DISTRIBUTED RESTARTING NEWTONCG METHOD FOR LARGE-SCALE EMPIRICAL RISK MINIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we propose a distributed damped Newton method in which sample size is gradually increasing to quickly obtain a solution whose empirical loss is under satisfactory statistical accuracy. Our proposed method is multistage in which the solution of one stage serves as a warm start for the next stage which contains more samples (including the samples in the previous stage). This overall multistage algorithm reduce the number of passes over data. Moreover, our algorithm in nature is easy to be distributed and shares the strong scaling property indicating that acceleration is always expected by using more computing nodes. Various iteration complexity results regarding descent direction computation and stopping criteria are analyzed under convex setting. Our results of experiments illustrate that the proposed algorithm can outperform other comparable methods for training machine learning tasks including neural networks.

1 INTRODUCTION

In the field of machine learning, solving the expected risk minimization problem has received lots of attentions over decades, which is in the form

$$\min_{w\in\mathbb{R}^d} L(w) = \min_{w\in\mathbb{R}^d} \mathbb{E}_z[f(w,z)], \tag{1}$$

where z is a $d+1$ dimensional random variable containing both feature variables and a response variable. $f(w,z)$ is a loss function with respect to w and any fixed value of z. In most practical problems, the distribution of z is either unknown or leading great difficulties to evaluate the expected loss. One general idea is to estimate the expectation with a statistical average over a large number of independent and identically distributed data samples of z, which is denoted by $\{z_1, z_2, \ldots, z_N\}$ where N is the total number of samples. Thus, the problem in (1) can be rewritten as the Empirical Risk Minimization (ERM) problem

$$\min_{w\in\mathbb{R}^d} L_N(w) = \min_{w\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} f_i(w), \tag{2}$$

where $f_i(w) = f(w, z_i)$.

A lot of studies have been done on developing optimization algorithms to find an optimal solution of above problem under different setting. For example, Beck & Teboulle (2009); Nesterov (2013); Drusvyatskiy et al. (2016); Ma et al. (2017) are some of the gradient-based methods which require at least one pass over all data samples to evaluate the gradient $\nabla L_N(w)$. As the sample size N becomes larger, these methods would be less efficient compared to stochastic gradient methods where the gradient is approximated based on a small number of samples Johnson & Zhang (2013); Roux et al. (2012); Defazio et al. (2014); Shalev-Shwartz & Zhang (2013); Konečnỳ & Richtárik (2017); Nguyen et al. (2017).

Second order methods are well known to share faster convergence rate by utilizing the Hessian information. Recently, several papers Byrd et al. (2015); Schraudolph et al. (2007); Mokhtari & Ribeiro (2015) have studied how to apply second orders methods to solve ERM problem. However, getting the inverse of Hessian matrix of a good approximation of it is always quite expensive, leading to a significant difficulty on applying these methods on large scale problems.
Following the idea of adaptive sample size discussed in Mokhtari & Ribeiro (2017); Eisen et al.

(2017); Mokhtari & Ribeiro (2016), the complexity of Newton's method can be reduced (Mokhtari & Ribeiro, 2016) if the dimension d is small, but it is impractical to compute the Hessian inverse for large dimensional problems. In order to decrease the cost of computing the Hessian inverse, Eisen et al. (2017) proposed the k-Truncated Adaptive Newton (k-TAN) approach. In this method, the inverse of such approximated Hessian is calculated by increasing the sample size adaptively and using a rank-k approximation of the Hessian. The cost per iteration is $\mathcal{O}((\log k+n)d^2)$. Again, note that either when d is large, or in the case when k is close to d, this method can be quite inefficient.

In this paper, we propose an increasing sample size second-order method which solves the Newton step in ERM problems more efficiently. Our proposed method, called Restarting NewtonCG (RNC) method, starts with a tiny number of samples and only considering the corresponding empirical risk based on these samples. This problem is solved up to some accuracy, and the solution of this stage is a warm start for the next stage in which we solve the next empirical risk with a larger number of samples, which contains all the previous samples. Such procedure is run iteratively until either all the samples have been included, or we find that it is unnecessary to further increase the sample size. Our RNC method combines the idea of increasing sample size and the inexact damped Newton method discussed in Zhang & Xiao (2015) and Ma & Takáč (2016). Instead of solving the Newton system directly, we apply preconditioned conjugate gradient (PCG) method as the solver for each Newton step. We show the required number of PCG steps in order to reach the statistical accuracy of the full dataset.

Also, it is always a challenging problem of running first order algorithms such as SGD and Adam Kingma & Ba (2014) in a distributed way. However, our algorithm is designed naturally to be easily parallelized and shares the strong scaling property. While splitting the gradient and Hessian-vector product computation based on local data stored across different machines, it is always expected to get extra acceleration via increasing the number of computational nodes. We show that, under distributed setting, our RNC algorithm is communication efficient in both theory and experiments.

We organize this paper as following. In Section 2, we introduce the necessary assumptions and the definition of statistical accuracy. Section 3 describes the proposed algorithm and its distributed version. Section 4 explores the theoretical guarantees on complexity. In Section 5, we demonstrate the outstanding performance of our algorithm in practice. Section 6 is the concludes our contribution.

2 Problem Formulation

In this paper, we focus on finding the optimal solution w^* of (1). As described earlier, we are trying to find a solution for the empirical loss function $L_N(w)$, which is the statistical mean over N samples. Now, consider the empirical loss $L_n(w)$ associated with $n \leq N$ samples. In Bousquet & Bottou (2007) and Bottou (2010s), the error between the expected loss and the empirical loss L_n is calculated. In Mokhtari & Ribeiro (2016), it is mentioned that L_n approximates the expected loss with statistical accuracy V_n for all $w \in \mathbb{R}^d$ with high probability(w.h.p),

$$\sup_{w\in\mathbb{R}^d} |L(w) - L_n(w)| \leq V_n. \tag{3}$$

In other words, there exists a constant δ such that the inequality (3) holds with probability of at least $1-\delta$. Generally speaking, statistical accuracy V_n depends on n (although it depends on δ too, but for simplicity in notation we just consider the size of the samples), and is of the order $V_n = \mathcal{O}(\frac{1}{n^\gamma})$ where $\gamma \in [0.5, 1]$ (Vapnik (2013); Bousquet (2002); Bartlett et al. (2006)).

For problem (2), if we've found an approximate solution w_n which satisfies the inequality $L_n(w_n) - L_n(\hat{w}_n) \leq V_n$, where $\hat{w}_n$ is the true minimizer of L_n, it is not necessary to go further and find a better solution (a solution with less optimization error). The reason comes from the fact that for a more accurate solution the summation of estimation and optimization errors does not become smaller than V_n. Therefore, when we say that w_n is an V_n-suboptimal solution for the risk L_n, it means that $L_n(w_n) - L_n(\hat{w}_n) \leq V_n$. In other words, w_n solves problem (2) to reach its statistical accuracy.

It is crucial to note that if we add an additional term in the magnitude of V_n to the empirical loss L_n, the new solution is also in the similar magnitude as V_n to the expected loss L. Therefore, we can regularize the non-strongly convex loss function L_n by $\frac{cV_n}{2}\|w\|^2$ and consider it as the following

problem:

$$\min_{w\in\mathbb{R}^d} R_n(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) + \frac{cV_n}{2}\|w\|^2. \tag{4}$$

The noticeable feature of the new empirical risk R_n is that R_n is cV_n-strongly convex, where c is a positive constant. Thus, we can utilize any practitioner-favorite algorithm. Specifically, we are willing to apply the inexact damped Newton method, which will be discussed in the next section. Due to the fact that a larger strong-convexity parameter leads to a faster convergence, we could expect that the first few steps would converge fast since the values of cV_n in these steps are large (larger statistical accuracy), as discussed in Theorem 1. From now on, when we say w_n is an V_n-suboptimal solution of the risk R_n, it means that $R_n(w_n) - R_n(w_n^*) \leq V_n$, where w_n^* is the true optimal solution of the risk R_n. Our final aim is to find w_N which is V_N-optimal solution for the risk R_N which is the risk over the whole dataset.

3 RESTARTING NEWTONCG METHOD WITH INCREASING SAMPLE SIZE

The inexact damped Newton method, which is discussed in the study of Zhang & Xiao (2015), is to find the next iterate based on an approximated Newton-type update. It has two important differences comparing to exact Newton method. First, as it clear from the word "damped", the learning rate of the inexact damped Newton type update is not 1, since it depends on the approximation of Newton decrement. The second distinction is that there is no need to compute exact Newton direction (which is very expensive to calculate in one step). Alternatively, an approximated inexact Newton type direction is calculated by applying an iterative process to obtain a direction with desirable accuracy under some measurement.

In order to utilize the important features of ERM, we combine the idea of increasing sample size and the inexact damped Newton method. In our proposed method, we start with handling a small number of samples, assume m_0 samples. We then solve its corresponding ERM to its statistical accuracy, i.e. V_{m_0}, by inexact damped Newton algorithm. At the next iteration, we increase the number of samples geometrically with rate of α, i.e., αm_0 samples. The approximated solution of the previous ERM can be used as a warm start point to find the solution of the new ERM. The sample size increases until it equals the number of full samples.

Consider the iterate w_m within the statistical accuracy of the set with m samples, i.e. $\mathcal{S}_m$ for the risk R_m. The inexact damped Newton method with increasing sample size finds the iterate w_n which is V_n-suboptimal solution for the sample set $\mathcal{S}_n$, i.e. $R_n(w_n) - R_n(w_n^*) \leq V_n$ after T_n iterations. We initialize $\tilde{w}_0 = w_m$ and consider the following update:

$$\tilde{w}_{k+1} = \tilde{w}_k - \frac{1}{1+\delta_n(\tilde{w}_k)} v_k, \tag{5}$$

where v_k is ϵ_k-Newton direction defined as

Definition 1 (ϵ_k-Newton direction)**.**

$$\|\nabla^2 R_n(\tilde{w}_k)v_k - \nabla R_n(\tilde{w}_k)\| \leq \epsilon_k. \tag{6}$$

Note that ϵ_k has a crucial effect in the speed of the algorithm. We use preconditioned CG (PCG) (by considering the preconditioned matrix $P = \tilde{H}_n + \mu I$, where $\tilde{H}_n = \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}} \nabla^2 R_n^i(w)$ and $\mathcal{A} \subset \mathcal{S}_n$) in order to find the vector v_k, which is an approximate solution of the system $P^{-1}\nabla^2 R_n(\tilde{w}_k)v_k = P^{-1}\nabla R_n(\tilde{w}_k)$. Moreover, $\delta_n(\tilde{w}_k) = \sqrt{v_k^T \nabla^2 R_n(\tilde{w}_k) v_k}$ is the approximation of (exact) Newton decrement. Also, we have

$$\nabla R_n(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w) + cV_n w, \quad \nabla^2 R_n(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla^2 f_i(w) + cV_n I. \tag{7}$$

Thus, after T_n-PCG iterations, $w_n = \tilde{w}_{T_n}$ (see Theorem 1). Also, because of $\epsilon_k = 0$, v_k is the exact Newton direction, and the update in (5) is the exact damped Newton step. Furthermore, in Theorem 1 we show that the number of total PCG iterations to reach V_N-optimal solution for the risk R_N is T_N. It means that when we start with the iterate w_{m_0} with corresponding m_0 samples, after T_N PCG iterations, we reach the point w_N with statistical accuracy of V_N for the whole dataset. Our proposed method is summarized in Algorithm 1. In the inner for loop of Algorithm 1, in order

to calculate the approximate Newton direction and approximate Newton decrement, we use PCG algorithm which is shown in Algorithm 2.

Thus, after T_n-PCG iterations, $w_n = \tilde{w}_{T_n}$ (see Theorem 1). Also, we can note that when $\epsilon_k = 0$, then v_k is the exact Newton direction, and the update in (5) is the exact damped Newton step. Furthermore, in Theorem 1 we show that the number of total PCG iterations to reach V_N-optimal solution for the risk R_N is T_N. It means that when we start with the iterate w_{m_0} with corresponding m_0 samples, after T_N PCG iterations, we reach the point w_N with statistical accuracy of V_N for the whole dataset.

Our proposed method is summarized in Algorithm 1. In the inner for loop of Algorithm 1, in order to calculate the approximate Newton direction and approximate Newton decrement, we use PCG algorithm which is shown in Algorithm 2.

Stopping Criterion Here we discuss two stopping criterions to fulfill the 10^{th} line from Algorithm 1. At first, considering w_n^* is unknown in practice, we can use strong convexity inequality as $R_n(\tilde{w}_k) - R_n(w_n^*) \leq \frac{1}{2cV_n}\|\nabla R_n(\tilde{w}_k)\|^2$ to find a stopping criterion for the inner loop, which satisfies $\|\nabla R_n(\tilde{w}_k)\| < (\sqrt{2c})V_n$. However, this stopping criterion can be too conservative in practice. Another stopping criterion is discussed by Zhang & Xiao (2015), using the fact that the risk R_n is self-concordant. This criterion can be written as $\delta_n(\tilde{w}_k) \leq (1-\beta)\sqrt{V_n}$ (see section 7.1), where $\beta \leq \frac{1}{20}$. The later stopping criterion implies that $R_n(\tilde{w}_k) - R_n(w_n^*) \leq V_n$ whenever $V_n \leq 0.68^2$. To compare these two criterions, the later criterion is more practical due to the fact that we have $\delta_n(\tilde{w}_k)$ in every iteration. While we need to calculate the gradient of the risk R_n in each iteration of the inner loop to use the first criterion.

Algorithm 1 Restarting NewtonCG algorithm

1: Initialization: Sample size increase constant α, initial sample size $n = m_0$ and $w_n = w_{m_0}$ with $\|\nabla R_n(w_n)\| < (\sqrt{2c})V_n$
2: **while** $n \leq N$ **do**
3: Update $w_m = w_n$ and $m = n$
4: Increase sample size: $n = \max\{\alpha m, N\}$
5: Set $\tilde{w}_0 = w_m$ and set $k = 0$
6: **repeat**
7: Calculate v_k and $\delta_n(\tilde{w}_k)$ by **Algorithm 2 PCG**
8: Set $\tilde{w}_{k+1} = \tilde{w}_k - \frac{1}{1+\delta_n(\tilde{w}_k)}v_k$
9: $k = k + 1$
10: **until** a stopping criterion is satisfied
11: Set $w_n = \tilde{w}_k$
12: **end while**

Algorithm 2 PCG - Algorithm 2 in Zhang & Xiao (2015)

1: Input: $\tilde{w}_k \in \mathbb{R}^d$, ϵ_k, and $\mu \geq 0$
2: Let H denote the Hessian $\nabla^2 R_n(\tilde{w}_k)$ and $P = \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}} \nabla^2 R_n^i(\tilde{w}_k) + \mu I$
3: Set $r^{(0)} = \nabla R_n(\tilde{w}_k)$, $s^{(0)} = P^{-1}r^{(0)}$, $v^{(0)} = 0$, $u^{(0)} = s^{(0)}$, $t = 0$
4: **repeat**
5: Calculate $Hu^{(t)}$ and $Hv^{(t)}$
6: Compute $\gamma_t = \frac{\langle r^{(t)}, s^{(t)}\rangle}{\langle u^{(t)}, Hu^{(t)}\rangle}$
7: Set $v^{(t+1)} = v^{(t)} + \gamma_t u^{(t)}$, $r^{(t+1)} = r^{(t)} - \gamma_t Hu^{(t)}$
8: Compute $\beta_t = \frac{\langle r^{(t+1)}, s^{(t+1)}\rangle}{\langle r^{(t)}, s^{(t)}\rangle}$
9: Set $s^{(t+1)} = P^{-1}r^{(t+1)}$, $u^{(t+1)} = s^{(t+1)} + \beta_t u^{(t)}$
10: Set $t = t + 1$
11: **until** $\|r^{t+1}\| \leq \epsilon_k$
12: **return** $v_k = v^{(t+1)}$ and $\delta_n(\tilde{w}_k) = \sqrt{v_k^T H v^{(t)} + \gamma_t v_k^T H u^{(t)}}$

Distributed Implementation Similar to the algorithm in Zhang & Xiao (2015), Algorithm 1 and 2 can also be implemented in a distributed environment. Suppose the entire dataset is stored across K machines, i.e., each machine stores N_i data samples such that $\sum_{i=1}^{K} N_i = N$. Under this setting, each iteration in Algorithm 1 can be executed on different machines in parallel with $\sum_{i=1}^{K} n_i = n$, where n_i is the batch-size on i^{th} machine. To implement Algorithm 2 in a distributed way, a broadcast operation is needed at each iteration to guarantee that each machine will share the same $\tilde{w}_k$ value. Moreover, the gradient and Hessian-vector product can be computed locally and later reduce to the master machine. With the increasing of batch size, computation work on each machine will increase while we still have the same amount of communication need. As a consequence, the computation expense will gradually dominate the communication expense before the algorithm terminates. Therefore the proposed algorithm could take advantage of utilizing more machines to shorten the running time of Algorithm 2.

4 Convergence Analysis

In this section, first we define the self-concordant function. This kind of function has the property that its third derivative can be controlled by its second derivative. By assuming that function $f : \mathbb{R}^d \to \mathbb{R}$ has continuous third derivative, we define self-concordant function as follows.

Definition 2. *A convex function $f : \mathbb{R}^d \to \mathbb{R}$ is M_f-self-concordant if for any $w \in dom(f)$ and $u \in \mathbb{R}^d$ we have*

$$|u^T(f^{'''}(w)[u])u| \leq M_f(u^T\nabla^2 f(w)u)^{\frac{3}{2}}, \tag{8}$$

where $f^{'''}(w)[u] := \lim_{t\to 0} \frac{1}{t}(\nabla^2 f(w+tu) - \nabla^2 f(w))$. As it is discussed in Nesterov (2013), any self-concordant function f with parameter M_f can be rescaled to become standard self-concordant (with parameter 2). There are many well-known empirical loss functions which are self-concordant such as linear regression, Logistic regression and squared hinge loss. In order to prove our results the following conditions are considered in our analysis.

Assumption 1. *The loss functions $f(w, z)$ are convex w.r.t w for all values of z. In addition, their gradients $\nabla f(w, z)$ are L-smooth*

$$\|\nabla f(w, z) - \nabla f(w', z)\| \leq L\|w - w'\|, \ \forall z. \tag{9}$$

Assumption 2. *The loss functions $f(w, z)$ are self-concordant w.r.t w for all values of z.*

The immediate conclusion of Assumption 1 is that both $L(w)$ and $L_n(w)$ are convex and L-smooth. Also, we can note that $R_n(w)$ is cV_n-strongly convex and $(cV_n + L)$-smooth. As it is discussed in Zhang & Xiao (2015) we use the following auxiliary functions, which will be used in the analysis of the self-concordant functions:

$$\omega(t) = t - \log(1 + t), \qquad t \geq 0. \qquad \text{and} \qquad \omega_*(t) = -t - \log(1 - t), \qquad 0 \leq t < 1. \tag{10}$$

The above functions can be very helpful in analyzing the self-concordant functions. Also, for the risk R_n, the same as Zhang & Xiao (2015) we can define the following auxiliary vectors:

$$\tilde{u}_n(\tilde{w}_k) = [\nabla^2 R_n(\tilde{w}_k)]^{-1/2}\nabla R_n(\tilde{w}_k) \qquad \text{and} \qquad \tilde{v}_n(\tilde{w}_k) = [\nabla^2 R_n(\tilde{w}_k)]^{1/2} v_n. \tag{11}$$

We can note that $\|\tilde{u}_n(\tilde{w}_k)\| = \sqrt{\nabla R_n(\tilde{w}_k)[\nabla^2 R_n(\tilde{w}_k)]^{-1}\nabla R_n(\tilde{w}_k)}$, which is the exact Newton decrement, and, the norm $\|\tilde{v}_n(\tilde{w}_k)\| = \delta_n(\tilde{w}_k)$ which is the approximation of Newton decrement (and $\tilde{u}_n(\tilde{w}_k) = \tilde{v}_n(\tilde{w}_k)$ in the case when $\epsilon_k = 0$).

In the rest of this section, we analyze the upper bound for the number of iterations needed to solve every subproblem up to its statistical accuracy.

We prove a linear convergence rate for our algorithm. We analyze the case when we have w_m which is a V_m-suboptimal solution of the risk R_m, and we are interested in deriving a bound for the number of required iterations, T_n, to ensure that w_n is a V_n-suboptimal solution for the risk R_n. We use the analysis of DiSCO algorithm discussed in Zhang & Xiao (2015) to find the bound for T_n.

Theorem 1. *Suppose that Assumptions 1 and 2 hold. Consider w_m which satisfies $R_m(w_m) - R_m(w_m^*) \leq V_m$ and also the risk R_n corresponding to sample set $\mathcal{S}_n \supset \mathcal{S}_m$ where $n = \alpha m, \ \alpha > 1$. Set the parameter ϵ_k (the error in* (6)*) as following*

$$\epsilon_k = \beta(\tfrac{cV_n}{L+cV_n})^{1/2}\|\nabla R_n(\tilde{w}_k)\|, \tag{12}$$

where $\beta \leq \frac{1}{20}$. Then the variable w_n is an V_n-suboptimal solution for the risk R_n, i.e $R_n(w_n) - R_n(w_n^*) \leq V_n$ if the number of iterations T_n satisfies in the following:

$$T_n \geq \left(\left\lceil \frac{R_n(w_m) - R_n(w_n^*)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2(\frac{2\omega(1/6)}{V_n}) \right\rceil \right) \left(\left\lceil \sqrt{1 + \frac{2\mu}{cV_n}} \log_2 \left(\frac{2(cV_n + L)}{\beta c V_n} \right) \right\rceil \right), \quad \textit{w.h.p.} \tag{13}$$

Here $\lceil t \rceil$ shows the smallest nonnegative integer larger than or equal to t.

By utilizing Theorem 1, the following number of iterations, T_N, is needed to reach the statistical accuracy of V_N of the full training set with high probability:

$$T_N \geq \sum_{i=2}^{|\mathcal{P}|} \left(\left\lceil \frac{R_{\mathcal{P}[i]}(w_{\mathcal{P}[i-1]}) - R_{\mathcal{P}[i]}(w^*_{\mathcal{P}[i]})}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2(\frac{2\omega(1/6)}{V_{\mathcal{P}[i]}}) \right\rceil \right) \left\lceil \sqrt{1 + \frac{2\mu}{cV_{\mathcal{P}[i]}}} \log_2 \left(\frac{2(cV_{\mathcal{P}[i]} + L)}{\beta c V_{\mathcal{P}[i]}} \right) \right\rceil, \tag{14}$$

where $\mathcal{P} = \{m_0, \alpha m_0, \alpha^2 m_0, \ldots, N\}$. Also, based on the result in (13), by considering the risk R_n, we can note that when the strong-convexity parameter for the mentioned risk (cV_n) is large, less number of iterations are needed (or equally faster convergence is achieved) to reach the iterate with V_n-suboptimal solution; and this happens in the first steps.

Corollary 1. *Suppose that Assumptions 1 and 2 hold. By assuming that w_m is V_m-suboptimal solution and also consider the risk R_n corresponding to sample set $\mathcal{S}_n \supset \mathcal{S}_m$ where $n = 2m$. If we set parameter ϵ_k (the error in* (6)*) as* (12)*, then after $\tilde{T}_n$ iterations, where with high probability:*

$$\tilde{T}_n \geq \left(\left\lceil \frac{\left(3 + \left(1 - \frac{1}{2^\gamma}\right)\left(2 + \frac{c}{2}\|w^*\|^2\right)\right) V_m}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2(\frac{2\omega(1/6)}{V_n}) \right\rceil \right) \left(\left\lceil \sqrt{1 + \frac{2\mu}{cV_n}} \log_2 \left(\frac{2(cV_n + L)}{\beta c V_n} \right) \right\rceil \right), \tag{15}$$

we reach the point w_n with statistical accuracy of V_n for the risk R_n. Moreover, after $\tilde{T}_N$ iterations we reach a point with the statistical accuracy of V_N of the full training set:

$$\begin{aligned} \tilde{T}_N \geq & \left(2\log_2 \tfrac{N}{m_0} + \left(\frac{\left(3 + \left(1 - \frac{1}{2^\gamma}\right)\left(2 + \frac{c}{2}\|w^*\|^2\right)\right)}{\frac{1}{2}\omega(1/6)} \frac{1 - (\frac{1}{2^\gamma})^{\log_2 \frac{N}{m_0}}}{1 - \frac{1}{2^\gamma}} V_{m_0} \right) \right. \\ & \left. + \log_2 \tfrac{N}{m_0} \log_2(\tfrac{2\omega(1/6)}{V_N}) \right) \left(\left\lceil \sqrt{(1 + \tfrac{2\mu}{cV_N})} \log_2 \left(\tfrac{2}{\beta} + \tfrac{2L}{\beta c} \cdot \tfrac{1}{V_N} \right) \right\rceil \right), \quad \textit{w.h.p}, \end{aligned} \tag{16}$$

where m_0 is the number of initial sample.

By Corollary 1, we can notice that $\tilde{T}_N = \mathcal{O}(\gamma(\log_2 N)^2 \sqrt{N^\gamma} \log_2 N^\gamma)$, and when $\gamma = 1$, we have $\tilde{T}_N = \mathcal{O}((\log_2 N)^3 \sqrt{N})$, and for $\gamma = 0.5$, the result is $\tilde{T}_N = \mathcal{O}((\log_2 N)^3 N^{\frac{1}{4}})$.

5 Numerical Experiments

In this section, we present numerical experiments on several large real-world datasets to show that our restarting NewtonCG algorithm can outperform other existed methods on solving both convex and non-convex problems. Also, we compare the results from utilizing different number of machines to demonstrate the nice scaling property for our algorithm. All the experiments are performed on a cluster with 24 Xeon E5-2620 CPUs (2.40GHz), and all the algorithms are implemented in Python with PyTorch library. In the plots, we use the pink vertical dashed lines to represent when restarting happens in our NewtonCG algorithm.

Convex case First, we compare our restarting NewtonCG algorithm with two other distributed optimization algorithms CoCoA Smith et al. (2016) and Disco Zhang & Xiao (2015), on solving convex problems. We choose these two algorithms in consideration of attaining a fair comparison between distributed first-order (CoCoA) method and distributed second-order (DiSCO) approach. Binary classification tasks based on two datasets rcv1 and news20 chosen from Chang & Lin (2011) are solved using logistic regression model. We choose this two datasets following the principle from Zhang & Xiao (2015), where those two datasets show different relations between number of features and number of data samples (larger and smaller). The empirical loss function we are trying to minimize is stated as in (4). We use logistic loss function defined as $f_i(w) := \log(1 +$

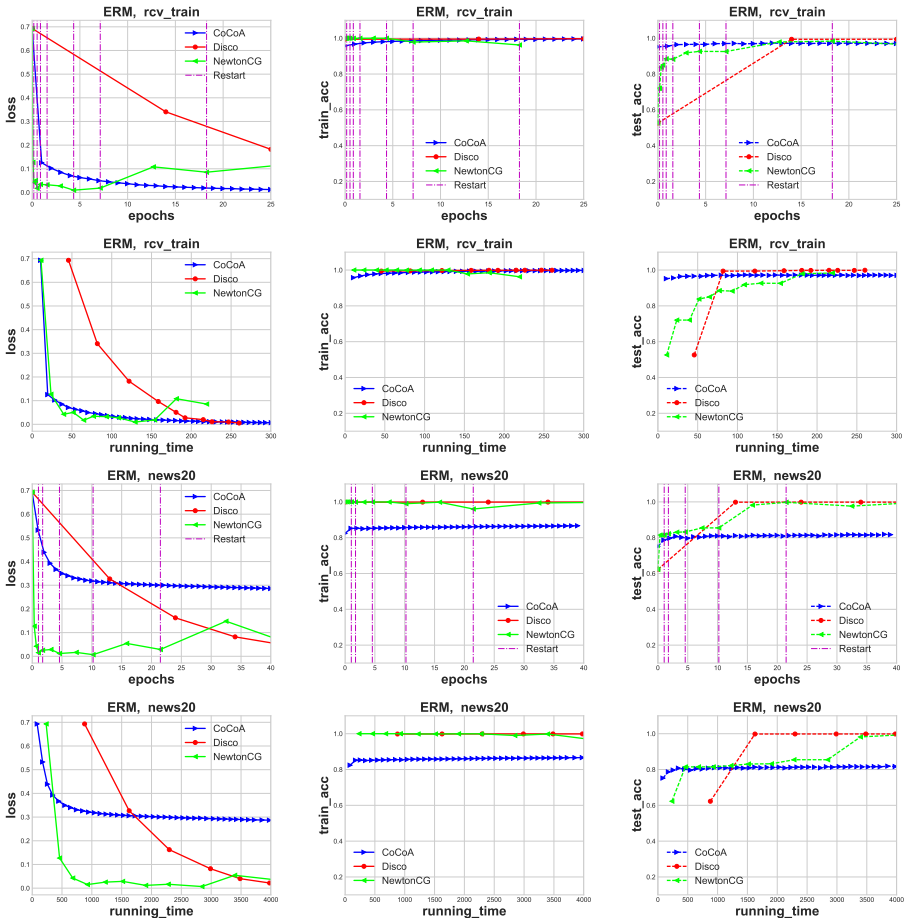

Figure 1: Restarting Newton-CG v.s. DiSCO v.s CoCoA for Logistic Regression

$\exp(-y_i w^T x_i))$, where $x_i \in \mathbb{R}^d$ is data sample and $y_i \in \{-1, 1\}$ are binary label. Note that there is a fixed regularization parameter 10^{-6} in DiSCO and CoCoA, while our restarting NewtonCG has Regularization of $1/\sqrt{m}$ which depends on the size of samples m.

We run all these three algorithms using 8 nodes. The starting batch-size on each node for restarting NewtonCG is set to 16 for a faster beginning, while other two will go over the whole dataset at each iteration. For restarting NewtonCG implementation, number of samples used to form the new ERM loss are doubled from previous iteration after each restarting. Furthermore, restarting happens whenever norm of loss gradient is lower than $1/\sqrt{m}$.

From Figure 1, we observe consistently that the restarting NewtonCG algorithm has a better performance over the other two in the begin stages. Both loss value and training accuracy under our restarting NewtonCG algorithm converges to optimality by passing a very small number of samples, which suggests that the restarting NewtonCG can find a good solution in a warm starting manner. Compared with DiSCO, our restarting approach helps to get rid of spending too much computation at the beginning iterations, where second order methods are usually less efficient than first order methods. Also, our algorithm can still converge fast when we are close to optimal solution, while first order methods become weak since the gradient becomes more and more close to zero.

Non-convex case Even though the iteration complexity analysis only covers the convex case, we want to point out that our algorithm is also able to handle nonconvex problems efficiently. In this section, we compare the performance of increasing sample size method with the well known Adam Kingma & Ba (2014) method on solving convolution neural network. We do experiments on the standard image classification dataset Mnist with a 5-layer convolutional neural network. In Fig-

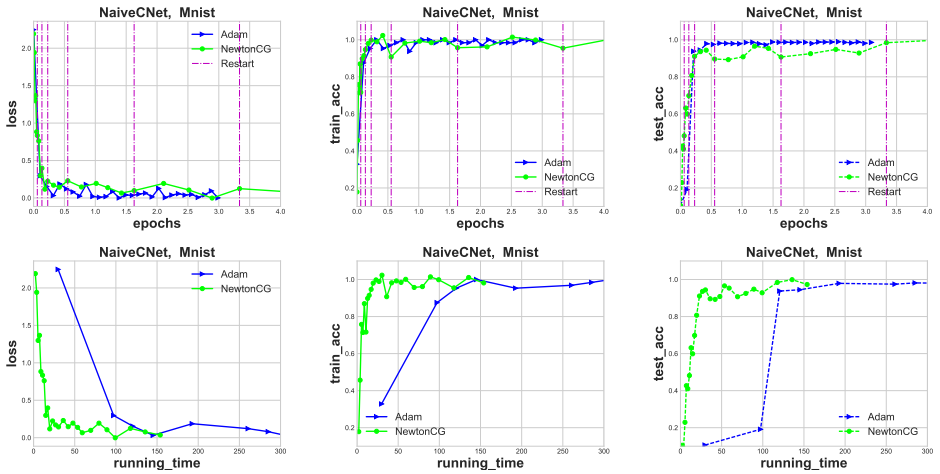

Figure 2: Restarting Newton-CG v.s. Adam using NaiveCNet on Mnist dataset

ure 2, we compare Adam with our restarting NewtonCG approach. The Adam is implemented using bulid-in optimizer in pytorch library, and we choose the best batch-size as 64 from the range $\{16, 32, 64, 128\}$ and initial learning rate as 0.005 from the range $\{0.001, 0.005, 0.01, 0.05\}$. Regarding our restarting NewtonCG, we experiment on 32 nodes. The initial batch size are set to 8 for each node, i.e, set 256 as our initial total batch size across all nodes. As it is clear shown in Figure 2, our restarting NewtonCG with 32 nodes could outperform serial Adam. Note that Adam, i.e., the stochastic first-order method variant, can not be distributed easily, since a small batch size is require to have start-of-art performance He et al. (2016). While we could further improve our restarting approach by utilizing more nodes.

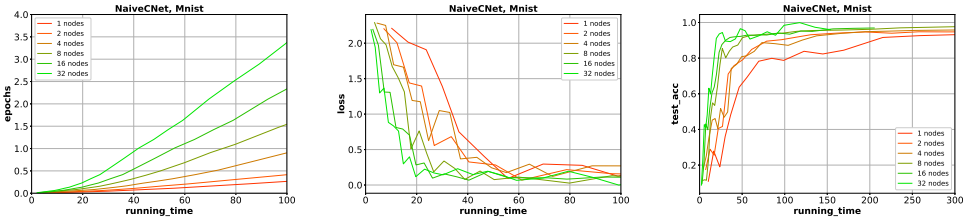

Figure 3: Performance of restarting NewtonCG algorithm with different computing threads.

Strong scaling As the last experiment, we demonstrate that our restarting NewtonCG algorithm shares a strong scaling property. As shown in Figure 3, whenever we increase the number of nodes, we can always obtain acceleration towards optimality. The leftmost plot in Figure 3 shows that the speed of passing over data increases along the increase of number of nodes used, since distributed computation among nodes will obtain the gradient and Hessian-vector product faster. As it is shown in rightmost plot in Figure 3, to reach 0.96 testing accuracy, it is about 12 times slower by only using 1 node than using 32 nodes.

6 Conclusion

We propose a restarting NewtonCG method with increasing sample size strategy to solve the expected risk minimization problem. Our algorithm can converge to a low statistical accuracy in very few epochs and also be implemented in a distributed environment naturally. We show linear convergence rate for convex empirical risk minimization under mild assumptions. Numerical experiments are presented to demonstrate the advantages of our proposed algorithm on both convex and non-convex problems.

REFERENCES

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2009.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. *In Proc. of COMPSTAT*, 2010s.

olivier Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, Biologische Kybernetik, 2002.

Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. *NIPS*, 2007.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Richard H. Byrd, Samantha L. Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM J. Optim., 26(2), 10081031. (24 pages)*, 2015.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *NIPS*, 2014.

Dmitriy Drusvyatskiy, Maryam Fazel, and Scott Roy. An optimal first order method based on optimal quadratic averaging. *arXiv preprint arXiv:1604.06543*, 2016.

Mark Eisen, Aryan Mokhtari, and Alejandro Ribeiro. Large scale empirical risk minimization via truncated adaptive newton method. *arXiv preprint arXiv:1705.07957*, 2017.

Xi He, Dheevatsa Mudigere, Mikhail Smelyanskiy, and Martin Takáč. Large scale distributed hessian-free optimization for deep neural network. *arXiv preprint arXiv:1606.00511*, 2016.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *NIPS*, 2013.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jakub Konečnỳ and Peter Richtárik. Semi-stochastic gradient descent methods. *Frontiers in Applied Mathematics and Statistics*, 2017.

Chenxin Ma and Martin Takáč. Distributed inexact damped newton method: Data partitioning and load-balancing. *arXiv preprint arXiv:1603.05191*, 2016.

Chenxin Ma, Naga Venkata C Gudapati, Majid Jahani, Rachael Tappenden, and Martin Takáč. Underestimate sequences via quadratic averaging. *arXiv preprint arXiv:1710.03695*, 2017.

Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory bfgs. *Journal of Machine Learning Research*, 16(1):3151–3181, 2015.

Aryan Mokhtari and Alejandro Ribeiro. Adaptive newton method for empirical risk minimization to statistical accuracy. *arXiv preprint arXiv:1605.07659*, 2016.

Aryan Mokhtari and Alejandro Ribeiro. First-order adaptive sample size methods to reduce complexity of empirical risk minimization. *arXiv preprint arXiv:1709.00599*, 2017.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Lam Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. *arXiv preprint arXiv:1703.00102*, 2017.

Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems*, pp. 2663–2671, 2012.

Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pp. 436–443, 2007.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 2013.

Virginia Smith, Simone Forte, Chenxin Ma, Martin Takac, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*, 2016.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer, 2013.

Yuchen Zhang and Lin Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. *arXiv preprint arXiv:1501.00263*, 2015.

7 APPENDIX

Before talking about the main results, two following lemmas are used in our analysis.

Lemma 1. *(Lemma 4 in Zhang & Xiao (2015)) Suppose Assumption 1 holds and* $\|\tilde{H}_n - \nabla^2 R_n(\tilde{w}_k)\| \leq \mu$. *Then, Algorithm 2, after* T_μ *iterations calculates* v_n *such that* $\|\nabla^2 R_n(\tilde{w}_k)v_n - \nabla R_n(\tilde{w}_k)\| \leq \epsilon_k$, *where*

$$T_\mu = \Big\lceil \sqrt{1+\tfrac{2\mu}{cV_n}} \log\Big(\frac{2\sqrt{\frac{cV_n+L}{cV_n}}\|\nabla R_n(\tilde{w}_k)\|}{\epsilon_k}\Big)\Big\rceil. \tag{17}$$

Lemma 2. *(Proposition 5 in Mokhtari & Ribeiro (2016)) Consider the sample sets* $\mathcal{S}_m$ *with size* m *and* $\mathcal{S}_n$ *with size* n *such that* $\mathcal{S}_m \subset \mathcal{S}_n$. *Let* w_m *is* V_m*-suboptimal solution of the risk* R_m. *If assumptions 1 and 2 hold, then the following is true:*

$$R_n(w_m) - R_n(w_n^*) \leq V_m + \tfrac{2(n-m)}{n}(V_{n-m}+V_m) + 2(V_m - V_n) + \tfrac{c(V_m-V_n)}{2}\|w^*\|^2, \;\; w.h.p. \tag{18}$$

If we consider $V_n = \mathcal{O}(\frac{1}{n^\gamma})$ where $\gamma \in [0.5, 1]$, and assume that $n = 2m$ (or $\alpha = 2$), then (18) can be written as:

$$R_n(w_m) - R_n(w_n^*) \leq \Big[3 + \Big(1 - \tfrac{1}{2^\gamma}\Big)\Big(2 + \tfrac{c}{2}\|w^*\|^2\Big)\Big]V_m. \tag{19}$$

7.1 PRACTICAL STOPPING CRITERION

As a result of Theorem 1 in the study Zhang & Xiao (2015), we have:

$$(1-\beta)\|\tilde{u}_n(\tilde{w}_k)\| \leq \|\tilde{v}_n(\tilde{w}_k)\| \leq (1+\beta)\|\tilde{u}_n(\tilde{w}_k)\|, \tag{20}$$

where $\beta \leq \frac{1}{20}$. Also, by the equation in (11), we know that $\|\tilde{v}_n(\tilde{w}_k)\| = \delta_n(\tilde{w}_k)$. As it is discussed in the section 9.6.3. of the study Boyd & Vandenberghe (2004), we have $\omega_*(t) \leq t^2$ for $0 \leq t \leq 0.68$.

According to Theorem 4.1.13 in the study Nesterov (2013), if $\|\tilde{u}_n(\tilde{w}_k)\| < 1$ we have:

$$\omega(\|\tilde{u}_n(\tilde{w}_k)\|) \leq R_n(\tilde{w}_k) - R_n(w_n^*) \leq \omega_*(\|\tilde{u}_n(\tilde{w}_k)\|). \tag{21}$$

Therefore, if $\|\tilde{u}_n(\tilde{w}_k)\| \leq 0.68$, we have:

$$\begin{aligned} R_n(\tilde{w}_k) - R_n(w_n^*) &\leq \omega_*(\|\tilde{u}_n(\tilde{w}_k)\|) \leq \|\tilde{u}_n(\tilde{w}_k)\|^2 \\ &\overset{(20)}{\leq} \tfrac{1}{(1-\beta)^2}\|\tilde{v}_n(\tilde{w}_k)\|^2 = \tfrac{1}{(1-\beta)^2}\delta_n^2(\tilde{w}_k) \end{aligned} \tag{22}$$

Therefore, we can note that $\delta_n(\tilde{w}_k) \leq (1-\beta)\sqrt{V_n}$ concludes that $R_n(\tilde{w}_k) - R_n(w_n^*) \leq V_n$ when $V_n \leq 0.68^2$.

7.2 PROOF OF THEOREM 1

According to the Theorem 1 in Zhang & Xiao (2015), we can derive the iteration complexity by starting from w_m as a good warm start, to reach w_n which is V_n-suboptimal solution for the risk R_n. By considering assumption 2, we can assume that R_n is a standard self-concordant function. According to the Corollary 1 in Zhang & Xiao (2015), we can note that if we set ϵ_k the same as (12), after K iterations we reach the solution w_n such that $R_n(w_n) - R_n(w_n^*) \leq V_n$ where

$$K = \left\lceil \frac{R_n(w_m) - R_n(w_n^*)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2(\frac{2\omega(1/6)}{V_n}) \right\rceil. \tag{23}$$

Also, according to Lemma 1, we can note that the number of PCG steps needed to reach the approximation of Newton direction with precision ϵ_k is as following:

$$\begin{aligned} T_\mu &= \left\lceil \sqrt{1 + \tfrac{2\mu}{cV_n}} \log_2 \left(\frac{2\sqrt{\frac{cV_n + L}{cV_n}} \|\nabla R_n(\tilde{w}_k)\|}{\epsilon_k} \right) \right\rceil \\ &\overset{(12)}{=} \left\lceil \sqrt{1 + \tfrac{2\mu}{cV_n}} \log_2 \left(\frac{2(cV_n + L)}{\beta c V_n} \right) \right\rceil. \end{aligned} \tag{24}$$

Therefore, we can note that when we start from w_m, which is V_m-suboptimal solution for the risk R_m, after T_n PCG steps, where $T_n \geq KT_\mu$, we reach the point w_n which is V_n-suboptimal solution of the risk R_n, which follows (13).
Suppose the initial sample set contains m_0 samples, and consider the set $\mathcal{P} = \{m_0, \alpha m_0, \alpha^2 m_0, \ldots, N\}$, then after T_N PCG steps, we reach V_N-optimal solution for the whole data set:

$$T_N \geq \sum_{i=2}^{|\mathcal{P}|} \left(\left\lceil \frac{R_{\mathcal{P}[i]}(w_{\mathcal{P}[i-1]}) - R_{\mathcal{P}[i]}(w^*_{\mathcal{P}[i]})}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2(\frac{2\omega(1/6)}{V_{\mathcal{P}[i]}}) \right\rceil \right) \left\lceil \sqrt{1 + \tfrac{2\mu}{cV_{\mathcal{P}[i]}}} \log_2 \left(\frac{2(cV_{\mathcal{P}[i]} + L)}{\beta c V_{\mathcal{P}[i]}} \right) \right\rceil. \tag{25}$$

7.3 PROOF OF COROLLARY 1

The proof of the first part is trivial. According to Lemma 2, we can find the upper bound for $R_n(w_m) - R_n(w_n^*)$, and when $\alpha = 2$, by utilizing the bound (19) we have:

$$\begin{aligned} K &= \left\lceil \frac{R_n(w_m) - R_n(w_n^*)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2(\frac{2\omega(1/6)}{V_n}) \right\rceil \\ &\overset{(19)}{\leq} \underbrace{\left\lceil \frac{\left(3 + \left(1 - \frac{1}{2^\gamma}\right)\left(2 + \frac{c}{2}\|w^*\|^2\right)\right) V_m}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2(\frac{2\omega(1/6)}{V_n}) \right\rceil}_{:=\tilde{K}}. \end{aligned} \tag{26}$$

Therefore, we can note that when we start from w_m, which is V_m-suboptimal solution for the risk R_m, after $\tilde{T}_n$ PCG steps, where $\tilde{T}_n \geq \tilde{K}T_\mu$, T_μ is defined in (24), we reach the point w_n which is V_n-suboptimal solution of the risk R_n, which follows (15).
Suppose the initial sample set contains m_0 samples, and consider the set $\mathcal{P} = \{m_0, 2m_0, 4m_0, \ldots, N\}$, then the total number of PCG steps, $\tilde{T}_N$, to reach V_N-optimal solution

for the whole data set is as following:

$$\sum_{i=2}^{|\mathcal{P}|}\left(\left\lceil\frac{\left(3+\left(1-\frac{1}{2^\gamma}\right)\left(2+\frac{c}{2}\|w^*\|^2\right)\right)V_{\mathcal{P}[i-1]}}{\frac{1}{2}\omega(1/6)}\right\rceil+\left\lceil\log_2(\frac{2\omega(1/6)}{V_{\mathcal{P}[i]}})\right\rceil\right)\left(\left\lceil\sqrt{1+\frac{2\mu}{cV_{\mathcal{P}[i]}}})\right.\right.$$
$$\left.\left.\log_2\left(\frac{2(cV_{\mathcal{P}[i]}+L)}{\beta cV_{\mathcal{P}[i]}}\right)\right\rceil\right)$$
$$\leq\left(\log_2\frac{N}{m_0}+\left(\frac{\left(3+\left(1-\frac{1}{2^\gamma}\right)\left(2+\frac{c}{2}\|w^*\|^2\right)\right)}{\frac{1}{2}\omega(1/6)}\frac{1-(\frac{1}{2^\gamma})^{\log_2\frac{N}{m_0}}}{1-\frac{1}{2^\gamma}}V_{m_0}\right)\right.$$
$$\left.+\sum_{i=2}^{|\mathcal{P}|}\left\lceil\log_2(\frac{2\omega(1/6)}{V_{\mathcal{P}[i]}})\right\rceil\right)\left(\left\lceil\sqrt{1+\frac{2\mu}{cV_N}})\log_2\left(\frac{2}{\beta}+\frac{2L}{\beta c}\cdot\frac{1}{V_N}\right)\right\rceil\right)$$
$$\leq\left(2\log_2\frac{N}{m_0}+\left(\frac{\left(3+\left(1-\frac{1}{2^\gamma}\right)\left(2+\frac{c}{2}\|w^*\|^2\right)\right)}{\frac{1}{2}\omega(1/6)}\frac{1-(\frac{1}{2^\gamma})^{\log_2\frac{N}{m_0}}}{1-\frac{1}{2^\gamma}}V_{m_0}\right)\right.$$
$$\left.+\log_2\frac{N}{m_0}\log_2(\frac{2\omega(1/6)}{V_N})\right)\left(\left\lceil\sqrt{1+\frac{2\mu}{cV_N}})\log_2\left(\frac{2}{\beta}+\frac{2L}{\beta c}\cdot\frac{1}{V_N}\right)\right\rceil\right)\leq\tilde{T}_N. \tag{27}$$