4]. First a set of classifiers decision tree for See-5 and a rule hierarchy for RAGA was generated from a sample set of 10000 data. Then several test sets of 1000 data were used test the classifiers. Training accuracy of 64.25% was obtained for See-5. After testing the average correctness was only 36.16%. For the same dataset RAGA achieves 90.39% correctness on the training set and an average accuracy of 57.6% over all test sets. The authors concluded that RAGA does not rely on the default hierarchy to boost the predictive accuracy.

Another work with the same data is made by Nikola Ivanic [5]. The objective of this work is the same with the projects goal, that is, to train the neural network to predict which poker hand do we have based on cards given as input attributes. The system is designed with the aid of a software called Neuroph Studio. Because of software limitation 1003 instances of data is used. Because each one of the five cards in the hand has its own suit (set of 4 values) and rank (set of 13 values), the system has 5x(4+13)=85 inputs. 9 outputs are used for describing the poker hand (high carded hand is accepted as nothing in the hand for this work). Supervised training type is preferred to minimize the error of prediction through an iterative procedure. The transfer function is selected sigmoid function because the range of the data is 0-1, if it was -1 to 1 tanh function will be selected. Backpropagation with momentum learning rule is chosen because it is most commonly used technique and most suited for this type of problem. In backpropagation the objects in the training set are given to the network one by one in random order and the regression coefficients are updated each time in order to make the current prediction error as small as it can be. Momentum is an extra term that added to the standard backpropagation formulae to improve the efficiency. In first design, system had only five hidden neurons, the aimed maximum error is 0.01, learning rate is 0.2 and the momentum is 0.7. After 8000 iterations the error did not drop below 0.05. A larger learning rate (0.5) is selected but the results was the same. The main reason for those results was the small amount of hidden neurons. With 10 hidden neurons, system again was not trained. For 10 hidden neurons, maximum error of 0.03 is selected then the system is trained. To summarize all the attempts and their results, a table is given below (Table 1).

Cattral and Oppacher made another data mining-classifying study with poker hand dataset [6]. This time their proposed algorithm RAGA is compared with Waikato environment for knowledge analysis (Weka). Their goal of learning was to achieve better than 50.121%. Experiments with algorithms in Weka showed that most of them achieved 50.121%, best achievement was JRip algorithm with 56.616% accuracy. However, RAGA with 100 rules was able to achieve an average of 99.56% accuracy over 10 turns, best score was 99.76% correct. With 1000 rules the best case in 10 runs rose to 99.96%.

Mazur used neural network for design a computer bot that plays poker [7]. This time another type of poker game is concerned. Player will receive two cards and five cards will be on the table, at first three cards will open then players make their decisions, then another

card will be added on the table and players again make their decisions and last card will be added on the table then players make their final moves. 7 inputs are used for this network. These inputs are, the numeric value of first card in the hand scaled from 0 to 1, second card's value scaled from 0 to 1, whether or not they are suited, position at the table (to determine when to act), the average value of the two cards in hand, the difference between the first card's value and the average, the difference between the second card's value and the average. The output is 1 if the decision is raise and 0 if it is fold. 10461 datas are used. Results are given as correctly predicted raise is 88.6%, correctly predicted fold is 98.7% and overall accuracy is 97.9%. Author did not tell how many neurons he used or which transfer function he chose.

Nicolai and Hilderman presented several algorithms for teaching agents how to play no-limit Texas Hold'em Poker using a hybrid method known as evolving neural networks [8]. Evolutionary algorithms mimic natural evolution and reward good decisions while punishing less desirable ones. The system has 35 inputs, 20 hidden nodes and 3 outputs. Inputs of the system are chips in pot, chips to call, number of opponents, percentage of hands that will win, number of hands until dealer, chip counts, overall aggressiveness and recent aggressiveness. The outputs are fold, call and raise. The output of the network is stochastic to attempt to model bluffing.

Vlieg tested the suitability of neural networks to poker agents. He used MATLAB to design neural network system [9]. As targets (check-call, bet-raise, fold) can be seen as classes, the patternnet network said to be most appropriate for the poker learning problem. The algorithms trainrp and trainscg were recommended in combination with patternnet. Both functions were tested on a sample dataset of 500000 instances to compare their performance. They achieved almost the same result, trainscg was slightly better and required less time. Number of nodes is also tested with hiring 6, 8, 9, 10, 11 and 12 nodes. All configurations gave similar results but the fastest been the system with 11 nodes. Output transfer function was set to logsig to get desired output values between 0 and 1. The system tested with poker bots supplied by Poker Academy Pro. Experiments showed that all the trained bots are weaker than their opponent bots. The author said the main reason for the bad resuls might be the lack of an opponent model among the input features of the neural network.

Computers playing poker is a research topic on its own [10]. One of the most known poker research group is University of Alberta Computer Poker Research Group [11]. They are the developers of "Polaris", the program that played poker with six human champions and win with three wins, two lost and one tied game. Computer versus computer competitions also held to compare different algorithms and methods designed by different researches [12]. The innovations in computer poker can be seen with this kind of competitions.

Billings et al., discussed the challenges of the poker and described design considerations and architecture of the poker program "Poki" which is developed by University of Alberta Computer Poker Research Group [13].

III. First Training Attempts

A neural network system with 25010x10 input matrix and same sized output matrix is created with the aid of MATLAB Neural Networks Toolbox. Number of hidden neurons is

selected as 130 to handle large amount of data. Some of the system parameters are given as epoch = 1000, learning rate = 0.01, training function = resilient backpropagation, transfer functions = tansig. 80% of the input data is used to train the system, and 10% of the input data is used to test the results. Resilient backpropagation is a batch update algorithm and one of the fastest weight update mechanisms [14]. Some of the other training functions will be also used and results will be compared with each other in the scope of the project.

80% of the data is used to train the system, 10% of the data is used to validation and the other 10% is used to test the system. The training time for the parameters given above is 4.06 minutes. Mean squared error plot is given in figure 1. The result showed that the error is getting smaller as iterations goes on. Figure 2 shows the confusion matrix. It can be seen the best result is for the first class (high card case), 98.9% of the data estimated correctly. Overall results showed 92.4% of the data is classified truly. The results are also obtained by comparing the system output to desired output. This is done by a small MATLAB code. The result is 92.407037185125940, which matches with the matrix results perfectly.

Another training test is made with same parameters except the hidden neuron numbers. This time 50 neurons are employed and the calculated result is 80.5878%. If the transfer function changes to ‘logsig’ for this case, the result drops to 50.3559% with 3.33 minutes training time.

Using gradient descent with momentum backpropagation, 0.2 momentum constant and employing ‘tansig’ transfer function the calculated result is 45.8896%.

IV. Expected Results

The training examples showed that small numbers of hidden neurons may not produce acceptable results. The next step is to set some parameters like learning rate, neuron numbers etc. and only change one parameter to systematically show the effect of the change. Over 80% correct classifying will be accepted as sufficient result but the aim will be adjust the network to achieve over 90% correction. Parameters from Table 1 will be used to check if the results from MATLAB are similar with [5] or not. To make a reliable comparison, input data size may be reduced to 1003, the same size that used in [5]. Due to this reduction a decrease in neuron numbers and iterations is expected.

Training attempt	Number of hidden neurons	Training set	Maximum error	Learning rate	Momentum	Total mean square error	Number of iterations	5 random inputs test - number of correct guesses	Network trained
1	5	full	0.01	0.2	0.7	/	>8000	/	no
2	5	full	0.01	0.5	0.7	/	>5000	/	no
3	10	full	0.01	0.2	0.7	/	>3300	/	no
4	10	full	0.03	0.2	0.7	0.00738	134	3/4	no
5	12	full	0.02	0.2	0.7	0.0055	224	3/5	no
6	**12**	**full**	**0.01**	**0.1**	**0.7**	**0.00198**	**185**	**3full+2partialy/5**	**yes**
7	**15**	**full**	**0.01**	**0.2**	**0.7**	**0.00225**	**152**	**3full+2partialy/5**	**yes**
8	15	full	0.01	0.1	0.8	0.00246	121	/	no
9	18	full	0.01	0.5	0.8	/	>1400	/	no
10	**18**	**full**	**0.01**	**0.1**	**0.2**	**0.00176**	**278**	**5/5**	**yes**
11	18	only 20% of instances used	0.01	0.1	0.2	0.00214	169	1/5	no
12	18	only 80% of instances used	0.01	0.1	0.2	0.00156	109	2full+3partialy/5	no

Table 1. Table of training attempts and their results for [5].

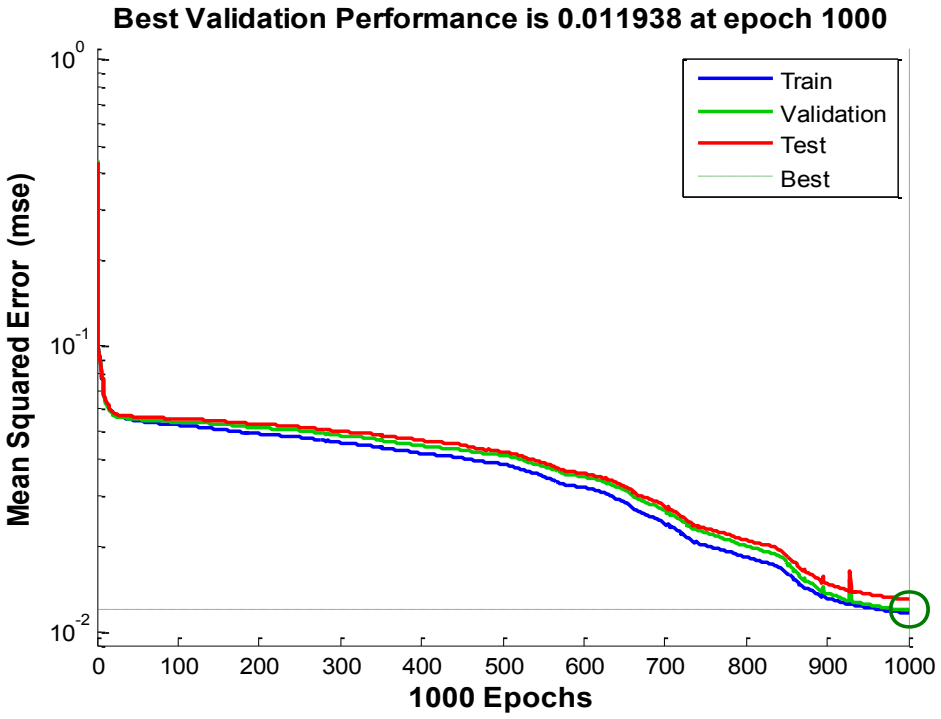

Fig 1. MSE vs. Epochs for first training attempt.

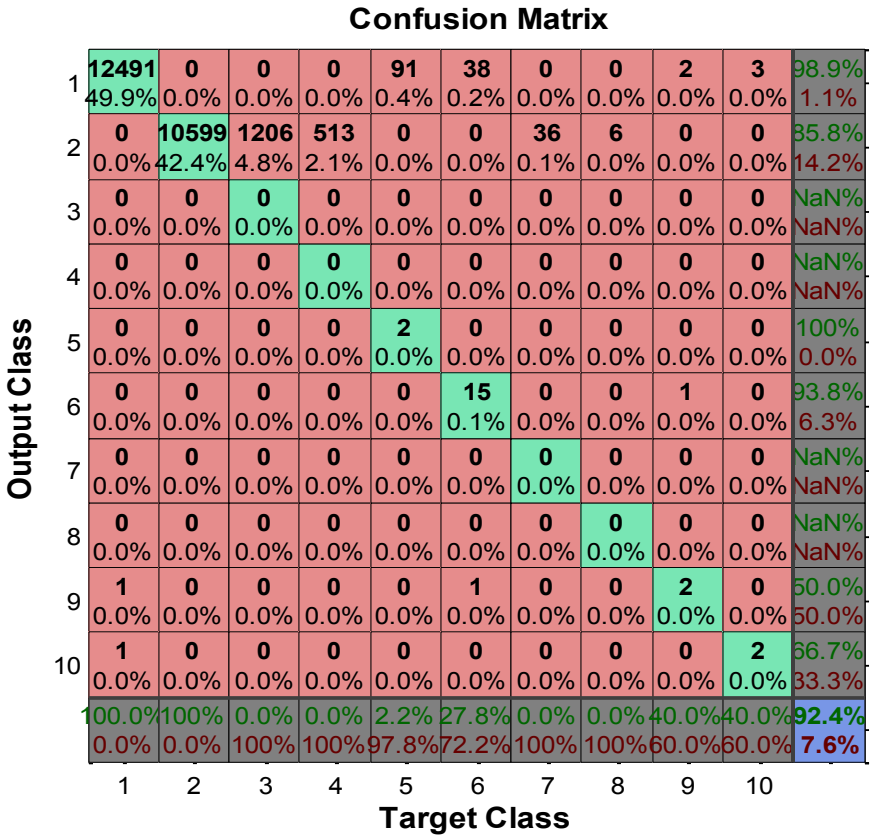

Fig.2 Confusion Matrix for first training attempt.

References

1. http://archive.ics.uci.edu/ml/datasets/Poker+Hand
2. http://en.wikipedia.org/wiki/Poker
3. R. Cattral, F. Oppacher, D. Deugo. Evolutionary Data Mining with Automatic Rule Generalization. Recent Advances in Computers, Computing and Communications, pp.296-300, WSEAS Press, 2002.
4. http://www.rulequest.com/see5-win.html
5. http://neuroph.sourceforge.net/tutorials/PredictingPokerhands/Predicting%20poker%20hands%20with%20neural%20networks.htm
6. Cattral, Robert, and Franz Oppacher. "Discovering rules in the poker hand dataset." *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007.
7. http://mattmazur.com/2009/10/13/experimenting-with-a-neural-network-based-poker-bot/
8. Nicolai G, Garrett, and Robert J. Hilderman. "Algorithms for Evolving No-Limit Texas Hold'em Poker Playing Agents." *IJCCI (ICEC)*. 2010.
9. Tjebbe Laurens Vlieg. "Computer Poker with Neural Networks", Bachelor thesis, University of Amsterdam, 2012.
10. Rubin, J., & Watson, I. (2011). Computer poker: A review. *Artificial Intelligence*, *175*(5), 958-987.
11. http://poker.cs.ualberta.ca/
12. http://www.computerpokercompetition.org/
13. Billings D., Davidson A., Schaeffer J., Szafron D. *"The Challenge of Poker"*, Artificial Intelligence, Vol:134, Issues:1-2, 201-140, 2002.
14. http://en.wikipedia.org/wiki/Rprop