Breaking Ties of Plurality Voting in Ensembles of Distributed Neural Network Classifiers Using Soft Max Accumulations

Yiannis Kokkinos and Konstantinos G. Margaritis*

Parallel and Distributed Processing Laboratory, Department of Applied Informatics, University of Macedonia, 156 Egnatia str., P.O. Box 1591, 54006, Thessaloniki, Greece

Abstract. An ensemble of distributed neural network classifiers is composed when several different individual neural networks are trained based on their local training data. These classifiers can provide either a single class label prediction, or the normalized via the soft max real value class-outputs that represent posterior probabilities which give the confidence levels. To form the ensemble decision the individual classifier decisions can be combined via the well known majority (or plurality) voting that sums the votes for each class and selects the class that receives most of the votes. While the majority voting is the most popular combination rule many ties in votes can occur, especially in multi-class problems. Ties are usually broken either randomly where the unknown instance is assigned randomly to one of the tied classes or using the class proportions where the tied class with the largest proportion wins. We present a tie breaking strategy that uses soft max confidence accumulations. Every class accumulates a vote and a confidence for this vote. If a tie occurs then the tied class with the maximum confidence sum wins. The proposed tie breaking in the voting process performs very well in all cases of different data distributions on various benchmark datasets.

Keywords: Neural Networks, distributed computing, ensemble classifiers, majority voting, plurality voting.

1 Introduction

Ensembles of neural network classifiers [1][2][3][4] are very popular tools in pattern recognition. Ensembles methods [5][6][7] are well known for their high accuracy and robustness since the combined predictions of several classifiers outperforms their individual predictions. In addition to the generalization performance, for which they are preferred, ensemble methods are usually the only choice available for other reasons like: (i) reduction of computational complexity by partitioning the data set into several smaller sub-sets, training different estimators on these sub-sets, and combining their predictions, (ii) modularity, by building modular systems that work on different input spaces, (iii) distributed learning from physically distributed data

* Corresponding author.

L. Iliadis et al. (Eds.): AIAI 2014, IFIP AICT 436, pp. 20–28, 2014.

repositories as well as Peer-to-Peer systems where gathering large volumes of data to a single location is unfeasible, (iv) their native parallelism and data scaling via the divide-and-conquer approach.

Distributed data collections can usually lay either in physically distributed database systems, in which case a small number of locations hold large volumes of data, or in Peer-to-Peer systems, where a large number of locations have typically small volumes of data. In any case the individual neural network classifiers are independently constructed in parallel, based on their local training data. After this training phase is finished, the classification predictions are combined via several schemes [8] in which the most popular one is majority voting [9][10][11][12] or its most frequently used version the plurality voting combining scheme. In plurality voting (for reviews see [5] [8] [10][11] [12]) a classification of an unlabeled instance is performed according to the class that obtains the highest number of votes (most of the votes). The strict version of (majority) voting needs the agreement of more than half of the participants to reach a decision, although this scheme can work for binary class problems only. Thus in reality, to cover the multi-class problems, the commonly used voting process is plurality voting, which selects the candidate class that receives the most votes. That is why usually one do not distinguish between plurality voting and majority voting, and the term "majority voting" is used even if the underlining criterion is plurality voting. This method is also known as the basic ensemble method [6].

However there may be more than one class that receives the largest number of classifications (or votes) by the ensemble members. Hence a tie can occur. In real life cases a tie-breaking chairperson can resolve this problem [13]. In classification problems such ties are usually broken arbitrary by randomly selecting one of the tied classes [5]. This random tie-breaking receives justification by the fact that all possible ways that other voters can vote are equi-probable. In the case of strictly binary class datasets one can just use an odd number of classifiers. In the case of k-nearest neighbour classifiers a nearest neighbour tie-break can be used. Another common strategy is choosing the class that is selected most often among the tied classes or tied classifiers [14]. This strategy predicts the class based on the largest class proportion among the ties classes. Another similar strategy can use the frequencies of predicted classes that occur during the training phase.

In this work we utilize confidence (soft max) accumulation as a method of breaking the ties in majority voting. Our motivation was the fact that a neural network classifier outputs, by default, the predicted class label accompanied by the confidence of this prediction given by the soft max function. Hence, we have tried to exploit further this extra information to improve the tie-breaking process and we have find out that it usually performs much better. Thus, the proposed tie-breaking method requires no additional computing resources other than those of simple majority voting. It works with classifiers, like neural networks, than in principle can accompany their class-prediction with a level of confidence, by using the soft max function. Every neural network classifier simply sends the predicted class *label* and its confidence *conf* for this class-prediction. Thus every prediction constitutes a pair {*label*, *conf*}. When a tie occurs in the voting process and two or more classes get the same number of votes then the proposed tie-breaking method sums up the received confidences for each of these tied classes. The class that accumulates the maximum confidence sum wins.

2 The Ensemble of Distributed Regularization Networks

We employ Regularization Neural Networks [15] [16] [17] for implementing the Neural Network classifiers. Regularization Networks are well known for using the real training data points as centers for their hidden neuron kernels. This is valuable when data features have discrete values, like in cases of image processing, computer vision and data mining. Using the real training data as kernel centers is a common strategy in kernel methods that can capture the data closeness approximate of any underlined problem distribution. These qualities elevate such type of stable kernel based learning methods to state of the art in modern machine learning [18].

We assume that each data location holds N_p training examples and there are L different data locations. Thus location p can train a neural network classifier $f_p()$ based on its local training set$\{\mathbf{x}_n, y_n\}_{n=1}^{N_p}$. Such an ensemble composed of Regularization Networks is illustrated in fig. 1 for the three-class case.

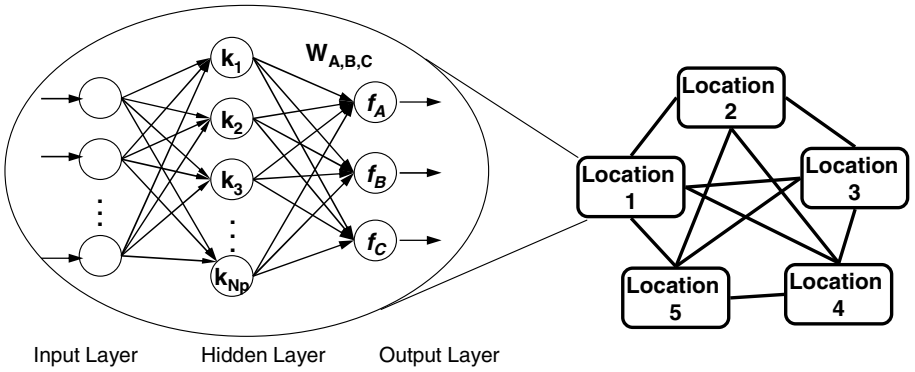

Fig. 1. An ensemble of Regularization Neural Networks distributed in several locations

The Regularization Network input neurons are as many as the number of data features. The hidden neurons are as many as the number of the training instances. A location p has N_p training instances. The output neurons are as many as the classes. The hidden-to-output layer weights $\mathbf{w}$ are computed for each class-output by solving a regularized risk functional. Assume three classes A, B and C then for the paradigm in fig. 1 the class output estimations of classifier $f_p()$ for an unknown $\mathbf{x}$ will be given by:

$$f_{p,A}(\mathbf{x}) = \sum_{n=1}^{N_p} w_{A,n} \cdot k(\mathbf{x}_n, \mathbf{x}) \tag{1a}$$

$$f_{p,B}(\mathbf{x}) = \sum_{n=1}^{N_p} w_{B,n} \cdot k(\mathbf{x}_n, \mathbf{x}) \tag{1b}$$

$$f_{p,C}(\mathbf{x}) = \sum_{n=1}^{N_p} w_{C,n} \cdot k(\mathbf{x}_n, \mathbf{x}) \tag{1c}$$

The class-output with the maximum value is chosen for the predicted class. The kernel function $k(\cdot,\cdot)$ can be any symmetric, positive semi-definite function, and the most commonly used is the Gaussian kernel $k(\mathbf{x}_n, \mathbf{x})=\exp(-\|\mathbf{x}_n-\mathbf{x}\|^2/\sigma^2)$ centered at a particular $\mathbf{x}_n$ point. The local kernel matrix $\mathbf{K}$ with entries $K_{ij} = k(x_i,x_j)$ and size $N_p \times N_p$ contains the information regarding the high density regions inside classes and the separating planes in between classes. To find the weights $\mathbf{w}$ in the Regularization Network (RN) [15] [16] [17] the learning problem is stated as the minimization on a Reproducing kernel Hilbert space (RKHS) of a regularized risk functional which has the usual data error term plus a regularization term that embodies the stabilizer:

$$\arg\min_{f\in H_K}\left\{ \frac{1}{N_p}\sum_{n=1}^{Np} (y_n - f_p(\mathbf{x}_n))^2 + \gamma\left\|f_p\right\|_K^2 \right\} \tag{2}$$

The data term is scaled proportionally to the number of data points, and $\gamma > 0$ is the regularization parameter. The minimization solution set is unique and the weight vectors $\mathbf{w}_A$, $\mathbf{w}_B$, and $\mathbf{w}_C$ for the three class paradigm that correspond to a regularization network classifier p are given by solving the linear systems [15] [1] [17] in eq. 3:

$$\mathbf{w}_A = (\mathbf{K} + N_p\,\gamma\,\mathbf{I})^{-1}\,\mathbf{y}_A \tag{3a}$$

$$\mathbf{w}_B = (\mathbf{K} + N_p\,\gamma\,\mathbf{I})^{-1}\,\mathbf{y}_B \tag{3b}$$

$$\mathbf{w}_C = (\mathbf{K} + N_p\,\gamma\,\mathbf{I})^{-1}\,\mathbf{y}_C \tag{3c}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{K}$ is the local kernel matrix (of size $N_p \times N_p$) and $\mathbf{y}_A$, $\mathbf{y}_B$ and $\mathbf{y}_C$ are the vectors (of size N_p) that hold the desired labels for each class A, B and C respectively, which in the hot encoding are 1 for labels of the same class points and 0 for the others. The rescaled regularization parameter $N_p\gamma$ is usually small and can be found during training via cross validation. In the experimental runs we use a typical value $N_p\gamma = 0.1$ for all the local Regularization Networks.

For an unknown $\mathbf{x}$ the class-output responses $f_{p,A}(\mathbf{x})$ $f_{p,B}(\mathbf{x})$ and $f_{p,C}(\mathbf{x})$ of each neural network f_p are normalized, so that they fall in the unity interval [0,1] and thus to reflect posterior probabilities. These posteriors produced by the softmax function are interpreted as confidence [19]. For instance the softmax confidence for the A class is:

$$h_{p,A}(\mathbf{x}) = \exp(f_{p,A}(\mathbf{x})) \,/\, (\exp(f_{p,A}(\mathbf{x}))+\exp(f_{p,B}(\mathbf{x}))+\exp(f_{p,C}(\mathbf{x}))) \tag{4}$$

Eq. 4 gives theoretically the Bayesian posterior probability (see also [19] for proof) as approximated by the regularization network. Thus each classifier outputs the predicted class together with the confidence for this class.

3 Combining Predictions Using Majority Voting

In a distributed parallel environment the combination of the decisions is straightforward. The majority voting is used in classification tasks where the individual classifiers produce a single class label, where in this case each classifier “votes” for a

particular class, and the class with the majority vote on the ensemble wins. This means that for an unknown **x** each classifier $f_p(\mathbf{x})$ outputs one integer number, given by the max argument of its class-outputs $f_{p,m}(\mathbf{x})$, that designates the most probable class label. With M classes and L classifiers the correct class is the k^{th} class that collects the largest number of votes, given by:

$$\text{class}(\mathbf{x}) = \arg\max_{k=1}^{M}\Big(\sum_{p}^{L} \{ \text{if } k = \arg\max_{m=1}^{M}\big(f_{p,m}(\mathbf{x}) \big) \text{ then 1 else 0 } \} \Big) \tag{5}$$

The simplest method to handle ties relays on a *First Labelled* basis which in case of a tie favours the class that was labelled earlier (or first). The drawback is that in a tie the class 1 (the first) will never lose, while the class M (the last) will never win.

The most commonly used tie-breaking method during the voting process is to break the ties arbitrary by *Randomly Selecting* one of the tied classes [5]. If a tie occurs this method choose a random class among the tied ones.

Differently from the random selection another way is to check which of the tied classes has higher prior probability and accordingly make the decision. Usually the class proportions given by the number of examples in every class (for the three class problem these are N_A/N, N_B/N and N_C/N) can representing the priors. Hence this method uses *Class Proportions*, and the ties are going in favour of the largest proportions.

Another way for counting priors is to monitor how many times a class occurs during the training process. Thus if class A and class B receive the same number of votes but the class B has higher frequency of occurrence, meaning that is predicted more often that the class A during the training of all the classifiers, then B is selected. Hence this method uses *Class Frequencies* during training.

A variant of the previous method is to count the correct predictions during the training of all classifiers and use these *Class Correctly Frequencies* as priors.

Using neural networks each classifier can afford to send a pair {*label*, *conf*}, where *label* is the predicted class m of the unknown **x** and *conf* is the soft max confidence of this prediction. The proposed method in this paper when a tie occurs is to sum the confidences for the received votes of the tied classes and select the one class that accumulates the *Maximum Confidence sum*. Therefore, with L classifiers which output their prediction m and their soft max confidence $h_{p,m}(\mathbf{x})$ when classifying **x** in class m, the correct class is the k^{th} class that collects the largest confidence sum (eq. 6):

$$\text{class}(\mathbf{x}) = \arg\max_{k=1}^{M}\Big(\sum_{p}^{L} \{\text{if } k = \arg\max_{m=1}^{M}\big(f_{p,m}(\mathbf{x}) \big) \text{ then } h_{p,m}(\mathbf{x}) \text{ else } 0\} \Big) \tag{6}$$

To provide an example consider the running paradigm in fig. 1, which illustrates a three-class problem with 5 neural network classifiers where each one can produce a pair {*label*, *conf*}. Assuming that for an unknown **x** the classifier $f_1()$ votes for class A with confidence 0.7 which gives a pair {A, 0.7} while $f_2()$ gives {A, 0.8}, $f_3()$ gives {B, 0.5}, $f_4()$ gives {B, 0.6} and $f_5()$ gives {C, 0.7}. A tie occurs for the classes A and B with 2 votes each. In this case the class A wins with sum of confidences 1.5.

4 Experimental Simulations

The classification performance of every tie-breaking method for majority voting are measured on a number of publicly available real-world benchmark problems which are downloaded from the UCI machine learning data repository [20] (http://archive.ics.uci.edu/ml). The specific details of these datasets are illustrated in table 1, where they are ordered by the number of their classes.

Table 1. Benchmark dataset details (ordered by class population)

Dataset Name	instances	Features	Classes
Sonar	208	60	2
Wisconsin (Diagnostic)	683	9	2
Diabetes Pima Indians	768	8	2
Spambase	4601	57	2
Phoneme	5404	5	2
Wine	178	13	3
Vehicle Silhouettes	846	18	4
Ecoli	336	7	5
Page Blocks	5473	10	5
Glass	212	9	6
Dermatology	358	34	6
Satellite Image	6435	36	6
Yeast	1479	8	9
Optical Digits	5620	64	10
Vowel Context	990	11	11
Spectrometer	531	100	24
Letter	20000	16	26

The groups of experiments aim at comparing the error rates of the proposed tie-breaking method and to make direct comparisons against the others. The experimental design is as follows:

(i) A dataset is randomly split into a training set (80%) and a test set (20%) with stratification

(ii) The training set of size N, is divided equally into L disjoint partitions ($N_p=N_1=N_2=\ldots=N_L$) which are distributed to the different locations,

(iii) every location p holds N_p data and builds a Regularization Neural Network classifier $f_p()$ based on its own local examples,

(iv) The classifiers are combined and the ensemble is tested on the test set.

(v) The procedure is repeated 30 times for each benchmark dataset and each ensemble population.

We measure the classification error rate which counts the number of incorrectly classified examples and divides by the test set population number.

The comparison results of the error rates and the standard deviations for each dataset and each tie-breaking method are illustrated in table 2.

Table 2. Error rates for the majority voting tie-breaking methods

Dataset	Ensemble size	First labeled	Random selection	Class Proportions	Class Frequencies	Class Correctly	Use max Confidence
Sonar	10	25.58 ±6.33	26.35 ±6.47	27.36 ±6.34	27.36 ±6.34	27.36 ±6.34	**23.25** ±5.50
Wisconsin	20	4.50 ±1.62	**4.28** ±1.57	4.50 ±1.62	4.50 ±1.62	4.50 ±1.62	**4.28** ±1.55
Diabetes	20	24.95 ±2.42	25.08 ±2.32	24.95 ±2.43	24.95 ±2.43	24.95 ±2.43	**24.27** 2.54
Spambase	40	9.64 ±0.83	9.66 ±0.91	9.65 ±0.95	9.65 ±0.95	9.65 ±0.95	**9.54** ±0.90
Phoneme	40	**18.42** ±1.15	18.54 ±1.29	**18.42** ±1.15	**18.42** ±1.15	**18.42** ±1.15	18.56 ±1.25
Wine	10	2.61 ±2.25	2.52 ±2.50	2.43 ±2.45	**2.34** ±2.29	2.43 ±2.45	**2.34** ±2.39
Vehicle	20	31.03 ±2.59	31.55 ±2.48	31.50 ±2.37	31.48 ±2.47	31.77 ±2.30	**30.86** ±2.63
Ecoli	10	14.25 3.26	14.49 3.21	14.58 3.36	14.58 3.36	14.58 3.36	**13.94** 3.47
Page Blocks	40	6.37 ±0.35	6.33 ±0.32	6.37 ±0.35	6.37 ±0.35	6.37 ±0.35	**6.28** ±0.37
Glass	10	39.55 ±5.33	39.55 ±5.72	39.70 ±5.73	39.70 ±5.87	40.00 ±5.62	**39.40** ±5.03
Dermatology	10	**3.12** ±1.88	3.56 ±2.27	**3.12** ±1.88	3.60 ±2.18	**3.12** ±1.88	3.51 ±2.35
Satellite Image	40	12.85 ±0.54	12.89 ±0.55	12.89 ±0.58	12.85 ±0.56	12.85 ±0.56	**12.81** ±0.86
Yeast	20	39.79 ±1.95	39.73 ±2.02	39.79 ±1.95	39.87 ±1.90	39.87 ±1.90	**39.65** ±1.97
Optical Digits	40	2.87 ±0.41	2.83 ±0.40	2.88 ±0.40	2.88 ±0.41	2.88 ±0.39	**2.76** ±0.40
Vowel Context	10	24.17 ±4.16	23.95 ±4.45	24.17 ±4.16	24.66 ±4.90	23.85 ±3.85	**19.12** ±4.32
Spectrometer	10	55.66 ±4.37	56.16 ±4.02	56.16 ±4.64	56.19 ±4.72	56.31 ±4.35	**54.95** ±4.49
Letter	40	11.51 ±0.59	11.44 ±0.61	11.47 ±0.57	11.60 ±0.59	11.45 ±0.60	**11.04** ±0.62

In table 2 the dataset name is in the first column. The second column has the number of classifiers in the ensemble. The third column illustrates the tie-breaking using the simplest “first labelled” method. The fourth column shows the results for the widely used method of breaking ties arbitrary by randomly selecting one of the ties classes in the voting. The fifth column is the tie-breaking by selecting among the tied classes the one with the largest “class proportion”, namely the largest number of training examples.

The sixth column uses "class frequencies" and selects among the tied classes the one class that had the maximum occurrence during training process of the classifiers (the one that appears most frequently). The seventh column corresponds to the tie-breaking that uses "class correctly frequencies" which selects the class among the tied classes that had the maximum number of correctly occurrences during the training process. The last column is the proposed method that sums also the received confidence of those votes and selects the tied class with the *maximum confidence sum*. Note that in many cases there are very small differences of the proposed method with the others. The differences depend only on the number of ties since the voting algorithm is otherwise the same. This simply means that not many ties where encountered during the testing.

We run several experiments and in most of the cases the proposed method of using confidence produces better results than the other tie-breaking strategies. In general, one can notice from table 2 that the differences of the methods in the comparisons are not substantial when not many ties occur. However when several ties occur during the voting process then the difference of the proposed method as compared with the others becomes considerable. In some multi-class datasets like the Vowel the proposed tie-breaking method has 4 units less error rate than that of the other methods.

5 Conclusions

Majority or plurality voting is the most popular decision combination rule in neural network ensembles. In many cases, regardless of number of classes and neural networks in the ensemble a tie in voting occurs. Usually such ties are broken arbitrary. We present a tie-breaking technique that uses confidence. Since each neural network classifier can afford to send by default the predicted label and the confidence for this prediction (via the soft max function), the proposed method receives pairs of {predicted class, confidence} for every unknown **x**. Thus every class accumulates a voting sum and a confidence sum. If a tie occurs then the tied class with the maximum confidence sum wins. When compared with other tie-breaking methods, like random class selection, or class selection using prior class probabilities, the proposed tie breaking performs very well in all cases of different data distributions on various benchmark datasets.

References

1. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12, 993–1001 (1990)
2. Perrone, M.P., Cooper, L.N.: When networks disagree: ensemble method for neural networks. In: Mammone, R.J. (ed.) Neural Networks for Speech and Image Processing. Chapman & Hall, Boca Raton (1993)
3. Hashem, S.: Optimal linear combinations of Neural Networks. Neural Networks 10(4), 599–614 (1997)
4. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. Artificial Intelligence 137, 239–263 (2002)

5. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley Interscience (2004)
6. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33, 1–39 (2010)
7. Seni, G., Elder, J.: Ensemble Methods in Data Mining. Morgan & Claypool Publishers (2010)
8. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On Combining Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
9. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics 22(3), 418–435 (1992)
10. Lam, L., Suen, C.Y.: A theoretical analysis of the application of majority voting to pattern recognition. In: 12th International Conf. on Pattern Recognition II, pp. 418–420 (1994)
11. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(1), 66–75 (1994)
12. Lam, L., Suen, C.Y.: Application of majority voting to pattern recognition: An analysis of the behavior and performance. IEEE Transactions on Systems, Man, and Cybernetics 27(5), 553–567 (1997)
13. Brams, S.J., Fishburn, P.C.: Voting Procedures. In: Arrow, K.J., Sen, A.K., Suzumura, K. (eds.) Handbook of Social Choice and Welfare, vol. 1. Elsevier (2002)
14. Woods, K., Kegelmeyer, W.P., Bowyer, K.: Combination of Multiple Classifiers Using Local Accuracy Estimates. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(4), 405–410 (1997)
15. Poggio, T., Girosi, F.: Regularization algorithms for learning that are equivalent to multilayer networks. Science 247, 978–982 (1990)
16. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. Neural Computation 7, 219–269 (1995)
17. Evgeniou, T., Pontil, M., Poggio, T.: Regularization Networks and Support Vector Machines. Advances in Computational Mathematics (2000)
18. Kashima, H., Ide, T., Kato, T., Sugiyama, M.: Recent Advances and Trends in Large-scale Kernel Methods. IEICE Transactions on Information and systems E92-D (7), 1338–1353 (2009)
19. Bishop, C.M.: Neural Network for Pattern Recognition. Oxford University Press Inc., New York (1995)
20. Frank, A., Asuncion, A.: UCI Machine Learning Data Repository. University of California, CA (2014), `http://archive.ics.uci.edu/ml`