American Journal of Engineering and Applied Sciences

Investigations

A Conceptualization of Distributed Computation for Machine Learning: The Voting Algorithm

Talal Talib Jameel

Department of Dentistry, Al Yarmouk University College, Baghdad, Iraq

Article history
Received: 22-12-2016
Revised: 18-01-2017
Accepted: 17-02-2017

Email: talal.alhabeeb@gmail.com

Abstract: This paper describes a voting algorithm that can be used to find the most optimal solution to clustering problems in machine learning. As part of the family of algorithms known as Condorcet methods, the voting algorithm is used to choose a particular candidate, even in the absence of a definitive majority. The algorithm proceeds in two steps: Renormalization and reconciliation. In the renormalization step all probability measure are reset so that the ensemble probability is always unity. In the reconciliation step a best choice is made based on the renormalized data. The result showed an excellent performance due to the use of linear time computations.

Keywords: Machine Learning, Cordorcet Method, Voting Algorithms, Clustering

Introduction

This paper examined the methods for parallelizing certain types of machine learning computations, using distributed computation architectures to split a large computation into a set of smaller computations. The goal is to provide a means for these smaller computations to be run in parallel while minimizing the amount of intercommunication that needs to be performs between these subcomputations.

Additional goals include the ability for these subcomputations to be order-independent and locality-independent. To be order independent means that the results of performing the set of computations should not depend on the order in which they are performed. Thus a set of computations [{a b}{c d}] should produce the same result as [{c d}{a b}]. To be locality-independent means that it should not matter which node or CPU executes a given set of computations (Pingali *et al*., 2003; Chen *et al*., 2004).

The final goal of this work is for the result of combining the subcomputions into a single result should produce only consistent results. We do not guarantee that the distributed computation will compute the same result as a serial computation would. This latter restriction arises from the fact that most machine learning algorithms involve random data (Witten and Frank, 2005; Snoek *et al*., 2012; Aha *et al*., 1991), so that a computation may not be reproducible. Thus, the only guarantee that we can offer is the result should be internally consistent: No axioms of the algorithm are violated by the final algorithm.

However, we do want to guarantee that the results are globally reproducible, in the sense that if the first algorithm converges then the final algorithm converges; if either algorithm diverges then both diverge; and finally that if both converge then the Euclidean distance between the two sets of weights is small. For the purposes of this paper we define “small” to mean that the Euclidean distance between the weights is less than the absolute value of any weight itself.

Here, we focused on classification algorithms. The goal of classification algorithms is to divide a set of data points P into categories (usually called clusters) that contain points with the same or similar set of properties (Hsu *et al*., 2003). For example, clusters can be composed of the subset of points from the set P that are all within a certain Euclidean distance of a certain point (the “centroid”). An example clustering result is shown in Fig. 1. There are hundreds of classification algorithms, so for the purposes of this document we will specifically consider three algorithms: Fixed Width Clustering (FWC) (Barbara and Jajodia, 2002), K-Means Clustering (KMC) (Bradley and Fayyad, 1998) and Fuzzy C-Means Clustering (FCMC) (Bezdek *et al*., 1984).

DOI: 10.3844/ajeassp.2017.151.155

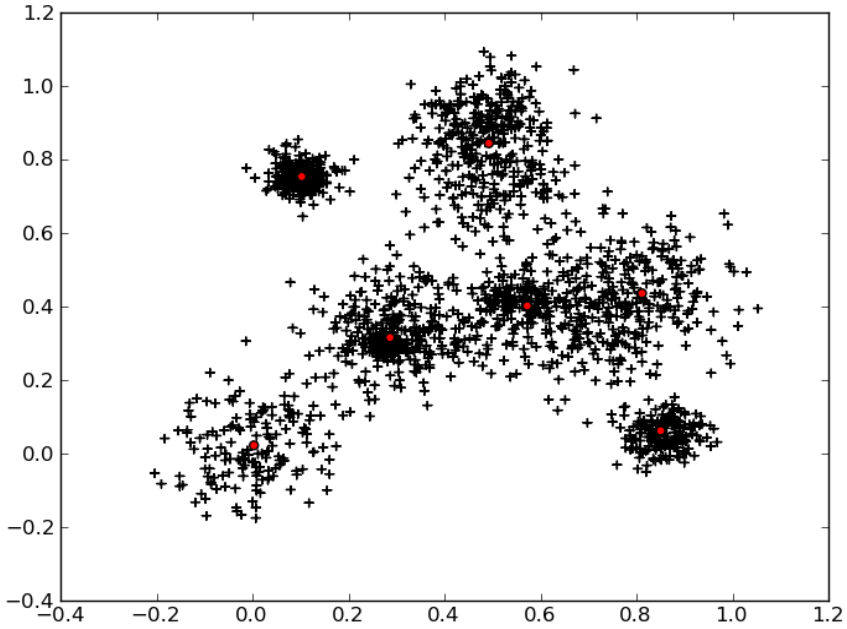

Fig. 1. An example of clustering

We define the membership matrix M to be the set of probability values such that M(I,J) is the probability that point I is a member of cluster J (Krishnapuram and Keller, 1993; Jain *et al*., 1999; Banerjee *et al*., 2005). For some algorithms (FWC, KMC) the elements of M are binary: Either a point belongs (1) or does not belong (0). We refer to this type of algorithm as *univalent*. Other algorithms, such as FCMC, allow the probability values to be any real number in the range [0,1] (Bensaid *et al*., 1996; Graves and Pedrycz, 2010).

We refer to such algorithms as *polyvalent*. In either type of algorithm it must be the case that the probability of any point belonging to some cluster (or to belong probabilistically to more than one cluster) is 1.0. Thus the sum of the membership probabilities for each point must sum to 1,0. The estimated number of clusters based on the classification processes were 3 (Fig. 1).

Since each point corresponds to a row of M, we have a requirement, which we refer to as the normalization requirement, that rowsum(J, M(I,J)) = 1.0. One of the difficulties is with distributed computations that only add entries to a subset of M is that when M is completely populated it may be the case that rowsum(J, M(I,J)) != 1.0. This leads to part 1 of the voting algorithm, the renormalization step:

Renormalization Step

If V(I) = rowsum(J, M(I,J)) != 1.0, then divide each element of row I by V(I).

While renormalization is necessary, it is not sufficient. To see why this is the case, consider a single point p whose membership is being computed relative to two sets of clusters C1 and C2. When both computations are done p will have a set of membership probabilities {p1 … px} for C1 (where |C1| = x) and a second set of membership probabilities {q1 … qy} for C2 (where |C2| = y).

In the univalent case a point x can only be assigned to one cluster, so if one of the pj probabilities is 1 and also one of the qk probabilities is 1, then we have an inconsistency. Let us refer to these two clusters as c1' and c2'. Since x cannot belong to both c1' and c2', we must set one of these probabilities to zero and leave the other probability as 1. To do this we employ the second part of the voting algorithm, the reconciliation step (univalent):

Reconciliation Step (u)

If a point appears to belong to more than one cluster then recalculate the metric distance of that point to each of the clusters and choose the cluster to which it is closest. Set this probability to be 1 and all other probabilities to be zero, in row x of the membership matrix.

For polyvalent algorithms reconciliation is more complex. In such algorithms we wish to identify the primary cluster to which a point belongs. Typically this cluster will have a probability membership value of at least 0.5.

However, if we fill the membership matrix and then renormalize, it may be the case that none of the probability values is greater than one half. To address this we use the reconciliation rule for polyvalent algorithm, which is used in conjunction with renormalization.

Reconciliation Rule (p)

Let E1 be the largest probability value and E2 be the second largest value of a given point (row) in the membership matrix. Suppose that after renormalization it is the case that both E1<0.5 and E2<0.5. Then we adjust the value of E1 to be 0.5 and adjust the value of E2 to be E2 + (0.5-E1). This operations preserves normalization so a second renormalization pass is not required.

Given these rules it is straightforward to describe the general distributed algorithm. First becomes the membership matrix M into a set of non-overlapping regions M1 … Mw. Second, given that there are N compute nodes, distribute the regions as evenly as possible among the N nodes. Run each node to completion and allow it to fill in the corresponding entries in the membership matrix. Once M is fully populated then run renormalization + reconciliation(u) for univalent algorithms and renormalization + reconciliation(p) for polyvalent algorithms.

The voting algorithm described above is part of a class algorithms known collectively as the "Condorcet method" (Black *et al*., 1958). The Condorcet method is a voting algorithm that will always elect the candidate that is the most preferred with respect to pairwise comparisons. In the worst cast, the total number of such comparisons that need to be performed is ½*N*(*N*-1), which is quadratic in the number of candidates. The winner of the election is referred to as the Condorcet winner.

A voting algorithm that always elects the Condorcet winner is described as a system that satisfies the Condorcet criterion. It is straightforward to verify that the voting protocol described at the beginning of this document satisfies the Condorcet criterion. In addition, our voting algorithm is purely linear in the number of candidates, thus significantly improving performance over the default (brute force) algorithm.

In systems that do not satisfy the Condorcet criteria it is possible that circular ambiguities arise as a result of the voting algorithm. That is, the result of an election can be intransitive even though all individual voters expressed a transitive preference. In a Condorcet election it is impossible for the preferences of a single voter to be cyclical, because a voter must rank all candidates in order and can only rank each candidate once, but the paradox of voting means that it is still possible for a circular ambiguity to emerge. A straightforward inspection of the reconciliation steps shows that our algorithm can never enter a limit cycle.

In Condorcet methods, as in most electoral systems, there is also the possibility of an ordinary tie. This occurs when two or more candidates tie with each other but defeat every other candidate. As in other systems this can be resolved by a random method, as our reconciliation step uses. In our algorithm the reconciliation step serves as a tie-breaker method and insures there is at most one selection (candidate) with a maximal score. A mechanism for resolving an ambiguity is known as ambiguity resolution or Condorcet completion method.

Results and Discussion

The performance results of the combined solution is shown in Fig. 2. From the figure it can be concluded that voting algorithm provide a relatively less time to process the cases of a dataset. This can be reasoned to the use of the history record of redundant modules to compute the final output. On the other hand, the FCMC was found to consume longer time followed by FWC and KMC respectively. As such, it can be said that our proposed solution provide an effective way to process and classify cases in large dataset.

Moreover, the performance of the proposed solution with regard to the number of iterations was also investigated. Figure 3 shows the comparison result of voting algorithm with FWC, FCMC and KMC based on the number of iterations. From the figure, it can be said that the proposed algorithm provided a stable performance results when the number of iterations is increased. However, FCMC was found to be the least performed algorithm followed by KMC and FWC respectively. This can be reasoned to that the proposed algorithm defined the parameters that need to be appropriately recognized when processing cases.

Future Work

Our form of voting algorithm insures that a maximal clustering selection is made in linear time. It is free from limit cycles and has an unambiguous procedure for breaking ties. Further, the algorithm we have defined is completely independent of the underlying machine learning algorithm that is used. However, the algorithm as currently described tends to favor clusters that form earlier (in terms of iterations). In the future, it would be desirable to find a variant of the reconciliation step that does not have this bias.

DOI: 10.3844/ajeassp.2017.151.155

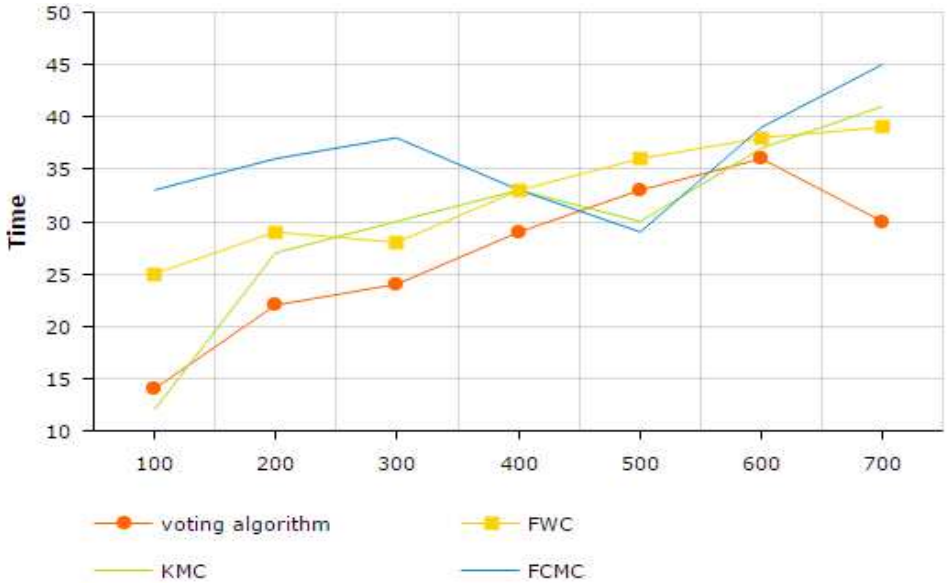

Fig. 2. A comparison of voting algorithm with FWC, FCMC and KMC based on the time for processing cases

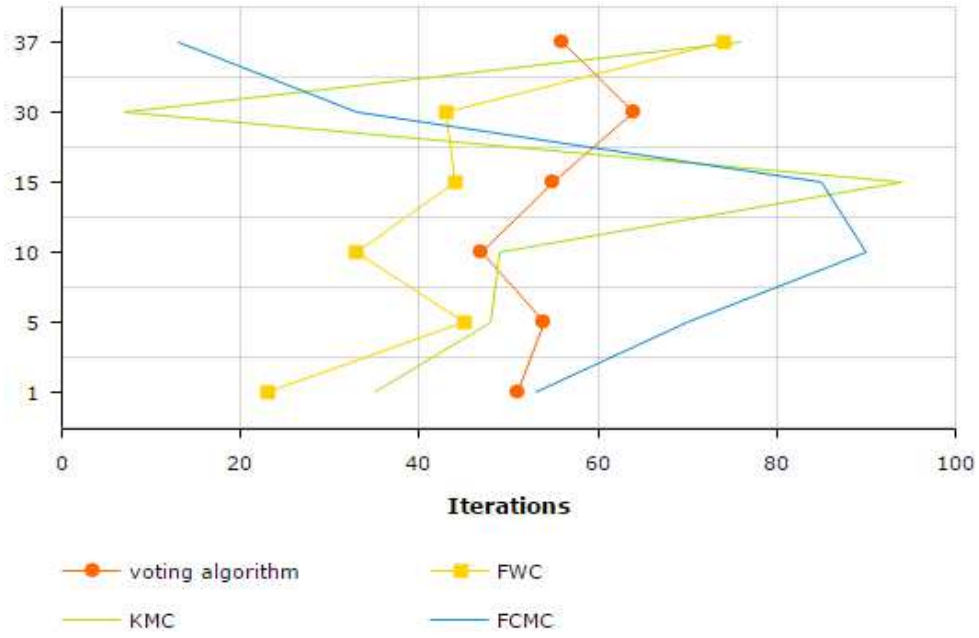

Fig. 3. A comparison of voting algorithm with FWC, FCMC and KMC based on the number of iterations

Conclusion

This study described the main aspects for applying discriminative machine learning named voting algorithm in sequential models. The process involve developing a sequence of operations in order to rerank the n-best outputs based on the Condorcet method. The main idea behind this work consists of utilizing global features as well as local features to help provide efficient classification performance of large data. Here, it was assumed that most machine learning for distributed computing may lack of ranking large dataset due to the need for alleviate the impact of the label bias problem, that get penalized due to the label bias problem. The prospective of renormalization and reconciliation were proposed in this study as a way for associating parses as a special form of sequential model without experiencing any reduction in data generality. At the beginning of this work we sought a method that will be easy to use, efficient and that would preserve

normalization. All of these goals have been achieved. Our form of Condorcet voting algorithm satisfies the Condorcet criterion, does not have circular ambiguities and breaks ties in a single iteration, as has linear performance. Our voting algorithm is also completely decouple from the underlying machine learning algorithm that produces the elements in the membership matrix, so it can be applied uniformly to any such algorithm without requiring any customization.

Acknowledgement

Thanks to the reviewers for their attention to detail and many valuable suggestions.

Funding Information

This research received no specific grant from any funding agency.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Aha, D.W., D. Kibler and M.K. Albert, 1991. Instance-based learning algorithms. Mach. Learn., 6: 37-66. DOI: 10.1007/BF00153759

Banerjee, A., C. Krumpelman, J. Ghosh, S. Basu and R.J. Mooney, 2005. Model-based overlapping clustering. Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Aug. 21-24, ACM, USA, pp: 532-537. DOI: 10.1145/1081870.1081932

Barbara, D. and S. Jajodia, 2002. Applications of Data Mining in Computer Security. 1st Edn., Springer Science and Business Media,
ISBN-10: 1402070543, pp: 252.

Bensaid, A.M., L.O. Hall, J.C. Bezdek, L.P. Clarke and M.L. Silbiger *et al*., 1996. Validity-guided (re)clustering with applications to image segmentation. IEEE Trans. Fuzzy Syst., 4: 112-123. DOI: 10.1109/91.493905

Bezdek, J.C., R. Ehrlich and W. Full, 1984. FCM: The fuzzy c-means clustering algorithm. Comput. Geosci., 10: 191-203.
DOI: 10.1016/0098-3004(84)90020-7

Black, D., R.A. Newing, I. McLean, A. McMillan and B.L. Monroe, 1958. The Theory of Committees and Elections. 1st Edn., Springer, Boston,
ISBN-10: 0898381894, pp: 241.

Bradley, P.S. and U.M. Fayyad, 1998. Refining initial points for k-means clustering. Proceedings of the 15th International Conference on Machine Learning, Jul. 24-27, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA., pp: 91-99.

Chen, G., O. Ozturk and M. Kandemir, 2004. An ILP-based approach to locality optimization. Proceedings of the 17th International Workshop on Languages and Compilers for Parallel Computing, Sept. 22-24, Springer, West Lafayette, IN, USA, pp: 149-163. DOI: 10.1007/11532378_12

Graves, D. and W. Pedrycz, 2010. Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study. Fuzzy Sets Syst., 161: 522-543. DOI: 10.1016/j.fss.2009.10.021

Hsu, C.W., C.C. Chang and C.J. Lin, 2003. A practical guide to support vector classification.

Jain, A.K., M.N. Murty and P.J. Flynn, 1999. Data clustering: A review. ACM Comput. Surveys, 31: 264-323. DOI: 10.1145/331499.331504

Krishnapuram, R. and J.M. Keller, 1993. A possibilistic approach to clustering. IEEE Trans. Fuzzy Syst., 1: 98-110. DOI: 10.1109/91.227387

Pingali, V.K., S.A. McKee, W.C. Hsieh and J.B. Carter, 2003. Restructuring computations for temporal data cache locality. Int. J. Parallel Programm., 31: 305-338. DOI: 10.1023/A:1024556711058

Snoek, J., H. Larochelle and R.P. Adams, 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In: Advances in Neural Information Processing Systems, pp: 2951-2959.

Witten, I.H. and E. Frank, 2005. Data Mining: Practical Machine Learning Tools and Techniques. 2nd Edn., Morgan Kaufmann, IS San Francisco,
ISBN-10: 008047702X, pp: 560.