Voting with Random Neural Networks: a Democratic Ensemble Classifier

Michele Donini[1], Andrea Loreggia[2], Maria Silvia Pini[2], Francesca Rossi[3]

1: CSML - Istituto Italiano di Tecnologia, Genoa, Italy
2: University of Padova, Italy, Email: andrea.loreggia@gmail.com, pini@dei.unipd.it
3: IBM T.J. Watson Research Center, Yorktown Heights, NY, USA (on leave from University of Padova). Email: frossi@math.unipd.it

Abstract. One of the main problems in machine learning is the investigation of the relationships between features in different datasets. In this case, using different classifiers can be more appropriate for different regions. Many efforts have been spent in looking for the best classifier and in fine tuning of the parameters. This requires knowledge of the domain and good expertise in fine tuning of the various classifiers. We propose a technique of voting with neural networks that does not require knowledge of the domain and avoids for a specific research of hyper-parameters values, because neural networks topology and the hyper-parameters are generated at random from a uniform distribution. The output of each neural network can be considered as a ranking over the probabilities that each sample belongs to a class, so the output of each neural network is a preference over the classes. We aggregate these preferences via different voting rules and we provide an empirical evaluation of the approach.

1 Introduction

In recent years researchers did a lot of work using multiple classifiers to solve complex classification problems, indeed different classifiers can be more appropriate for different regions of the solution space.

Finding the best classifier and fine-tuning its hyper-parameters is a high-consuming task, which requires many efforts and, at the same time, knowledge of the domain and good expertise in fine-tuning of the various classifiers. Many different research works have shown that no single learning algorithm can uniformly outperform other algorithms over all data sets [8, 5], but during the last years neural networks show promising performances in many different domains attracting attention of both researchers and companies for their theoretical properties and their performance in real-world scenarios.

Many researchers have studied methods for constructing good ensembles of classifiers [6, 8, 5] and the main discovery is that ensembles are often much more accurate than the individual classifiers [6]. Classifier combination is widely applied in many different fields [4, 14], although in many cases the performance of a combination method cannot be accurately estimated theoretically but can only be evaluated on an experimental basis in specific working conditions (a specific set of classifiers, training data and sessions, etc.). Techniques from decision theory are already been used in the past, trying to improve the performance of some ensemble classifiers methods [7, 5].

In this paper we provide a new approach in the ensemble classifier area that does not require knowledge of the domain and that does not require a good expertise in fine tuning of the parameters. This method exploits neural networks and voting theory.

Voting theory [1] is a wide research area that considers similar scenarios: in an election, voters vote by expressing their preferences over a set of candidates (that we call objects or decisions), and a voting rule decides who the winner candidate is. Voting theory provides many rules to aggregate preferences. If there are only two objects, the best rule, according to many criteria, is majority voting [1]. When there are more than two objects, there are many voting rules one could use (Plurality, Borda, k-approval, etc.), each with its pros and cons. Each rule takes in input (a part of) the preference orderings of the voters and gives in output the winner object, that is, the object that is considered to be the best according to the rule.

We propose a technique of voting with neural networks that does not require knowledge of the domain and avoids for a specific research of hyper-parameters values, because neural networks topology and the hyper-parameters are generated at random from a uniform distribution. The output of each neural network can be considered as a ranking over the probabilities that each sample belongs to a class, so the output of each neural network is a preference over the classes. We aggregate these preferences via different voting rules.

We provide an empirical evaluation of this approach by performing experiments on several datasets (from UCI [10]). Experimental results show that most of the times this approach outperforms some other ensemble classifiers methods on standard benchmark datasets.

The paper is organized as follows. In Section 2 and 3 we provide the basic notions of neural networks and voting rules. In Section 4 we present our approach that exploits voting theory in the ensemble classifier domain using neural networks and we provide experimental results. Finally, in Section 5 we summarize the results of the paper and we give some hints for future work.

2 Neural networks

A neural network is a structure used for classification or regression tasks especially when the dimensionality of data is high and non-linearity make these tasks hard to accomplish [3, 13].

The neural network technique is inspired by the brain neural activity [12]: a set of nodes [11] are used to read the input information, these input are then aggregated using a weighted sum that in turn it is processed with an activation function f. Figure 1 describes these relationships between inputs and outputs of a single node.

We can use multiple nodes at the same time, this allows for a more complex structure which is able to learn complex function from high-dimensional and non-linear data [9]. Figure 2 shows an example of of neural network with multiple nodes. The modular approach of the deep network allows for a stack implementation of the network using different framework.

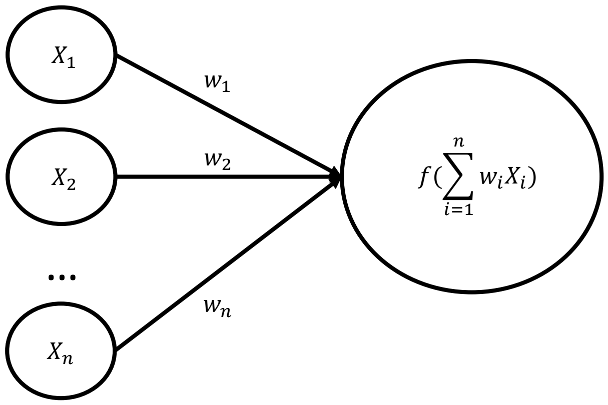

Fig. 1: Inupt and output relationship in a single node of a neural network.

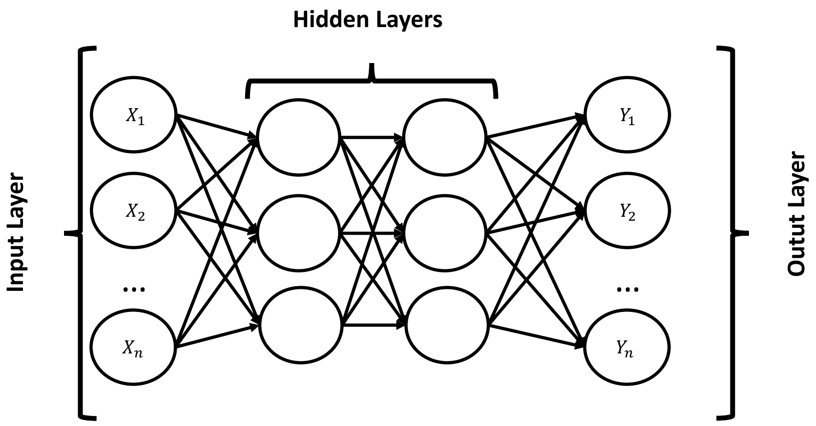

Fig. 2: A neural network with multiple nodes.

3 Voting rules

A voting rule [1] allows a set of voters to choose one among a set of candidates.

Voters need to submit their vote, that is, their preference ordering over the set of candidates (or part of it), and the voting rule aggregates such votes to yield a final result, usually called the winner.

In the classical setting [1], given a set of candidates $C = \{c_1, ..., c_n\}$, a profile is a collection of n total orderings over the set of candidates, one for each voter $x_i \in \{1, ..., n\}$.

Given a profile, a voting rule, also known as a social choice function, maps it onto a winning candidate.

Some examples of widely used voting rules are:

- *Plurality*: each voter states who the preferred candidate is, and the candidate who is preferred by the largest number of voters wins;
- *Borda*: given m candidates, each voter gives a ranking of all candidates, the i-th ranked candidate has a score $m - i$, and the candidate with the greatest sum of scores wins;
- k-*Approval*: each voter approves k candidates on m total candidates, and the candidate with most votes of approval wins.

Each of these rules has its advantages and drawbacks. Voting theory provides an axiomatic characterization of voting rules in terms of desirable properties such as: the absence of a dictator, unanimity, anonymity, neutrality, monotonicity, independence of irrelevant alternatives, and resistance to manipulation [1].

4 Our approach and experimental results

In this section we briefly describe our approach and we report a preliminary evaluation.

We propose a technique of voting with neural networks. Each neural network is a voter and the output of each neural network can be considered as a ranking over the probabilities that each sample belongs to a class, so the output of each neural network is a preference over the classes. We aggregate these preferences via different voting rules.

Neural Networks give very good performances in many different domains, for this reason in this approach we use these structures as classifiers, but usually experts spend a lot of time in coming up with a network that has the best structure and fine-tuning its hyper-parameters to reach best performances. In this work we argue in favour of a random search of these values, as already studied in literature [2].

Networks are generated to use the same input and with the same numbers of output nodes, this because they use the same input and they have to predict the same type of output. The number of hidden layers h_i is drawn from a uniform distribution in the range $2 \leq h_i \leq 5$. Given the number of features f in input, the number of nodes n_i i is such that $2f \leq n_i \leq f^2$ for the i-th hidden layer. The output of the last layer of each network can be seen as a ranking over the probability that a sample belongs to a class. The class with higher value is the most preferred one. These rankings or preferences can be aggregated using a voting rule, in such a way we can reward with different points classes that are ranked in the output of each neural network.

Different neural networks can generalize the learning functions in different ways as to learn different areas of the solutions space. So, while the most preferred class of some neural networks can be wrong, its full ranking over classes can bring diversity in the community of neural networks and their aggregation can reward the right class. There could exist classes with the same probabilities, in these cases ties are broken lexicographically in favor of the class with smaller value.

In the experimental tests we consider voting systems with 10 voters, where each voter is a random network. We aggregate outputs of neural networks using different voting systems as well as a weighted version of each voting system. This means that after the training phase, each network shows a training accuracy which can be used as a trustworthy factor. In the weighted version of the election, we use the training accuracy value of each neural network as a multiplication factor of the output. This means that for each candidate c_i, its total score can be computed as follow:

$$tot\_score(c_i) = \sum_{n \in N} acc(n) \times score_V(pos(c_i, n))$$

where: $acc(n)$ is the training accuracy value for the neural network n, $pos(c_i, n)$ is the rank of candidate c_i in the output of the network n, and $score(pos)$ is the reward that a specific voting rule V gives to a candidate ranked in the pos-th position.

It is important to notice that different voting systems have the same performances when the number of classes (or candidates) is 2.

Table 1 reports a preliminary analysis of our approach on 17 datasets from the UCI [10] repository. We aggregated the output of 10 random neural networks using different voting rules, namely Borda, Plurality and k-approval, where k is $\frac{n}{2}$, that is the number of classes n for each dataset divided by 2.

We performed two different elections for each voting rule: an unweighted election, where each neural network (which is a voter in the elections) has the same weight, and a weighted election, where the contribution that each neural network preference gives to a candidate is multiply by its training accuracy. The weighted voting system represents an election where the training performance of each network is considered as its trustworthiness parameter.

Table 1 reports accuracies that are averaged over 10-fold cross validation, and the numbers inside parenthesis are the standard deviations. The columns of Table 1 labelled with Local Voting and Local NB report performances shown in [7], while the column labelled with Voting reports performances presented in [5].

Experimental results show that our approach outperforms the approaches presented in [7, 5] in 7 out of 17 datasets, just for 1 dataset (i.e. mushroom) the performance is the same. For the remaining 9 datasets performances are very close to the state-of-the-art with a mean absolute error of 0.0499.

Dataset	Borda		Plurality		$\frac{n}{2}-$Approval		Local Voting [7]	Local NB [7]	Voting [5]
	Unweighted	Weighted	Unweighted	Weighted	Unweighted	Weighted			
anneal	0.9749 (0.0190)	0.9749 (0.0190)	0.9749 (0.0190)	0.9749 (0.0190)	0.6098 (0.1914)	0.5314 (0.1766)	-	-	0.9902
breast-w	**0.9659** (0.0220)	0.9659 (0.0220)	0.9659 (0.0220)	0.9659 (0.0220)	0.9659 (0.0220)	0.9659 (0.0220)	0.9645	0.9632	-
balance-s	0.8920 (0.0349)	0.8880 (0.0349)	0.8920 (0.0349)	0.8880 (0.0349)	0.8920 (0.0349)	0.8880 (0.0349)	0.8870	0.9018	0.8038
crx	**0.8606** (0.0387)	0.8588 (0.0424)	0.8606 (0.0387)	0.8588 (0.0424)	0.8606 (0.0387)	0.8588 (0.0424)	0.8539	0.8287	-
dermatology	0.9659 (0.0264)	0.9659 (0.0264)	0.9659 (0.0264)	0.9659 (0.0264)	0.8731 (0.0800)	0.8731 (0.0800)	-	-	0.9732
glass	**0.7954** (0.0704)	0.7954 (0.0704)	0.7895 (0.0703)	0.7954 (0.0653)	0.6137 (0.1160)	0.6310 (0.1274)	0.7548	0.7266	0.7482
haberman	**0.7420** (0.1130)	0.7420 (0.1130)	0.7420 (0.1130)	0.7420 (0.1130)	0.7420 (0.1130)	0.7420 (0.1130)	0.6989	0.7130	-
heart-c	0.6247 (0.0731)	0.6163 (0.0857)	0.6288 (0.0742)	0.6122 (0.0657)	0.6077 (0.0710)	0.5955 (0.0754)	0.8053	0.8136	-
hepatitis	0.8385 (0.0720)	0.8385 (0.0720)	0.8385 (0.0720)	0.8385 (0.0720)	0.8385 (0.0720)	0.8385 (0.0720)	0.8471	0.8618	0.8178
ionosphere	**0.9321** (0.0250)	0.9321 (0.0250)	0.9321 (0.0250)	0.9321 (0.0250)	0.9321 (0.0250)	0.9321 (0.0250)	0.8824	0.8191	0.9180
iris	0.8333 (0.1236)	0.8083 (0.1346)	0.7833 (0.1404)	0.7917 (0.1304)	0.7833 (0.1404)	0.7917 (0.1304)	0.9453	0.9567	0.9493
lymphogra	0.8205 (0.0986)	0.8205 (0.0986)	0.8205 (0.0986)	0.8121 (0.1007)	0.8295 (0.0937)	0.8121 (0.0935)	0.8200	0.8295	0.8307
mushroom	1.0000 (0.0000)	1.0000 (0.0000)	1.0000 (0.0000)	1.0000 (0.0000)	1.0000 (0.0000)	1.0000 (0.0000)	-	-	1.0000
primary-t	**0.4499** (0.1040)	0.4389 (0.0959)	0.4497 (0.1058)	0.4425 (0.0973)	0.2802 (0.0913)	0.1549 (0.0515)	0.4490	0.4446	-
vowel	0.9457 (0.0161)	0.9457 (0.0161)	0.9457 (0.0115)	0.9482 (0.0165)	0.2058 (0.0459)	0.2500 (0.0552)	0.9123	0.9456	0.9791
wine	**0.9965** (0.0030)	0.9965 (0.0030)	0.9965 (0.0030)	0.9965 (0.0030)	0.9965 (0.0030)	0.9965 (0.0030)	0.9707	0.9892	-
zoo	0.9375 (0.1008)	0.9375 (0.1008)	0.9375 (0.1008)	0.9375 (0.1008)	0.8375 (0.0976)	0.8625 (0.1038)	0.9711	0.9753	0.9575

Table 1: Performances of our approach with Borda, Plurality and n/2-Approval voting rules w.r.t. previous results [7, 5]. The table reports accuracy. The numbers in parenthesis are standard deviations.

5 Conclusions and future work

In this paper we have proposed a technique of voting with neural networks. This approach does not require knowledge of the domain and avoids for a specific research of hyper-parameters values. We have evaluated the approach experimentally and we have noted that the performance of our method in some cases is also better than the state of the art methods that require expertise both in the knowledge of the domain and in machine learning.

An interesting future work would be to extend our work by considering complex neural structures able to deal with structured data or images. In this way we could compare our approach with the one shown in the literature [2].

Bibliography

[1] K. J. Arrow, A. K. Sen, and K. Suzumura. *Handbook of Social Choice and Welfare.* North-Holland, 2002.

[2] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[4] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[5] I. Gandhi and M. Pandey. Hybrid ensemble of classifiers using voting. In *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on*, pages 399–404. IEEE, 2015.

[6] J. Kittler, M. Hatef, and R. P. W. Duin. Combining classifiers. In *Proceedings of the Sixth International Conference on Pattern Recognition*, pages 897–901, Silver Spring, MD, 1996. IEEE Computer Society Press.

[7] S. B. Kotsiantis and P. E. Pintelas. Local voting of weak classifiers. *KES Journal*, 9(3):239–248, 2005.

[8] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. Machine learning: a review of classification and combining techniques. *Artif. Intell. Rev.*, 26(3):159–190, 2006.

[9] M. Kubat. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, ISBN 0-02-352781-7. *Knowledge Eng. Review*, 13(4):409–412, 1999.

[10] C. B. D. Newman and C. Merz. UCI repository of machine learning databases, 1998.

[11] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[12] F. Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

[13] D. Rumelhart, G. Hinton, and R. Williams. *Learning Internal Representations by Error Propagation*. 1986.

[14] R. Wall, P. Cunningham, P. Walsh, and S. Byrne. Explaining the output of ensembles in medical decision support on a case by case basis. *Artificial Intelligence in Medicine*, 28(2):191–206, 2003.