A SURVEY OF DISTRIBUTED DATA CLASSIFICATION WITH BOOSTING[1]

Karlen Mkrchyan

Abstract: *Distributed data classification is becoming more and more actual nowadays. The growing data and it's decentralized nature forces to analyze data in distributed fashion. Many conventional algorithms tackle this problem well in case of centralized data, which differs from the case of distributed data. In distributed case may arise the problem of data privacy. We survey the distributed learning frameworks for classification and modifications of boosting algorithms.*

Keywords: *Ensemble Learning, Boosting, Distributed Boosting*

ITHEA Keywords: *A.1 Introductory and Survey, I.5 Pattern recognition: G.3 Probability and Statistics,I.2.6 Learning*

Introduction

Machine learning can be broadly defined as computational methods transforming the experience into expertise or knowledge. With the growing of available data size storing the whole data in one place becomes more and more inconvenient, and in some cases even impossible. Naturally, comes an idea of distributed data storages. One of the central learnig tasks is supervised

[1] Partially supported by grants № 18RF-144, and № 18T-1B407 of the Science Committee of the Ministry of Education and Science of Armenia

classification problem. Generally, the classification is the procedure which put objects into categories or classes. Consider the formal statement of distributed classification probem. Let the objects (observations) represented by corresponding feature vectors belonging to some subset $\mathcal{X}$ of n-dimensional euclidean spase R^n. Let we have k(k>1) data centers, each of which has its own training set $S_i = \{(\bar{x}_{i,1}, y_{i,1}), \dots, (\bar{x}_{i,m_i}, y_{i,m_i})\}$, i=1,...,k, $y_{i,j} \in Y$, where Y is a finite set of labels (categories/classes). The main characteristics of the learning algorithm in distributed case are

Classification accuracy (compared with centralized case),

Computational complexity,

Communication complexity.

Mention, that sometimes in addition may arise the issue of data privacy.

Distributed Boosting Algorithms

Boosting is one of most popular ensemble methods, was discovered in Breiman's work [2]. Among the boosting methods widely used the AdaBoost method [3]. Boosting is increased accuracy of classification by combining multiple weak learners into one strong learner.

Lazarevic and Obradovic proposed a distributed boosting algorithm [4]. The main objective of the distributed boosting algorithm is to construct an ensemble model on distributed system, which prediction accuracy will be close to the one, build in centralized way.

Denote by $\Delta_{i,t}$ the local distribution of i^{th} data center and with $w_{i,t}$ the local weights of each data center. The pseudo-code for distributed boosting framework [4] is given in fig. 1.

1. Each participant is given $S_i = \{ (x_{i1}, y_{i1}) , \dots, (x_{im_i}, y_{im_i}) \}$.
   Let $B_i = \{(j, y), j = 1,2, \dots, m_i, y \neq y_{i,j}\}$.
2. On each data center i the $\Delta_{i,1}$ is the initial distribution of examples such as $\Delta_{i,1} = 1/|B_i|$, and compute sums $\sum_{(j,y_i)\in B_i} \Delta_{i,1}(j, y)$.
3. Each participant i broadcasts the computed sums and makes a distribution $D_{j,1}$, which is part of global distribution. It is done by initializing the j^{th} interval $[\sum_{p=1}^{i-1} m_p, \sum_{p=1}^{i} m_p]$ in the distribution $D_{i,1}$ with values $1/|B_i|$.
4. Each site normalizes the $D_{i,1}$, such that $D_{i,1}$ is a distribution.
5. For $j = 1, \dots, k$ (For all distributed participants)
   For $t = 1, 2, \dots, T$
   $Q_{i,t}$ set is drawn from global distribution, Selected elements of B_i with highest weights Select training examples according to their indexes. Train a weak learner $C_{i,t}$ on $Q_{i,t}$
6. Create ensemble $E_{i,t}$ on $Q_{i,t}$, $i = 1,2, \dots, k$ at each site by combining $C_{i,t}$
7. Compute weak hypothesis $h_{i,t}: X * Y \rightarrow [0,1]$ by using $E_{i,t}$ ensembles
8. pseudo-loss for $h_{i,t}$ is $\varepsilon_{i,t} = \frac{1}{2}\sum_{(j,y)\in B_i} \Delta_{i,t}(j, y)(1 - h_{i,t}(x_{i,t}, y_{i,j}) + h_{i,t}(x_{i,t}, y_i))$
9. Let $\beta_{i,t} = \varepsilon_{i,t}/(1 - \varepsilon_{i,t})$
10. $w_{i,t}(i, y) = \beta_{i,t}^{\frac{1}{2}\left(1 - h_{i,t}(x_{i,t}, y_{i,j}) + h_{i,t}(x_{i,t}, y_i)\right)}$.
11. $V_{i,t} = \sum_{(j,y_i)\in B_i} w_{i,t}(j, y)$
12. Compute weight vector $U_{i,t}$ such that i^{th} interval $[\sum_{p=1}^{i-1} m_p, \sum_{p=1}^{i} m_p]$ is weight vector $w_{i,t}$, and the values in j^{th} interval are set to $V_{j,t}$, $i \neq j$.
13. Normalize all $D_{i,t}$ with normalization factor $Z_{i,t}$, after normalization $D_{i,t}$ make local distribution $\Delta_{i,t}$
14. $D_{i,t+1}(i) = \frac{D_{i,t}(i)}{Z_{i,t}} * \beta_{i,t}^{\frac{1}{2}\left(1 - h_{i,t}(x_{i,t}, y_{i,j}) + h_{i,t}(x_{i,t}, y_i)\right)}$, $Z_{i,t}$ normalization factor is the sum of all weights $D_{i,t}(i)$
15. Final hypothesis is as follows

$$h = arg\, max_{y\in Y} \sum_{t=1}^{T} \sum_{i=1}^{k} \left(log\left(\frac{1}{\beta_{i,t}}\right) h_{i,t}(x, y_i)\right).$$

Figure 1

As in original boosting algorithm the weight vectors $w_{j,i}$ are updated according to $E_{j,i}$ for every participant j. To make this part more clear we will go through update process in more details. In the step 9 we calculate the pseudo loss function value, which is used to evaluate the accuracy of the $h_{i,t}$ hypothesis. Then we update the weight of each training example in the following logic. If the example is classified correctly by $E_{j,i}$ then we reduce the weight, otherwise weight remains the same. Then by using sums $V_{j,i}$ the sum of weights are provided to all participants and the local distributions are changed according to $U_{j,i}$ weight vectors. Finally the $h_{j,i}$ classifiers, which are obtained at each step and at each participant, are combined to form the final hypothesis$h$.

There are many other variations of distributed boosting. In competing classifiers method classifiers $C_{i,t}$ such that, for any data pattern, it's assigned to some classifier and ensemble uses different classifiers depending on data example.

Above we mentioned that on each iteration we construct a classifier, which is ensemble of weak learners constructed in that step. In fact there is no single method for combining weak learners, and it strictly depends on the use case. One may use simple majority voting, weighted majority voting, confidence based voting and etc.The simplest method for composing the classifiers is simple majority voting, it can be interpreted as follows. $C_{i,t}$learned in distributed system produce $h_{i,j,t}$ on participant i, then weak hypostasis found in the following way $h_{i,t} = \frac{1}{k}\sum_{i=1}^{k} h_{i,j,t}$.

Weighted majority voting of classifiers is more complex way of combining weak learners. The weights $u_{i,j,t}$ of classifiers $C_{i,t}$, is proportional to their accuracy on their data center, thus the hypothesis computed as follows $h_{i,t} = \frac{\sum_{i=1}^{k} u_{i,j,t} * h_{i,j,t}}{\sum_{i=1}^{k} u_{i,j,t}}$.

The experiments on multiple data sets shows, that the stated distributed ensemble method is has approximately same prediction accuracy as classic ensemble method. In data examples were distributed data is not homogenous, the concurrent classifiers method shoed better results. Also experiments show,

that the algorithm depends on weighting method, and on same data set different results were obtained with varying weight methods. The main drawback of the stated algorithm is that it constructs too many classifiers, and then the final hypothesis constructed by combining huge amount of classifiers. The possible solution for this problem is to have pruning stage in this method, so that the number of insignificant classifiers could be reduced. As we can see from the algorithm, for constructing strong hypothesis algorithms does two mappings of hypothesis classes. The first one is to map weak hypothesis into ensembles, And then by combining these ensembles pick the final hypothesis. More formally let$H$ be a set of weak learners, from which our algorithm chooses hypothesis $L_{j,i}$ and let d_1 be a VC dimension of H. The algorithm firstly maps the classes of weak learners to ensembles of classifiers in the scope of PAC learnability [3]. More formally if we denote the family of functions, from which we choose ensembleswith H^E, then we will have the following (consider weighted voting for simplicity)

$$H^E = \{\, h: h(\bar{x}) = sign(\textstyle\sum_{t=1}^{T} a_t h_t(\bar{x})); h_t \in H, a_t \in R, t = 1,2 \dots T\}.$$

The values of a_t are infinite, consequently, H^E is infinite as well. After this step in the same way we map $H^E \to H^f$. Now let's assume the VC dimension of $H^E$is d_2, then theoretically for H^f we will have VC dimension $\tilde{O}(Td)$, where $\tilde{O}$ constrains logarithmic expressions and T is the number of rounds. Distributed version of AdaBoost differs from centralized version from this perspective, in which, there are only mapping from weak learners to strong learners. Thus, the a priori inference that the generalization error, would tend to centralized version generalization error would not be correct, even if training errors are close to each other. Let δ any number, then with probability $1 - \delta$ for generalization error for the case of centralized algorithm we will have the following

$$err_{centrlized}(h) \le \frac{1}{m}\textstyle\sum_{i=1}^{m} I(y_i h_e(\bar{x}_i) \le \theta_1) + O(\sqrt{\frac{\log \mathrm{m} \log |\mathrm{H}|}{m\theta^2}} + \log\frac{1}{\delta}).$$

Where θ is a margin measure, and ideally we would have θ at least equal to advantage of weak learner and m is the number of all data points. Now in the

place of h_e we have ensembles, which can be represented in the form above as well. After replacing h_e and other values for the case of distributed system, we will have the following

$$err_{distributed}(h) \leq \sum_{j=1}^{k} \frac{1}{m_j} \sum_{i=1}^{m_j} I\left(y_i f_j(\bar{x}_i) \leq \theta\right) + \sum_{j=1}^{k} \left(\sqrt{\frac{\log m_j \log |H|}{m_j \theta^2}} + \log \frac{1}{\delta}\right) +$$

$$O\left(\sqrt{\frac{\log m \log |H|}{m\theta^2}} + \log \frac{1}{\delta}\right).$$

Where $f_j = \sum_{i=1}^{T} a_i h_{j,i}(\bar{x})$. Note that above formula need explanation. The first problem in above formula is that it is not eligible to represent generalization error of final hypothesis, because ensembles are made on local data, thus, their generalization error is relative to their local data, not whole data. Let's assume that the data is homogenous. It means all data sets are drown from some unknown distribution, in the same way. In this case, naturally, any data center should have the same properties as others. Thus with this restriction one ensemble becomes similar to the case in centralized version, thus its generalization error could be spread on entire distribution. Let's denote M all data points distributed system. Now from the formula it can be inferred that the generalization error of the final hypothesis is at most the generalization error of the worst ensemble. Thus the generalization error of the final hypothesis could be bound in the following way. For each ensemble classifier we have that.

$$err_{j,t}(h) \leq \frac{1}{M} \sum_{i=1}^{M} I(y_i h_{j,t}(\bar{x}_i) \leq \theta) + O\left(\sqrt{\frac{\log M \log |H|}{M\theta^2}} + \log \frac{1}{\delta}\right).$$

Thus the stated above proves the following theorem for generalization error bound for distributed boosting algorithm stated above.

Theorem 2.1. Suppose we are a distributed system, with given horizontally distributed and homogeneous data, then for the distributed boosting algorithm holds true the following generalization error bound $err_{distributed}(h) \leq \max_{j,t} err_{j,t}(h)$.

In [12] there are results, which prove that boosting algorithm itself can be parallelized to some point and there is lower bound form minimum rounds

regardless of processing cores. Below we state the results from [12], but shortly, concerning parallel boosting, it states that at least $\Omega(\ log(1/\varepsilon)/\delta^2)$ stages of boosting needed for boosting δ-adantage (i.e classifier with $\frac{1}{2}+\delta$ error) weak learner for achieving classification accuracy of $1-\varepsilon$, even in case when any number of copies of weak classifier is used.

Definition 2.1 *A δ-advantage weak learner L is an algorithm that is given access to a source of independent random labeled examples drawn from an (unknown and arbitrary) probability distribution D over labeled examples $\{(x, f(x))\}x \in X$. L mustreturn a weak hypothesis $h: X \to \{-1,1\}$ that satisfies $\Pr_{(x,f(x))\leftarrow P}[h(x) = f(x)] \geq 1/2 + \delta$. Such an h is said to have advantage δw.r.t. P.*

The aim of boosting is to choose $D_1, D_2, \ldots$ distributions on given examples and then construct $h_1, h_2, \ldots$ weak hypothesis, in the final stage it combines all weak hypothesis to output h hypothesis, which lower error under$D$. The following is the definition of sequential booster from [12].

Definition 2.2 (Sequential booster*) A T-stage sequential boosting algorithm is defined by a sequence $a_1, \ldots, a_T$ of functions $a_t: \{-1,1\}^t \to [0,1]$ and a (randomized) Boolean function $h: \{-1,1\}T \to \{-1,1\}$. In the $t-th$ stage of boosting, the distribution $D_t$over labeled examples that is given to the weak learner by the booster is obtained from $D$by doing rejection sampling according to a_t. More precisely, a draw from D_t is made as follows: draw (x, f (x)) from $D$and compute the value $p_x := a_t(h_1(x), \ldots, h_{t-1}(x), f(x))$. With probability p_x accept $(x, f(x))$as the output of the draw from D_t, and with the remaining $1-p_x$ probability reject this $(x, f(x))$and try again. In stage t the booster gives the weak learner access to $D_t$as defined above, and the weak learner generates a hypothesis $h_t$that has advantage at least εw.r.t. D_t . Together with $h_1, \ldots, h_{t-1}$, this h_t enables the booster to give the weak learner access to D_{t+1}in the next stage. After T stages, weak hypotheses $h_1, \ldots, h_t$have been obtained from the*

weak learner. The final hypothesis of the booster is $H(x) := h(h_1(x), \ldots, h_T(x))$, *and its accuracy is*

$$\min_{(h_1,\ldots,h_T\ (x,f(x)\leftarrow D)} Pr[H(x) = f(x)],$$

where the minimum is taken over all sequences $h_1, \ldots, h_T$ *of* T *weak hypotheses subject to the condition that each* h_t *has advantage at least* ε *w.r.t.* D_t.

In fact, most of the boosting algorithms use a priori this definition of boosting in order to train a weak classifier. The results obtained from this algorithm need at least $\Omega(\ log(1/\varepsilon)/\delta^2)$ in order to train $\delta - advantage$ weak learner with $1-\varepsilon$ accuracy.

The parallel boosting is a generalization of sequential boosting algorithm. The method is implemented as follows. The main principle is to run weak learner many times in multiple probability distributions. From the boosting definition itself it means that in the t stage distributions that are used depend only on weak hypothesis obtained in earlier stages of algorithm. So it is obvious that at least some sequential behavior is observed, which cannot be avoided, in [12] is given the following definition of parallel boosting

Definition 2.3 (Parallel booster) *A T-stage parallel boosting algorithm with N-fold parallelisms defined by TN functions* $\{a_t, k\}^t \in [T], k \in [N]$ *and a (randomized) Boolean function h, where* $a_{t,k}: \{-1,1\}^{(t-1)N+1} \to [0,1]$ *and* $h: \{-1,1\}^{TN} \to \{-1,1\}$. *In the* t^{th} *stage of boosting the weak learneris run N times in parallel. For each* $k \in [N]$, *the distribution* $D_{t,k}$ *over labeled examples that is givento the* k^{th} *run of the weak learner is as follows: a draw from* $D_{t,k}$ *is made by drawing a labeled example(x, f (x)) from* D , *computing the value* $p_x: a_{t,k}k(h_{1,1}(x), \ldots, h_{t-1,N}(x), f(x))$, *and accepting* $(x, f(x))$ *as the output of the draw from* $D_{t,k}$ *with probability* p_x *(and rejecting it and trying again otherwise). In stage* t, *for each* $k \in [N]$ *the booster gives the weak learner access to* $D_{t,k}$ *as definedabove and the weak learner generates a hypothesis* $h_{t,k} that$ *has advantage at least* δ *w.r.t.* $D_{t,k}$.*Together with the weak hypotheses*$\{h_s, j\}$ $s \in [t-1]$, $j \in [N]$ *obtained in earlier stages, these* $h_{t,k}$ *'s*

enable the booster to give the weak learner access to each $D_{t+1,k}$ *in the next stage.After* T *stages,* TN *weak hypotheses* $\{h_{t,k}\} t \in [T], k \in [N]$ *have been obtained from the weak learner. The final hypothesis of the booster is* $H(x) := h(h_{1,1}(x), \ldots, h_{T,N}(x))$*, and its accuracy is*

$$\min_{ht,k} \Pr_{(x,f(x))\leftarrow D}[H(x) = f(x)]$$

where the minimum is taken over all sequences of TN weak hypotheses subject to the condition that each $h_{t,k}$ *has advantage at least* δ *w.r.t.* $D_{t,k}$.

N is the number of processors available simultaneously in algorithm. The number of stages of boosting algorithm is calculated as number of branches of the decision tree, and the parallel case in fact affects only number of nodes and not the depth of decision tree. Below is stated the theorem about lower bound for parallel boosting.

In [13] discussed the problems in distributed systems concerning communication costs, privacy and given lower and upper bounds for communication complexity for multiple distributed classification methods. Results include both agnostic and PAC-learning method analysis. Concerning ensemble methods boosting is reviewed as given lower and upper bounds for communication cost functions. Mainly, boosting algorithms are weight based and use different weighting methods for learning, some of them discussed above. In distributed system the communication is required to build a weak learner which would involve data points from all participants. The principle is as follows, in first step, enough data is sent from each participant to computation center so that weak hypothesis is constructed. Then it is given to all participants and each of them re-weights their examples according to the principle of traditional ensemble learning, and then weighted examples sent to the data center to construct next weak hypothesis, in the final step all obtained hypothesis are combined to for final hypothesis.

Theorem 2.3 Any class H can be learned to error ε in $O\left(log\frac{1}{\varepsilon}\right)$ rounds and $O(d)$ examples plus $O(klogd)$ bits of communication per round. For any $c \geq 1$,

H can be learned to error ε in $O(c)$ rounds and $O(\frac{d}{\varepsilon^{\frac{1}{c}}} log\frac{1}{\varepsilon})$ examples plus $O(klog\frac{d}{\varepsilon})$ bits communicated per round.

Consequently, any PAC learnable class can be learned within $O\,(log\frac{1}{\varepsilon})$ number of rounds and a total of number of examples used is $(dlog\frac{1}{\varepsilon})$ plus extra number of bits, which is very small.

In each round computation center receives $O((d/\delta)\,log(1/\delta))$ distributed according to D' distributed distribution. The variance between D' and the distribution obtained with boosting algorithm is at most $\frac{\delta}{2}$, which means overall error will be $\frac{\delta}{2}+\frac{\delta}{2}=\delta$where $\frac{\delta}{2}$ is the error of h_j weak learner. Then in every step in round 5 all weights are recalculated to ensure weight normalizations are correct $(\frac{w_{i,j}}{w_j})$. And so the overall examples sent to at each round is $O((d/\delta)\,log(1/\delta))$ plus $O(rlog\frac{d}{\varepsilon})$ for sending numbers$n_{i,j}$and $w_{i,j}$, and the numbers of rounds is $r(\varepsilon,\delta)$. It is clear that it depends only from $\varepsilon$and δ.

Conclusion

The algorithms we reviewed above are examples of the ones that are not designed for big and distributed data, but these algorithms can be transformed in such way, so that they can be used in distributed environment as mentioned above. which are also popular and applicable. For instance, in case of ensemble we have seen that it does not matter what kind of model we have, we can boost it anyway, it only a matter of choosing weak learner. There are methods for combing results, which proved to be robust enough. For instance, majority voting, this could be employed to combine the results of models trained independently. This method also gives the advantage of data privacy, because no transfer of data is made, and only trained models are transferred, which is secure as the model cannot reveal the data from which it was trained. From the higher level of perspective, we can emphasize the main metrics that we have; to measure the algorithms in distributes environments, here are some of them.

Scalability and reduced communication overhead. The communication is vital to solve the problems in distributed environments. The optimization here is to reduce the amount of data transferred among nodes(participants). For instance, approach in [14] relies on in-network processing with messages exchanged only among single-hop neighboring nodes. And this keeps the communication overhead per node at a relatively cheap level within its neighborhood. In fully connected approaches however, nodes consume increased resources to reach the fully connected as the coverage area grows. Another point is that, for example, [14] differs from [19] in a way that it does not exchange support vectors and incurs a fixed cost for inter-node communications at each iteration regardless of the size of the local training sets, which very important in case of big data sets.

Robustness to isolated points of failure, which means whether the algorithm continues to work if a single node stops working, while others are connected, and if yes, in what accuracy it will converge to the aimed function, or will in converge at all or not?

References

1. M. Mohri, A. Rostamizadeh, A. Talwalkar Foundations of Machine Learning, the MIT Press, 2012.
2. L.Breiman, Bagging Predictors, Machine Learning 24(2): 123–140, 1996.
3. J. R. Quinlan, Bagging, Boosting, and C4.5, AAAI Press / The MIT Press, Vol. 1, 725–730, 1996.

A. Lazarevic, Z. Obradovic, The Distributed Boosting Algorithm, Knowledge Discovery and Data Mining: 311-316, 2001.

4. P.K. Murphy, Machine Learning, A Probabilistic Perspective, 1st edition, 2012.
5. J. R. Quinlan, Review of C4.5: Programs for Machine Learning, Machine Learning 16(3):235-240, 1994.
6. J.R. Quinlan, Induction of Decision Trees,Machine Learning 1: 81-106, 1986.

7. M. Y. Moshkov Conditional tests, *Problems of Cybernetics,* 40: 131-170 (in Russian) 1983.

8. L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Belmont, CA: Wadsworh, 1984.

9. N. Smart, Cryptography: An Introduction, 3rd Edition:35-165.

10. M. Balcan, A. Blum, S. Fine, Y. Mansour, Distributed Learning, Communication Complexity and Privacy, JMLR: Workshop and Conference Proceedings vol.1-22, 2012.

11. H. Papadimitriou, Computational Complexity, Addison-Wesley, 1993.

B. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating Noise to Sensitivity in Private Data Analysis, 3rd Theory of Cryptography Conference, pp. 265–284, New York, USA, Mar. 4-7, 2006.

12. K. Chaudhuri, C. Monteleoni, Privacy-Preserving Logistic Regression, In Advances in Neural Information Processing Systems, 2008.

13. Y. Mansour AND R. L. Rivest, Results on Learnability and the Vapnik-Chervonenkis Dimension, Information and Computation 90, 33-49, 1991.

Authors' Information

Karlen Mkrtchyan *– Institute for Informatics and Automation Problems of the National Academy of Sciences of Armenia, PhD candidate; 1 P.Sevak str., Yerevan 0014, Armenia;*

e-mail*: mkrthcyan.karlen@icloud.com*

Major Fields of Scientific Research*: Distributed Computation, Distributed Machine Learning*