A Distributed Second-Order Algorithm You Can Trust

Celestine Dünner [1] Aurelien Lucchi [2] Matilde Gargiani [3] An Bian [2] Thomas Hofmann [2] Martin Jaggi [4]

Abstract

Due to the rapid growth of data and computational resources, distributed optimization has become an active research area in recent years. While first-order methods seem to dominate the field, second-order methods are nevertheless attractive as they potentially require fewer communication rounds to converge. However, there are significant drawbacks that impede their wide adoption, such as the computation and the communication of a large Hessian matrix. In this paper we present a new algorithm for distributed training of generalized linear models that only requires the computation of diagonal blocks of the Hessian matrix on the individual workers. To deal with this approximate information we propose an adaptive approach that - akin to trust-region methods - dynamically adapts the auxiliary model to compensate for modeling errors. We provide theoretical rates of convergence for a wide class of problems including L_1-regularized objectives. We also demonstrate that our approach achieves state-of-the-art results on multiple large benchmark datasets.

1. Introduction

The last decade has witnessed a growing number of successful machine learning applications in various fields, along with the availability of larger training datasets. However, the speed at which training datasets grow in size is strongly outpacing the evolution of the computational power of single devices, as well as their memory capacity. Therefore, distributed approaches for training machine learning models have become tremendously important while also being increasingly more accessible to users with the rise of cloud-computing. Scaling up optimization algorithms for training machine learning models in such a setting poses many challenges. One key aspect is communication efficiency; because communication is often more expensive than local computation, the overall speed of distributed algorithms strongly depends on how frequently information is exchanged between workers. In order to develop communication-efficient distributed algorithms, we advocate the use of second-order methods which benefit from faster rates of convergence compared to their first-order (gradient-based) counterparts, and hence require less communication rounds to achieve the same accuracy. However, second-order methods have the significant drawback of requiring the computation and storage – and potentially the communication – of a Hessian matrix. Exact methods are therefore elusive for large datasets and one has to resort to approximate methods. In this paper, we propose a method where every worker uses local Hessian information only (i.e., with respect to the local parameters on that worker), hence it does not require any second-order information to be communicated. Conceptually, this approach relies on approximating the full Hessian matrix with a block-diagonal version. At the same time, to automatically adapt to the model misfit, we use an adaptive approach similar in spirit to trust-region methods (Conn et al., 2000).

Problem Setup & Distributed Setting. We address the problem of training generalized linear models which are ubiquitous in machine learning, including e.g. logistic regression, support vector machines as well as sparse linear models such as lasso and elastic net. Formally, we address convex optimization problems with an objective of the form

$$F(\boldsymbol{\alpha}) := f(A\boldsymbol{\alpha}) + \sum\nolimits_i g_i(\alpha_i), \tag{1}$$

where we assume f to be smooth and convex, and g_i to be convex functions. $A \in \mathbb{R}^{d\times n}$ is a given data matrix and $\boldsymbol{\alpha} \in \mathbb{R}^n$ the parameter vector to be learned from data.

We assume that every worker $k \in \{1 \dots K\}$ only has access to its own local part of the data, which corresponds to a subset of the columns of the matrix A. In machine learning, these columns typically correspond to a subset of the features or data examples, depending on the application. For example, in the case where (1) corresponds to the objective of a regularized generalized linear model – i.e., where f is a data dependent loss and $g = \sum_i g_i$ a regularization term – the columns of A correspond to features. In another

[1] IBM Research – Zürich, Switzerland [2] ETH, Zürich, Switzerland [3] Albert-Ludwigs-Universität, Freiburg, Germany [4] EPFL, Lausanne, Switzerland. Correspondence to: Celestine Dünner <cdu@zurich.ibm.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

scenario where (1) corresponds to the dual representation of the respective problem, such as typically chosen for SVM models, the columns of A correspond to data examples.

Block-separable model. In such a distributed setting we suggest optimizing a block-separable auxiliary model which can be split over workers. This auxiliary model is then updated in each round, upon receiving a summary of the updates from all workers. A significant advantage of such a model is that the workload of a single round can be parallelized across the individual workers, where each worker computes an update for its own model parameters by solving a local optimizaition task. Then, to synchronize the work, each worker communicates this update to the master node which aggregates all the updates, applies them to the global model and shares this information with all the workers. One common problem faced with this type of distributed approach is to evaluate whether the local models can be trusted in order to update the global model. This is usually addressed by the selection of an appropriate step-size or by relying on a line-search approach. However, the latter uses a fixed model and typically requires multiple model evaluations which can therefore be computationally expensive. In this paper, we instead leverage ideas from trust-region methods (Conn et al., 2000), where we dynamically adapt the model based on how much we trust the approximate second-order information.

Contributions. We propose a new distributed Newton's method, built on an adaptive block-separable approximation of the objective function, and allowing the use of arbitrary solvers to solve the local subproblems approximately. Two characteristics differentiate our approach from existing work. First, unlike previous methods that rely on fixed step-size schedules or line-search strategies, our algorithm evaluates the fit of the auxiliary model using a trust-region approach. This yields an efficient method with global convergence guarantees for convex functions, while providing full adaptivity to the quality of the second-order model. Second, our method, to the best of our knowledge, is the first to give convergence guarantees for a distributed second-order method applied to problems with general regularizers (not necessarily strongly convex). This includes L_1-regularized objectives such as Lasso and sparse logistic regression as very important application cases, which were not covered by earlier methods such as (Shamir et al., 2014; Zhang & Lin, 2015; Wang et al., 2017; Lee & Chang, 2017).

2. Method Description

We present an iterative descent algorithm that minimizes the objective $F(\boldsymbol{\alpha})$ introduced in (1). At each step, we optimize an auxiliary *block-separable model* that acts as a surrogate for the objective $F(\boldsymbol{\alpha})$. This auxiliary model is adaptive and changes depending on its approximation quality.

2.1. Block-Separable Model

Let us, in every iteration of our algorithm, consider the following auxiliary model replacing (1):

$$\mathcal{M}_\sigma(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha}) := \hat{f}(A\boldsymbol{\alpha}, A\Delta\boldsymbol{\alpha}) + \sum_i g_i(\alpha_i + \Delta\alpha_i), \quad (2)$$

where $\hat{f}(A\boldsymbol{\alpha}, A\Delta\boldsymbol{\alpha})$ is a second-order approximation of the data-dependent term in (1), i.e.,

$$\begin{aligned}\hat{f}(A\boldsymbol{\alpha}, A\Delta\boldsymbol{\alpha}) := f(A\boldsymbol{\alpha}) + \nabla f(A\boldsymbol{\alpha})^\top A\Delta\boldsymbol{\alpha} \\ + \frac{\sigma}{2}\Delta\boldsymbol{\alpha}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}. \quad (3)\end{aligned}$$

The parameter $\sigma \in \mathbb{R}_+$ is introduced to control the approximation quality of the auxiliary model; its role will be detailed in Section 2.2.

Let us consider (3) for the case where $\tilde{H}(\boldsymbol{\alpha})$ is chosen to be the Hessian matrix $\nabla^2_{\boldsymbol{\alpha}} f(A\boldsymbol{\alpha})$. Then, the auxiliary model (2) with $\sigma = 1$ corresponds to a classical second-order approximation of the function f. However, this choice of $\tilde{H}$ is not feasible in a distributed setting where the data is partitioned among the workers, since the computation of the Hessian matrix requires access to the entire data matrix.

Partitioning. In particular, we assume each worker has access to a subset $\mathcal{I}_k$ of the columns of A. In our setting, $\mathcal{I}_k$ are disjoint index sets such that $\bigcup_k \mathcal{I}_k = [n], \;\; \mathcal{I}_i \cap \mathcal{I}_j = \emptyset \;\; \forall i \neq j$ and $n_k := |\mathcal{I}_k|$ denotes the size of partition k. Hence, each machine stores in its memory the submatrix $A_{[k]} \in \mathbb{R}^{d \times n_k}$ corresponding to its partition $\mathcal{I}_k$.

Given such a partitioning, we suggest choosing $\tilde{H}$ to be a block diagonal approximation to the Hessian matrix $\nabla^2_{\boldsymbol{\alpha}} f(A\boldsymbol{\alpha})$ aligned with the partitioning of the model parameters, such that

$$\Delta\boldsymbol{\alpha}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha} = \textstyle\sum_k \Delta\boldsymbol{\alpha}_{[k]}{}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}_{[k]}. \quad (4)$$

We use the notation $\mathbf{u}_{[k]}$ to denote the vector $\mathbf{u}$ with only non-zero coordinates for $i \in \mathcal{I}_k$. As a consequence of (4) the model presented in (2) splits over the K partitions, i.e.,

$$\mathcal{M}_\sigma(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha}) = \textstyle\sum_k \mathcal{M}_\sigma^{(k)}(\Delta\boldsymbol{\alpha}_{[k]};\boldsymbol{\alpha}), \quad (5)$$

where each subproblem $\mathcal{M}_\sigma^{(k)}(\Delta\boldsymbol{\alpha}_{[k]};\boldsymbol{\alpha})$ only requires access to the local data indexed by $\mathcal{I}_k$, the respective coordinates of the model $\boldsymbol{\alpha}$, as well as $\mathbf{v} := A\boldsymbol{\alpha}$:

$$\begin{aligned}\mathcal{M}_\sigma^{(k)}(\Delta\boldsymbol{\alpha}_{[k]};\boldsymbol{\alpha}) :=& \frac{1}{K} f(\mathbf{v}) + \nabla f(\mathbf{v})^\top A\Delta\boldsymbol{\alpha}_{[k]} \\ &+ \frac{\sigma}{2}\Delta\boldsymbol{\alpha}_{[k]}{}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}_{[k]} \\ &\textstyle\sum_{i\in\mathcal{I}_k} g_i((\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}_{[k]})_i). \quad (6)\end{aligned}$$

Hence, in a distributed setting, each worker is assigned the subproblem corresponding to its partition. These individual subproblems can be optimized independently and in parallel on the different workers. We note that this requires access to the shared information $\mathbf{v}$ on every node; we will detail in Section 3 how this can be efficiently achieved in a distributed setting. A significant benefit of this model is that it is based on local second-order information and does not require sending gradients and Hessian matrices to the master node, which would be a significant cost in terms of communication.

2.2. Approximation Quality of the Model

The role of the σ parameter introduced in (2) is to account for the loss of information that arises by enforcing the approximate Hessian matrix of f to have a block diagonal structure. The better the approximation, the closer to 1 the optimal σ parameter is. If the Hessian approximation is unreliable, then the model should be adapted accordingly by changing the value of σ. An alternative model to (2) would be to include a damping factor to the second-order term, i.e., use $\frac{\sigma}{2}\Delta\boldsymbol{\alpha}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha} + \sigma'\|\Delta\boldsymbol{\alpha}\|^2$ where $\sigma' > 0$. This type of model is usually employed in trust-region methods (Conn et al., 2000) where $\sigma = 1$, and $\sigma' > 0$ is chosen to ensure strong-convexity. The use of $\sigma' > 0$ might therefore not be necessary for models that are already (strongly)-convex. We conducted a set of experiments to determine whether this alternative model would achieve better empirical performance and we found little difference between the two models. We will therefore report results for our suggested model with $\sigma' = 0$ in the experimental section.

Adaptive Choice of σ. We have established that the σ parameter has a central role for the convergence and the practical performance of our method, and we therefore need an efficient way to choose and update this parameter in an adaptive manner. Here we suggest updating σ at each iteration of the algorithm using an update rule inspired by trust-region methods (Cartis et al., 2011a), where σ acts as the reciprocal of the trust-region radius. Further details are provided in Section 2.3.

2.3. Algorithm Procedure

The pseudo-code of the proposed approach, denoted as Adaptive Distributed Newton method (ADN), is summarized in Algorithm 1 and the four-stage iterative procedure is illustrated in Figure 1. We focus on a master-worker setting in this paper, but our algorithm could similarly be applied in a non-centralized fashion. Specifically, in every round, each worker k works on its local subproblem (6) to find an update $\Delta\boldsymbol{\alpha}_{[k]}$ to its local parameters of the model (stage 1). Then, it communicates this update to the master node (stage 2) which, aggregates the updates, and decides a new σ_{t+1} for the next iteration based on the misfit of the current model (stage 3). Finally, the master node broadcasts the new model together with σ_{t+1} to every worker (stage 4) for the next round. Note that in Algorithm 1 we have not explicitly stated the communication of $\mathbf{v}$ and two scalars that are necessary for evaluating the function values distributedly; we will elaborate more on this in Section 3.

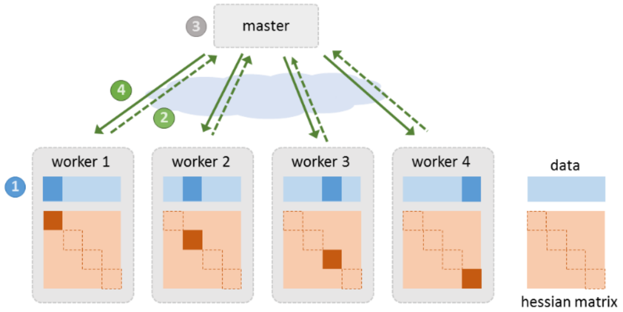


Figure 1. Four-stage algorithmic procedure of ADN. Every worker only has access to its local partition of the data matrix and the respective block of the Hessian matrix. Arrows indicate the (synchronous) communication per round.

Local Solver. The computation of the model update $\Delta\boldsymbol{\alpha}_{[k]}$ on every worker (stage 2 of our algorithm) can be done using any arbitrary solver, depending on user preference or the available hardware resources. As in (Smith et al., 2018), the amount of computation time spent in the local solver is a tunable hyperparameter. This allows the algorithm to be optimally adjusted according to the trade-off between communication and computation cost of a given system. To reflect this flexibility in our theory we will assume the local subproblems (6) are not necessarily optimized exactly but η-approximately, i.e., the local updates $\Delta\boldsymbol{\alpha}_{[k]}$ are such that:

$$\frac{\mathcal{M}_\sigma^{(k)}(\Delta\boldsymbol{\alpha}_{[k]};\boldsymbol{\alpha}) - \mathcal{M}_\sigma^{(k)}(\Delta\boldsymbol{\alpha}_{[k]}^\star;\boldsymbol{\alpha})}{\mathcal{M}_\sigma^{(k)}(\mathbf{0};\boldsymbol{\alpha}) - \mathcal{M}_\sigma^{(k)}(\Delta\boldsymbol{\alpha}_{[k]}^\star;\boldsymbol{\alpha})} \le \eta\,, \tag{7}$$

where $\Delta\boldsymbol{\alpha}_{[k]}^\star := \arg\min_{\Delta\boldsymbol{\alpha}_{[k]}} \mathcal{M}_\sigma^{(k)}(\Delta\boldsymbol{\alpha}_{[k]};\boldsymbol{\alpha})$.[1]

As previously mentioned, one of the key steps in the adaptive approach presented in Algorithm 1 is the strategy for adapting the model over iterations. This is done by adjusting σ in every iteration t resulting in a schedule described by the sequence $\{\sigma_t\}_{t\ge 0}$. In particular, after every iteration, we adjust σ_t based on the agreement between the model function (2) and the objective (1) for the current iterate. This is measured by the variable ρ_t defined in (8) in Algorithm 1. If ρ_t is close to 1 there is a good agreement between the

[1] Note that the notion $\eta \in [0, 1)$ of multiplicative subproblem accuracy is sensible even in the case when the block-wise Hessian matrix $\tilde{H}$ is not necessarily positive definite as long as g is strongly convex or of bounded support.

model $\mathcal{M}_{\sigma_t}(\cdot)$ and the function $F(\cdot)$ and we retain our current model. On the other hand, if the model over-estimates the objective, we decrease σ for the next iteration, which can be thought of as adjusting the trust in the current approximation of the Hessian. On the contrary, if our model under-estimates the objective we increase σ. In addition, we only apply updates $\Delta\boldsymbol{\alpha}$ that satisfy $\rho_t \geq \xi$ and hence provide sufficient function decrease. If this is not fulfilled, the step is rejected and a new update is computed in the next iteration, based on the adjusted model. In order to adequately deal with all these cases that influence σ, we introduce two constants ζ and γ that control how to update σ based on the value of ρ_t (see (10) in Algorithm 1). We will discuss the choice of these constants in the experiment section.

Algorithm 1 Adaptive Distributed Newton Method (ADN)

```
1:  Input: $\boldsymbol{\alpha}_0 \in \mathbb{R}^n$ (e.g., $\boldsymbol{\alpha}_0 = \mathbf{0}$) and $\sigma_0 > 0$.
        $\gamma, \zeta > 1$, $\frac{1}{\zeta} > \xi > 0$ and $\eta \in [0, 1)$
2:  for $t = 0, 1, \ldots,$ until convergence do
3:      for $k \in [K]$ in parallel do
4:          Obtain $\Delta\boldsymbol{\alpha}_{[k]}$ by minimizing $\mathcal{M}^{(k)}_{\sigma_t}(\Delta\boldsymbol{\alpha}_{[k]}; \boldsymbol{\alpha}^{(t)})$
            $\eta$-approximately
5:      end for
6:      Aggregate updates $\Delta\boldsymbol{\alpha} = \sum_k \Delta\boldsymbol{\alpha}_{[k]}$
7:      Compute $F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha})$ (distributed over workers)
8:      Compute $\mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)})$ (distributed over workers)
9:      Evaluate
            $\rho_t := \frac{F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha})}{F(\boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)})}$    (8)
10:     Set
            $\boldsymbol{\alpha}^{(t+1)} := \begin{cases} \boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha} & \text{if } \rho_t \geq \xi \\ \boldsymbol{\alpha}^{(t)} & \text{otherwise} \end{cases}$    (9)
11:     Set
            $\sigma_{t+1} := \begin{cases} \frac{1}{\gamma}\sigma_t & \text{if } \rho_t > \zeta \text{ (too conservative)} \\ \sigma_t & \text{if } \zeta \geq \rho_t \geq \frac{1}{\zeta} \text{ (good fit)} \\ \gamma\sigma_t & \text{if } \frac{1}{\zeta} > \rho_t \text{ (too aggressive)} \end{cases}$    (10)
12: end for
```

3. Implementation

In order to implement Algorithm 1 efficiently in a distributed environment, two key aspects need to be considered.

3.1. Shared Information

We have seen in Section 2.1 that every worker needs access to $\mathbf{v} := A\boldsymbol{\alpha}$ in order to evaluate the gradient $\nabla f(A\boldsymbol{\alpha})$ for solving the local subproblem. To avoid the evaluation of $\mathbf{v}$ in every round we suggest sharing and updating the vector $\mathbf{v} = A\boldsymbol{\alpha}$ throughout the algorithm – thus, the term shared vector. Hence, if the model parameters are updated locally, the respective change $\Delta\mathbf{v}_{[k]} = A\Delta\boldsymbol{\alpha}_{[k]}$ is shared between workers, whereas the local model parameters $\boldsymbol{\alpha}_{[k]}$ are kept local on every worker. A similar approach to achieve communication-efficiency is suggested in (Smith et al., 2018). They also emphasize that the vector to be communicated is d-dimensional which can be preferable compared to the n-dimensional model vector $\boldsymbol{\alpha}$, depending on the dimensionality of the problem. This shared vector modification is a minor change of step 6 in Algorithm 1, where $\Delta\mathbf{v} = \sum_k \Delta\mathbf{v}_{[k]}$ is aggregated and shared instead of $\Delta\boldsymbol{\alpha}$.

3.2. Communication-Efficient Function Evaluation

Let us detail how ρ_t in Step 9 of Algorithm 1 can be evaluated efficiently without central access to the model $\boldsymbol{\alpha}$. We therefore consider the individual terms in (8) separately: The cost $F(\boldsymbol{\alpha})$ is known from the previous iteration and can be stored in memory. The cost at the new iterate $F(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) = f(A(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha})) + \sum_i g_i((\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha})_i)$ is composed of two terms, where the first term can be computed on the master locally as $f(\mathbf{v} + \sum_k \Delta\mathbf{v}_k)$ and the second term needs to be computed in a distributed fashion. Every node computes $g_{(k)} := \sum_{i \in \mathcal{I}_k} g_i((\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}_{[k]})_i)$ based on its local model parameters and sends the resulting value to the master node, which adds the overall sum to the first term, completing the evaluation of the new objective value. Similarly, the model cost $\mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha})$ is computed distributedly by every node independently evaluating $\mathcal{M}^{(k)}_{\sigma_t}(\Delta\boldsymbol{\alpha}_{[k]}; \boldsymbol{\alpha}_{[k]})$ and then sharing the result. Note that this step can be computationally expensive, since it requires one pass through the local data on every node; the communication cost of the two scalar values is negligible.

4. Convergence Analysis

We now establish the convergence of Algorithm 1 for the general class of functions fitting (1).

Theorem 1 (non-strongly convex g_i)**.** *Let f be $\frac{1}{\tau}$-smooth and g_i be convex functions. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded by σ_{sup}.*[2] *Then, Algorithm 1 reaches a suboptimality $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \varepsilon$ within a total number of*

$$\frac{1}{\log(\gamma)} \log\left(\frac{\sigma_{sup}}{\sigma_0}\right) + \frac{2}{\varepsilon} C_1 \sigma_{sup}$$

iterations, where $C_1 > 0$ is a constant defined as $C_1 := \frac{2(4L^2R^2 + \tau\varepsilon_0)}{\tau\xi(1-\eta)}$ where $L, R > 0$ are such that $|\alpha_i^{(t)}| < L \;\; \forall i, t \geq 0$ and $\|A_{:,i}\| < R \; \forall i$, and $\varepsilon_0 := F(\boldsymbol{\alpha}_0) - F(\boldsymbol{\alpha}^\star)$ is the initial suboptimality.

For the special case where g_i are strongly-convex, Algo-

[2] We will theoretically establish the upper bound σ_{sup} for two general scenarios in Appendix A.7.

rithm 1 achieves a faster rate of convergence as described in the following theorem.

Theorem 2 (strongly-convex g_i)**.** *Let f be $\frac{1}{\tau}$-smooth and g_i μ-strongly convex. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded by σ_{sup}. Then, Algorithm 1 reaches a suboptimality $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \varepsilon$ within a total number of*

$$\frac{1}{\log(\gamma)} \log\left(\gamma \frac{\sigma_{sup}}{\sigma_0}\right) + \frac{2}{\log(C_2^{-1})} \log\left(\frac{\varepsilon_0}{\varepsilon}\right)$$

iterations, where $C_2 \in (0,1)$ is a constant defined as $C_2 := 1 - \xi(1-\eta)\frac{\mu\tau}{c_A \sigma_{\text{sup}} + \mu\tau}$ with $c_A = \max_k \|A_{[k]}\|^2$ and $\varepsilon_0 = F(\boldsymbol{\alpha}_0) - F(\boldsymbol{\alpha}^\star)$ measures the initial suboptimality.

Note that for strongly-convex functions g_i, similar global rates of convergence to the one derived in Theorem 2 are obtained by existing distributed second-order methods such as (Lee & Chang, 2017; Wang et al., 2017). However, we are not aware of any result similar to Theorem 1 in the more general case where g_i are non-strongly convex functions.

Proof Sketch

We summarize the main steps in the proof of Theorem 1 and 2, a detailed derivation is provided in the Appendix.

Step 1. Recall that the model with block diagonal Hessian approximation, described in Section 2.1, acts as a surrogate to minimize the function introduced in (1). The first step is therefore to establish a bound on the decrease of the auxiliary model for every step of the algorithm, given that each local subproblem is solved η-approximately. This bound on the model decrease $\mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha})$, stated in Lemma 3, is established using a primal-dual perspective on the problem, similar to (Shalev-Shwartz & Zhang, 2013).

Lemma 3. *Assume f is $\frac{1}{\tau}$-smooth and g_i are μ-strongly convex with $\mu \geq 0$. Then, the per-step model decrease of Algorithm 1 can be lower bounded as:*

$$\begin{aligned} &\mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha}^{(t)}) \\ &\geq (1-\eta)\left[\kappa\mathcal{G}(\boldsymbol{\alpha}^{(t)}) - \frac{\kappa^2}{2}R^{(t)}\right], \end{aligned}$$

where $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ denotes the duality gap, $\kappa \in (0,1]$ and

$$\begin{aligned} R^{(t)} :=& \sigma_t(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}) \\ &- \tfrac{\mu(1-\kappa)}{\kappa}\|\boldsymbol{\alpha}^{(t)} - \mathbf{u}^{(t)}\|_2^2 \end{aligned}$$

with $u_i^{(t)} \in \partial g_i^*(A_{:,i}^\top \nabla f(A\boldsymbol{\alpha}^{(t)}))$[3].

Step 2. For iterations that are successful (i.e., they provide sufficient function decrease as measured by $\rho_t \geq \xi$ in step 10 of Algorithm 1), the construction of Algorithm 1 allows us to relate the model decrease from Lemma 3 to the function decrease $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha})$ through the parameter ξ. This yields a lower bound on the function decrease for every successful update as provided in Lemma 4 below.

Lemma 4. *The function decrease of Algorithm 1 for a successful update $(\Delta\boldsymbol{\alpha}, \sigma_t)$ can be bounded as:*

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \geq \xi(1-\eta)\left[\kappa\mathcal{G}(\boldsymbol{\alpha}^{(t)}) - \frac{\kappa^2}{2}R^{(t)}\right],$$

where $\kappa \in (0,1]$ and $R^{(t)}$ is defined as in Lemma 3.

Step 3. At this stage, we have shown that each successful iteration decreases the function value, therefore making progress towards the optimum. However, unsuccessful iterations (for which $\rho_t < \xi$) do not decrease the objective and overall convergence to an optimum can only occur if the number of these iterations is limited. The next step is therefore to bound the number of unsuccessful iterations. This is accomplished by showing that the construction of the sequence $\{\sigma_t\}_{t\geq 0}$ is such that the number of successive unsuccessful iterations is bounded and, hence, increasing σ will eventually yield a successful iteration that will allow us to decrease the objective function. This results in a bound on the number of successful and unsuccessful iterations derived in the Appendix. Finally, the rate of convergence in Theorem 1 and Theorem 2 are obtained by combining the bound on the number of steps with the function decrease for each successful step.

Remark. Note that the update scheme (10) in Algorithm 1 is one of many that satisfy the conditions required for proving convergence. For further details, we refer the reader to the literature on trust-region methods (Conn et al., 2000).

5. Related Work

First-order Methods. Most first-order stochastic methods require frequent communication which comes with high costs in distributed settings, thus they are often prefered in multi-core settings. This is for example the case for the popular Hogwild! algorithm (Niu et al., 2011) that relies on asynchronous SGD updates in a lock-free setting and requires communication after each optimization step. Alternatives include variance-reduced methods such as (Lee et al., 2015) and coordinate descent methods such as (Richtárik & Takáč, 2016), however, they suffer similar communication bottlenecks.

Trust-region Methods. These methods use a surrogate model to approximate the objective within a region around the current iterate. The size of the trust region is expanded or contracted according to the fitness of the surrogate model to the true objective. For efficiency reasons, the surrogate

[3] g_i^* denotes the convex conjugate of the function g_i, which is defined as $g_i^*(u) := \sup_v[uv - g_i(v)]$.

model is often a quadratic model (Conn et al., 2000; Karimireddy et al., 2018), although cubic models can also be used (Nesterov & Polyak, 2006). Though trust-region methods have been extensively used in a single-machine setting, to the best of our knowledge we are the first to apply a trust-region-like approach in a *distributed* setting.

Line-search vs Trust-region. Line-search techniques are a popular way to guarantee convergence and they have recently been explored in distributed settings, e.g., (Hsieh et al., 2016; Lee & Chang, 2017; Trofimov & Genkin, 2017; Mahajan et al., 2017; Lee et al., 2018). Our trust-region approach has clear advantages compared to line-search methods: i) a line-search method assumes a *fixed* auxiliary model –which may be an arbitrarily bad approximation of the true objective– that is used to find an acceptable step size. In contrast, our approach adaptively tunes the auxiliary model to ensure that it is a good fit to the true objective. ii) in general, a line-search method requires multiple objective value evaluations in order to test different step sizes, while our approach only needs one objective value evaluation to calculate ρ_t. The advantages of our method are verified empirically in Section 6.

Approximate Newton-type Methods. For distributed L_1-regularized problems (Andrew & Gao, 2007) proposed a quasi-newton method without convergence guarantees. Most of the literature on Newton-type methods are otherwise designed to optimize strongly-convex objectives. DANE (Shamir et al., 2014) is a distributed approximate Newton-method with a linear rate of convergence for quadratic functions. AIDE (Reddi et al., 2016) is an accelerated version using the Catalyst scheme. Another similar approach is DiSCO (Zhang & Lin, 2015) which consists of an inexact damped Newton method using conjugate gradient steps, achieving a linear rate of convergence for self-concordant functions. Finally, GIANT (Wang et al., 2017) relies on conjugate gradient steps and achieves a local linear-quadratic convergence rate but does not provide a global rate of convergence. It was shown empirically to outperform DANE, AIDE and DiSCO. Note that the convergence results of these approaches require each subproblem to be solved with high accuracy, which is often prohibitive for large-scale datasets. Some approaches suggest using a block-diagonal Hessian approximation such as (Hsieh et al., 2016; Lee & Chang, 2017; Lee & Wright, 2018) but they all rely on a line-search approach which is shown to be inferior to our adaptive approach in the experimental section. While both our approach and (Lee & Chang, 2017) require $\mathcal{O}(\log(1/\varepsilon))$ iterations to reach ε accuracy for a strongly-convex g, we further provide a rate of convergence for the more general case where g is non-strongly convex.

Distributed Primal-Dual Methods. Approaches such as (Yang, 2013; Jaggi et al., 2014; Zhang & Lin, 2015; Zheng et al., 2017; Wang et al., 2017) are restricted to strongly-convex regularizers, and typically work on the dual formulation of the objective. CoCoA (Smith et al., 2018) provides an extension to a wider class of regularizers, including L_1, as of interest here. Although it allows for the use of arbitrary solvers on each worker to regulate the amount of communication, this approach is inherently based on a first-order model of the objective and does not use second-order information.

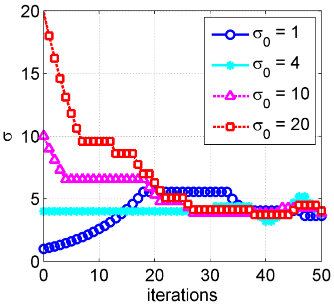


Figure 2. Robustness to initialization: Training the dual logistic regression model on a subsample (1 million examples) of the criteo dataset for different σ_0 with $\gamma = 1.1$, $\zeta = 1.1$ and $\xi = 0$.

In an earlier work by (Gargiani, 2017) a modification of CoCoA was discussed which incorporates local second-order information for the general class of problems (1). We here extend this approach to be adaptive to the quality of the local surrogate model in a trust region sense, in contrast to using fixed Hessian information (Hsieh et al., 2016; Gargiani, 2017; Lee & Chang, 2017; Lee & Wright, 2018).

6. Experimental Results

We devote the first part of this section to analysing the properties of our adaptive scheme. In the second part we evaluate its performance for training a logistic regression model regularized with L_1 and L_2 regularization. We compare ADN to state-of-the-art distributed solvers on four large-scale datasets (see Table 1). All algorithms presented in this section are implemented in C++, they are optimized for sparse data structures and use MPI to handle communication between workers. If not stated otherwise, we use $K = 8$ workers.

	# examples	# features	sparsity
url	2’396’130	3’230’442	3.58 E-05
webspam	262’938	680’715	2.24 E-04
kdda	8’407’751	19’306’083	1.80 E-06
criteo	45’840’617	1’000’000	1.95 E-06

Table 1. Datasets used for the experiments.

6.1. Algorithm Properties

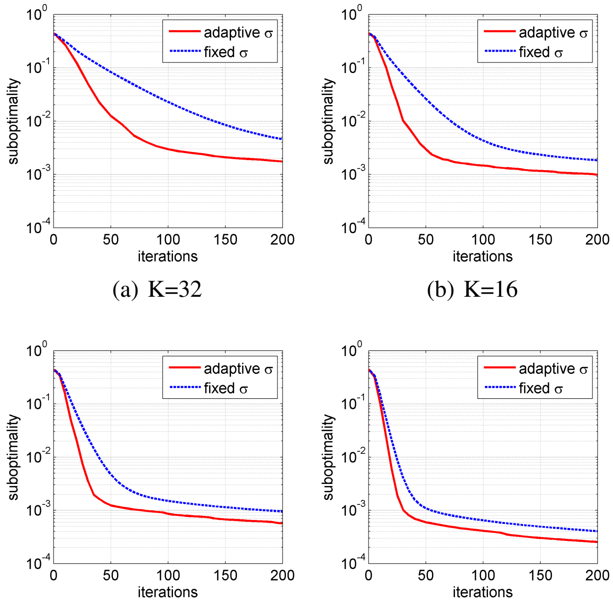
Initialization of σ. Given the wide dissemination of machine learning models to diverse fields, it is becoming increasingly important to develop algorithms that can be deployed without requiring expert knowledge to choose parameters. In this context we first check the sensitivity of our algorithm to the choice of σ_0. The results shown in Figure 2 demonstrate that our adaptive scheme dynamically finds an appropriate value of σ_t, independently of the initialization.

Parameter-Free Update Strategy. In addition to σ_0 there are three more parameters in Algorithm 1 – namely ζ, γ and ξ – that determine how to update σ_t. The most natural choice for ξ is a small positive value, as we do not want to discard updates that would yield a function decrease; we therefore choose $\xi = 0$. The convergence of Algorithm 1 is guaranteed for any choice of $\zeta, \gamma > 1$, and we found empirically that the performance is not very sensitive to the choice of these parameters and the optimal values are robust across different datasets (e.g., $\gamma = \zeta \approx 1.2$ is generally a good choice). However, to completely eliminate these parameters from the algorithm we suggest the following practical parameter-free update schedule:

$$\sigma_{t+1} := \frac{f(A(\boldsymbol{\alpha}^{(t)}+\Delta\boldsymbol{\alpha})) - f(A\boldsymbol{\alpha}^{(t)}) - \nabla f(A\boldsymbol{\alpha}^{(t)})A\Delta\boldsymbol{\alpha}}{\hat{f}(A\boldsymbol{\alpha}^{(t)}, A\Delta\boldsymbol{\alpha}) - f(A\boldsymbol{\alpha}^{(t)}) - \nabla f(A\boldsymbol{\alpha}^{(t)})A\Delta\boldsymbol{\alpha}}\sigma_t.$$

This scheme is not only parameter-free, but it also adapts σ proportionally to the misfit of the model. The evaluation of this scaling factor does not add any additional computation to the evaluation of ρ_t. Note that for this scheme to meet the required conditions of convergence presented in Section 4, we need to ensure that the sequence of σ_t is bounded, which can easily be done by defining an arbitrary maximum value although we empirically found that this was not necessary. Because of this appealing property of not requiring any tuning we will use this strategy for the following experiments.

Gain of Adaptive Strategy. In this section we investigate the benefits of using an adaptive σ as opposed to a static one. We focus on a dual L_2-regularized logistic regression model where f is a quadratic function and thus, its Hessian corresponds to a scaled identity matrix. This allows us to study the effect of adaptivity in isolation. It also allows us to compare to a reference model with $\sigma = K$ which comes with convergence guarantees, see (Smith et al., 2018). In Figure 3 we compare the two approaches and observe that with an increasing number of workers, the gains provided by the adaptive approach increase. This comes from the fact that the more workers we have, the less accurate the block diagonal approximation in the auxiliary model is and thus it is increasingly difficult to establish a safe fixed value for σ that covers any partitioning of the data in an ad hoc fashion. Note that the adaptive strategy does not only improve over the safe fixed value of σ as shown in Figure 3 but it also enables convergence for objectives to be guaranteed where no tight practical bound is known.



(a) K=32

(b) K=16

(c) K=8

(d) K=4

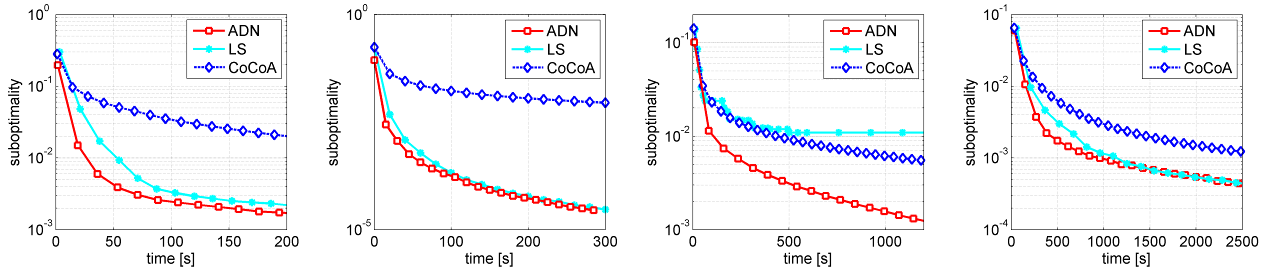
Figure 3. Comparison of using an adaptive approach for σ vs. using a fixed safe value for σ for different numbers of workers (K). Training L_2 logistic regression on a subsample (10 million examples) of the criteo dataset.

6.2. Performance for Logistic Regression

We now analyse the performance of ADN for training a Logistic Regression model on multiple large-scale datasets and compare it to different state-of-the-art methods. First, we will consider L_2 regularization, which results in a strongly-convex objective function. This enables the application of a broad range of existing methods. In the second part of this section we focus on L_1 regularization, where – to the best of our knowledge – the only existing baselines that come with convergence guarantees are CoCoA (Smith et al., 2018) and slower mini-batch proximal SGD.

Baselines. We compare our approach against *GIANT* as a representative scheme for the class of approximate Newton methods. This approach was shown in (Wang et al., 2017) to achieve competitive performance to other similar algorithms such as DANE or DiSCO. The main difference between these methods and ours is that they build updates based on a local approximation of the full Hessian matrix, whereas we work with exact blocks of the full Hessian matrix. In order to establish a fair comparison, we re-implemented GIANT using MPI while following the open source implementation provided by the authors[4]. We use conjugate gradient

[4] https://github.com/wangshusen/SparkGiant



(a) url (b) webspam (c) kdda (d) criteo

Figure 4. Performance comparison of primal solver for L_1-regularized Logistic Regression.

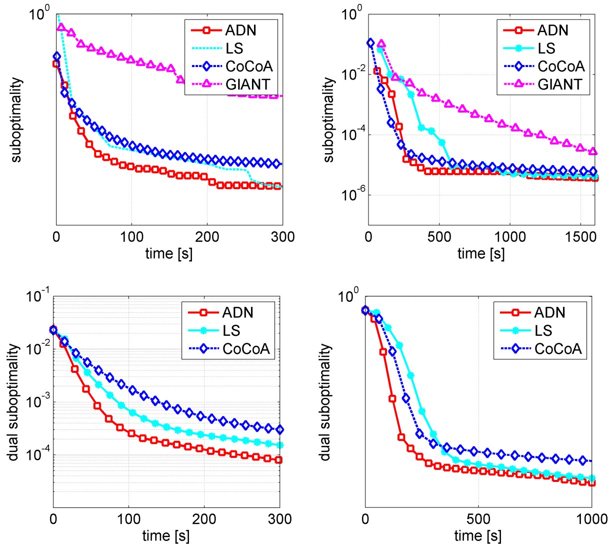


Figure 5. Performance comparison for L_2-regularized logistic regression on url dataset (left) and criteo dataset (right) for solving the primal problem (top) and the dual problem (bottom).

descent as a local solver and implemented the suggested backtracking line-search approach.

Our second baseline is the approach presented in (Lee & Chang, 2017) which is similar to ours as it builds on the same block diagonal approximation of the Hessian matrix. However, it uses a fixed model and then relies on a backtracking line search approach to guarantee convergence. We will refer to this scheme as *LS* in our experiments.

The third baseline is *CoCoA* which approximates the Hessian $\nabla^2 f(.)$ by a scaled identity matrix using the smoothness property of f. Their quadratic model performs well if f is indeed a quadratic function such as the least squares loss or the dual of the L_2 regularizer. However, we will see that this is not a good model for the logistic loss function.

L_1 **Regularization.** We consider the L_1-regularized logistic regression problem on the datasets introduced in Table 1. We compare CoCoA (applied to the L1 primal problem) and LS to ADN in Figure 4. In general, we see significant gains from ADN over CoCoA which can be attributed to CoCoA using a quadratic approximation to the logistic function which is not a good fit. The performance of LS is similar or slightly worse than our approach, depending on the dataset. However, as shown in Figure 4(c), it can be unstable since the line-search approach used in (Lee & Chang, 2017) does not come with any theoretical guarantees for functions that are not strongly-convex.

L_2 **Regularization.** For L_2-regularized logistic regression, CoCoA, ADN and LS use a dual solver. The results presented in Figure 5 show that CoCoA is competitive in this case since it uses the same block diagonal approximation of the Hessian matrix and benefits from cheap iterations as no function evaluations are needed. However, we can see that using an adaptive strategy nevertheless pays off and we can achieve a gain over CoCoA. For very high accuracy solutions (<10^{-6}), a solver that uses the full Hessian should be preferred if possible.

7. Conclusion

We have presented a novel distributed second-order algorithm that optimizes an auxiliary model with a block-diagonal Hessian matrix. The separable structure of this model makes its optimization easily parallelizable. Each worker optimizes its own local model and sends a minimal amount of information to the master node. Our framework therefore avoids the computation and communication of an expensive Hessian matrix. In order to adjust for the approximation error of the model, we proposed using an adaptive scheme that resembles trust-region methods. This allows us to derive global guarantees of convergence for convex functions. Specializing our approach to strongly-convex functions recovers convergence results derived by existing distributed second-order methods. From the practical side, we have proposed a parameter-free version of our algorithm, discussed how to develop an efficient implementation and demonstrated significant speed-ups over state-of-the-art baselines on several large-scale datasets.

References

Andrew, G. and Gao, J. Scalable training of L1-regularized log-linear models. In *International Conference on Machine Learning*, 2007.

Bach, F. et al. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4:384–414, 2010.

Cartis, C., Gould, N. I. M., and Toint, P. L. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, Apr 2011a. ISSN 1436-4646.

Cartis, C., Gould, N. I. M., and Toint, P. L. Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst case function and derivative evaluation complexity. *Mathematical Programming*, 127(2):245–295, Apr 2011b. ISSN 1436-4646.

Conn, A. R., Gould, N. I., and Toint, P. L. *Trust region methods*. SIAM, 2000.

Dünner, C., Forte, S., Takáč, M., and Jaggi, M. Primal-dual rates and certificates. In *International Conference on Machine Learning*, 2016.

Gargiani, M. Hessian-cocoa: a general parallel and distributed framework for non-strongly convex regularizers. Master's thesis, ETH Zurich, 2017.

Hsieh, C.-J., Si, S., and Dhillon, I. S. Communication-Efficient Parallel Block Minimization for Kernel Machines. *arXiv*, August 2016.

Jaggi, M., Smith, V., Takáč, M., Terhorst, J., Krishnan, S., Hofmann, T., and Jordan, M. I. Communication-efficient distributed dual coordinate ascent. In *Neural Information Processing Systems*, 2014.

Karimireddy, S. P., Stich, S. U., and Jaggi, M. Global linear convergence of Newton's method without strong-convexity or Lipschitz gradients. *arXiv*, June 2018.

Lee, C.-p. and Chang, K.-W. Distributed block-diagonal approximation methods for regularized empirical risk minimization. *arXiv preprint arXiv:1709.03043*, 2017.

Lee, C.-p. and Wright, S. J. Inexact successive quadratic approximation for regularized optimization. *arXiv preprint arXiv:1803.01298*, 2018.

Lee, C.-p., Lim, C. H., and Wright, S. J. A distributed quasi-newton algorithm for empirical risk minimization with nonsmooth regularization. In *KDD 2018 - The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, August 2018.

Lee, J. D., Lin, Q., Ma, T., and Yang, T. Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity. *arXiv preprint arXiv:1507.07595*, 2015.

Mahajan, D., Keerthi, S. S., and Sundararajan, S. A distributed block coordinate descent method for training l 1 regularized linear classifiers. *Journal of Machine Learning Research*, 18(91):1–35, 2017.

Nesterov, Y. and Polyak, B. T. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Niu, F., Recht, B., Ré, C., and Wright, S. J. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Neural Information Processing Systems*, 2011.

Reddi, S. J., Konečnỳ, J., Richtárik, P., Póczós, B., and Smola, A. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Richtárik, P. and Takáč, M. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 17:1–25, 2016.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

Shamir, O., Srebro, N., and Zhang, T. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pp. 1000–1008, 2014.

Smith, V., Forte, S., Ma, C., Takáč, M., Jordan, M. I., and Jaggi, M. CoCoA: A General Framework for Communication-Efficient Distributed Optimization. *Journal of Machine Learning Research (and arXiv:1611.02189)*, 2018.

Trofimov, I. and Genkin, A. Distributed coordinate descent for generalized linear models with regularization. *Pattern Recognition and Image Analysis*, 27(2):349–364, June 2017.

Wang, S., Roosta-Khorasani, F., Xu, P., and Mahoney, M. W. Giant: Globally improved approximate newton method for distributed optimization. *arXiv preprint arXiv:1709.03528*, 2017.

Yang, T. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pp. 629–637, 2013.

Zhang, Y. and Lin, X. Disco: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning*, pp. 362–370, 2015.

Zheng, S., Wang, J., Xia, F., Xu, W., and Zhang, T. A general distributed dual coordinate optimization framework for regularized loss minimization. *Journal of Machine Learning Research*, 18(115):1–52, 2017.

Appendix

A. Analysis

In order to prove convergence of Algorithm 1 we proceed as follows:

1. For the auxilary model with block diagonal $\tilde{H}$ as described in Section 2.1, we lower bound the decrease in the model achieved in each step of Algorithm 1. This yields the lower bound on $\mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha})$ provided in Lemma 3.
2. For iterations that are successful (i.e., they achieve $\rho_t \geq \xi$ and thus successfully decrease the objective function, see Definition 1), the construction of Algorithm 1 allows us to relate the model decrease to the function decrease $F(\boldsymbol{\alpha}) - F(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})$ through the constant ξ. This lets us establish a convergence rate in terms of the number of successful iterations, which is shown in Lemma 5, 6 for non-strongly convex g_i and strongly convex g_i, respectively.
3. Finally, in order to establish overall convergence of Algorithm 1 we need to bound the number of unsuccessful iterations (i.e., iterations for which $\rho_t < \xi$ where no update is applied to the model parameters). This is accomplished by showing that the construction of the sequence $\{\sigma_t\}_{t\geq 0}$ is such that the number of successive unsuccessful iterations is limited and Algorithm 1 will therefore eventually yields a successful iteration which will allow us to decrease the objective function. In details, this is accomplished as follows:
   - show that Algorithm 1 finds a successful step as soon as the penalty parameter σ_t exceeds some critical value, thereby the sequence $\{\sigma_t\}_{t\geq 0}$ is guaranteed to stay within some bounded positive interval.
   - use the boundedness of σ_t to establish an upper bound on the maximum number of unsuccessful iterations and hence the total number of steps to reach a target suboptimality.
   - lastly in Section A.7 we establish the boundedness of the sequence $\{\sigma_t\}_{t\geq 0}$ for two general situations.

A.1. Model Decrease

Lemma 3. *Assume f is $\frac{1}{\tau}$-smooth and g_i are μ-strongly convex with $\mu \geq 0$. Then, the per-step model decrease of Algorithm 1 can be lower bounded as:*

$$\begin{aligned}&\mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha}^{(t)})\\ &\quad\geq (1-\eta)\left[\kappa\mathcal{G}(\boldsymbol{\alpha}^{(t)}) - \frac{\kappa^2}{2}R^{(t)}\right],\end{aligned}$$

where $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ denotes the duality gap, $\kappa \in (0,1]$ and

$$\begin{aligned}R^{(t)} :=&\sigma_t(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})\\ &- \tfrac{\mu(1-\kappa)}{\kappa}\|\boldsymbol{\alpha}^{(t)} - \mathbf{u}^{(t)}\|_2^2\end{aligned}$$

with $u_i^{(t)} \in \partial g_i^*(A_{:,i}^\top \nabla f(A\boldsymbol{\alpha}^{(t)}))$[5].

Proof. Given that the updates $\Delta\boldsymbol{\alpha}_{[k]}$ optimize the respective local models (defined in (7)) η-approximately, we can relate the model decrease provided by $\Delta\boldsymbol{\alpha} = \sum_k \Delta\boldsymbol{\alpha}_{[k]}$ to the optimal model decrease as follows:

$$\begin{aligned}\mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha}^{(t)}) &= \mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - \sum_k \mathcal{M}_{\sigma_t}^{(k)}(\Delta\boldsymbol{\alpha}_{[k]};\boldsymbol{\alpha}^{(t)})\\ &\geq \mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - \sum_k[(1-\eta)\mathcal{M}_{\sigma_t}^{(k)}(\Delta\boldsymbol{\alpha}_{[k]}^\star;\boldsymbol{\alpha}^{(t)}) + \eta\mathcal{M}_{\sigma_t}^{(k)}(\mathbf{0};\boldsymbol{\alpha}^{(t)})]\\ &= \mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - [(1-\eta)\mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}^\star;\boldsymbol{\alpha}^{(t)}) + \eta\mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)})]\\ &= (1-\eta)[\mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}^\star;\boldsymbol{\alpha}^{(t)})],\end{aligned}$$

where $\Delta\boldsymbol{\alpha}_{[k]}^\star = \arg\min_{\Delta\boldsymbol{\alpha}_{[k]}} \mathcal{M}_{\sigma_t}^{(k)}(\Delta\boldsymbol{\alpha}_{[k]}^\star;\boldsymbol{\alpha}^{(t)})$ and $\Delta\boldsymbol{\alpha}^\star = \sum_k \Delta\boldsymbol{\alpha}_{[k]}^\star$.

[5] g_i^* denotes the convex conjugate of the function g_i, which is defined as $g_i^*(u) := \sup_v[uv - g_i(v)]$.

From here we proceed by bounding the model decrease for the optimal update, i.e., $\Delta_{\mathcal{M}} := \mathcal{M}_{\sigma_t}(\mathbf{0};\boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}^\star;\boldsymbol{\alpha}^{(t)})$ which, using (2), can be written as

$$\Delta_{\mathcal{M}} \quad = \quad -\nabla f(A\boldsymbol{\alpha})^\top A\Delta\boldsymbol{\alpha}^\star - \frac{\sigma}{2}\sum_k \Delta\boldsymbol{\alpha}^{\star}_{[k]}{}^\top \tilde{H}(\boldsymbol{\alpha})\Delta\boldsymbol{\alpha}^\star_{[k]} + \sum_i g_i(\alpha_i) - \sum_i g_i((\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}^\star)_i).$$

where we omit the superscript t for reasons of readability. Since $\Delta\boldsymbol{\alpha}^\star$ is the minimizer of $\mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha};\boldsymbol{\alpha})$ the following inequality must hold for an arbitrary update direction $\tilde{\mathbf{s}}$:

$$\Delta_{\mathcal{M}} \quad \geq \quad -\nabla f(A\boldsymbol{\alpha})^\top A\tilde{\mathbf{s}} - \frac{\sigma}{2}\sum_k \tilde{\mathbf{s}}_{[k]}^\top \tilde{H}(\boldsymbol{\alpha})\tilde{\mathbf{s}}_{[k]} + \sum_i g_i(\alpha_i) - g_i((\boldsymbol{\alpha}+\tilde{\mathbf{s}})_i). \tag{11}$$

Hence, let us consider the specific update $\tilde{\mathbf{s}} = \kappa(\mathbf{u}-\boldsymbol{\alpha})$ for some $\kappa \in (0,1]$ and $\mathbf{u} \in \mathbb{R}^n$. We find

$$\Delta_{\mathcal{M}} \quad \geq \quad -\nabla f(A\boldsymbol{\alpha})^\top A\kappa(\mathbf{u}-\boldsymbol{\alpha}) - \frac{\kappa^2\sigma}{2}\sum_k (\mathbf{u}-\boldsymbol{\alpha})_{[k]}^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha})_{[k]} \tag{12}$$

$$+\sum_i g_i(\alpha_i) - \sum_i g_i((\boldsymbol{\alpha}+\kappa(\mathbf{u}-\boldsymbol{\alpha}))_i). \tag{13}$$

Furthermore, using μ-strong convexity of g_i with $\mu \geq 0$, (i.e., the bound also holds for $\mu = 0$ in which case g_i is convex), we get

$$g_i((1-\kappa)\alpha_i + \kappa u_i) \leq (1-\kappa)g_i(\alpha_i) + \kappa g_i(u_i) - \frac{\mu}{2}\kappa(1-\kappa)(\alpha_i - u_i)^2,$$

which combined with (13) yields

$$\begin{aligned}\Delta_{\mathcal{M}} \quad \geq \quad & \kappa\sum_i g_i(\alpha_i) - \kappa\sum_i g_i(u_i) + \sum_i \frac{\mu}{2}\kappa(1-\kappa)(\alpha_i-u_i)^2 \\ & -\nabla f(A\boldsymbol{\alpha})^\top A\kappa(\mathbf{u}-\boldsymbol{\alpha}) - \frac{\kappa^2\sigma}{2}\sum_k (\mathbf{u}-\boldsymbol{\alpha})_{[k]}^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha})_{[k]} \\ = \quad & \underbrace{\kappa\left[\sum_i g_i(\alpha_i) - \sum_i g_i(u_i) - \nabla f(A\boldsymbol{\alpha})^\top A(\mathbf{u}-\boldsymbol{\alpha})\right]}_{\text{(gap)}} + \frac{\mu}{2}\kappa(1-\kappa)\|\boldsymbol{\alpha}-\mathbf{u}\|_2^2 \\ & -\frac{\kappa^2\sigma}{2}\sum_k (\mathbf{u}-\boldsymbol{\alpha})_{[k]}^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha})_{[k]}. \end{aligned} \tag{14}$$

To further simplify this bound we choose $\mathbf{u}$ such that $u_i \in \partial g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}))$ where $\mathbf{x}_i$ denote the columns of the data matrix A, $\mathbf{w}(\boldsymbol{\alpha}) := \nabla f(A\boldsymbol{\alpha})$ and g_i^* denotes the convex conjugate of the function g_i. For this particular choice the term "(gap)" in (14) corresponds to the duality gap of the objective at the iterate $\boldsymbol{\alpha}$. To see this, note that the duality gap (see, e.g., (Dünner et al., 2016)) for (1) can be written as

$$\begin{aligned}\mathcal{G}(\boldsymbol{\alpha}) \quad & = \quad \sum_i g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) + g_i(\alpha_i) + \alpha_i \mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}) \\ & \overset{(a)}{=} \quad \sum_i u_i(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha})) - g_i(u_i) + g_i(\alpha_i) + \alpha_i \mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}) \\ & = \quad \sum_i g_i(\alpha_i) - g_i(u_i) - (u_i - \alpha_i)\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}), \end{aligned} \tag{15}$$

where equality (a) holds for any $u_i \in \partial g_i^*(-\mathbf{x}_i^\top \mathbf{w})$ since for such an optimal u_i the Fenchel-Young inequality holds with equality, i.e.,

$$g_i(u_i) = u_i(-\mathbf{x}_i^\top \mathbf{w}) - g_i^*(-\mathbf{x}_i^\top \mathbf{w}). \tag{16}$$

Now combining (14) with (15) and (16) we find

$$\Delta_{\mathcal{M}} \quad \geq \quad \kappa\mathcal{G}(\boldsymbol{\alpha}) + \frac{\mu}{2}\kappa(1-\kappa)\|\boldsymbol{\alpha}-\mathbf{u}\|_2^2 - \frac{\kappa^2\sigma}{2}\sum_k (\mathbf{u}-\boldsymbol{\alpha})_{[k]}^\top \tilde{H}(\boldsymbol{\alpha})(\mathbf{u}-\boldsymbol{\alpha})_{[k]} \tag{17}$$

and Lemma 3 follows. □

A.2. Function Decrease

In order to relate the model decrease to the function decrease we use the fact that every update $\Delta\boldsymbol{\alpha}$ applied to the parameter vector in Algorithm 1 is successful in the following sense.

Definition 1 (successful update)**.** The update $(\Delta\boldsymbol{\alpha}, \sigma_t)$ is called successful if the following inequality is satisfied:

$$\xi \leq \rho_t := \frac{F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha})}{F(\boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)})} \tag{18}$$

otherwise it is called unsuccessful.

Lemma 4. *The function decrease of Algorithm 1 for a successful update $(\Delta\boldsymbol{\alpha}, \sigma_t)$ can be bounded as:*

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \geq \xi(1-\eta)\left[\kappa\mathcal{G}(\boldsymbol{\alpha}^{(t)}) - \frac{\kappa^2}{2}R^{(t)}\right],$$

where $\kappa \in (0,1]$ and $R^{(t)}$ is defined as in Lemma 3.

Proof. Starting from (18), and observing that $F(\boldsymbol{\alpha}^{(t)}) = \mathcal{M}_{\sigma_t}(\mathbf{0}; \boldsymbol{\alpha}^{(t)})$ we have

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \geq \xi(\mathcal{M}_{\sigma_t}(\mathbf{0}; \boldsymbol{\alpha}^{(t)}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)})). \tag{19}$$

Combining this inequality with Lemma 3 concludes the proof. □

A.3. Rate of Convergence

Let S denote the set of successful iterations as

$$S := \{t \geq 0 : \text{ iteration } t \text{ is successful in the sense of Definition 1}\}$$

Further, let us define two disjoint index sets $\mathcal{U}_T$ and $\mathcal{S}_T$, which represent the un- and successful steps that have occurred up to some iteration $T > 0$;

$$S_T := \{t \leq T : t \in S\}$$
$$U_T := \{t \leq T : t \notin S\}.$$

Now, we will use Lemma 4 to establish convergence of Algorithm 1 as a function of the number of successful iterations. Therefore, we will start with convex functions g_i where we show sublinear convergence and then show that for strongly convex functions g_i, this result can be improved to obtain a linear rate of convergence.

A.3.1. NON-STRONGLY CONVEX g_i.

Lemma 5. *(non-strongly convex g_i) Let f be $\frac{1}{\tau}$-smooth and g_i be convex with L-bounded support. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded above by σ_{sup}. Then, we can bound the suboptimality of Algorithm 1 as*

$$F(\boldsymbol{\alpha}^{(T)}) - F(\boldsymbol{\alpha}^\star) \leq \frac{2(C_5\sigma_{sup} + \varepsilon_0)}{\xi(1-\eta)}\frac{1}{|S_T|}$$

where $|S_T|$ counts the number of successful updates up to iteration T, $\varepsilon_0 := F(\boldsymbol{\alpha}_0) - F(\boldsymbol{\alpha}^\star)$ and $C_5 = \frac{4}{\tau}R^2L^2$ with $\|A_{:,i}\| \leq R \ \forall i$.

Proof. For non-strongly convex g_i (i.e., $\mu = 0$) we know from Lemma 4 that for any for successful update $(\Delta\boldsymbol{\alpha}, \sigma_t)$ the function decrease at iteration t can be lower bounded as

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \geq \xi(1-\eta)\left[\kappa\mathcal{G}(\boldsymbol{\alpha}^{(t)}) - \frac{\kappa^2\sigma_t}{2}(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})^\top\tilde{H}(\boldsymbol{\alpha}^{(t)})(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})\right]. \tag{20}$$

For our block diagonal hessian approximation

$$\tilde{H}(\boldsymbol{\alpha}) = \mathbf{diag}(A_{:\mathcal{I}_1}^\top\nabla^2 f(A\boldsymbol{\alpha})A_{:\mathcal{I}_1}, \ldots A_{:\mathcal{I}_K}^\top\nabla^2 f(A\boldsymbol{\alpha})A_{:\mathcal{I}_K})$$

it holds that

$$(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})^\top\tilde{H}(\boldsymbol{\alpha}^{(t)})(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}) \leq \frac{1}{\tau}\|A(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})\|^2 \leq \frac{4}{\tau}R^2L^2 \tag{21}$$

with $\|A_{:,i}\| \leq R \quad \forall i$. Inequality (21) relies on the assumption that g_i has L-bounded support: a) by duality between Lipschitzness and Bounded-Support (Dünner et al., 2016) of the univariate functions g_i we have $|\alpha_i| \leq L$ since α_i is in the support of g_i and b) by the equivalence between Lipschitzness and bounded subgradient we also have $|u_i| \leq L$ since $u_i \in \partial g_i^*(-\mathbf{x}_i^\top \mathbf{w}(\boldsymbol{\alpha}))$. Together this yields $|u_i - \alpha_i| \leq 4L^2$ and the bound (21) follows.

In the following we assume that the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded by σ_{sup}. We write $\varepsilon^{(t)} := F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star)$ for the suboptimality at step t and use that the duality gap upper-bounds the suboptimality, i.e, $\mathcal{G}(\boldsymbol{\alpha}^{(t)}) \geq \varepsilon^{(t)}$. Combining this with (20) yields

$$\varepsilon^{(t+1)} \leq (1 - \kappa\xi(1-\eta))\varepsilon^{(t)} + \xi(1-\eta)\frac{2\kappa^2}{\tau}R^2L^2\sigma_{\text{sup}}. \tag{22}$$

Let a, b being positive constants defined as $a = \xi(1-\eta)$ and $b = \frac{4}{\tau}R^2L^2\sigma_{\text{sup}}$, then the above inequality can be written as

$$\varepsilon^{(t+1)} \leq (1 - \kappa a)\varepsilon^{(t)} + \frac{\kappa^2}{2}ab. \tag{23}$$

and holds for any $\kappa \in (0,1]$. Now let us choose κ to minimize the RHS of (23) which yields $\kappa = \frac{1}{b}\varepsilon^{(t)}$ and to have $\kappa \in (0,1]$ we further constrain κ to

$$\kappa = \min\left\{1, \frac{1}{b}\varepsilon^{(t)}\right\}.$$

Now let us consider the two cases separately:

1. $\underline{\varepsilon^{(t)} \geq b.}$ In this case we choose $\kappa = 1$. Thus, from (23) we get
$$\varepsilon^{(t+1)} \leq (1-a)\varepsilon^{(t)} + \frac{ab}{2} \leq (1-a)\varepsilon^{(t)} + \frac{a}{2}\varepsilon^{(t)} = \left(1 - \frac{a}{2}\right)\varepsilon^{(t)} \tag{24}$$
2. $\underline{\varepsilon^{(t)} < b.}$ In this case we choose $\kappa = \frac{1}{b}\varepsilon^{(t)}$ and hence from (23) we get
$$\varepsilon^{(t+1)} \leq \varepsilon^{(t)} - \frac{a}{2b}\varepsilon^{(t)}\varepsilon^{(t)}. \tag{25}$$

Note that the inequalities (24) and(25) together with non-negativity of $\varepsilon^{(t)}$ and $a \in (0,1)$ imply that $\varepsilon^{(t+1)} \leq \varepsilon^{(t)}$ and thus $\{\varepsilon^{(t)}\}$ is a decreasing sequence. Combining the two inequalities (24) and (25) we get the following bound which holds for both cases and thus for every t:

$$\varepsilon^{(t+1)} \leq \varepsilon^{(t)} - \frac{a}{2}\min\left\{\frac{1}{b}\varepsilon^{(t)}, 1\right\}\varepsilon^{(t)} \leq \varepsilon^{(t)} - \frac{a}{2}\min\left\{\frac{1}{b+\varepsilon_0}\varepsilon^{(t)}, 1\right\}\varepsilon^{(t)} \leq \varepsilon^{(t)} - \frac{a}{2}\frac{1}{b+\varepsilon_0}\varepsilon^{(t)}\varepsilon^{(t)}. \tag{26}$$

where we used $\varepsilon_0 \geq \varepsilon^{(t)} \; \forall t > 0$. Thus it holds that

$$\varepsilon^{(t+1)} \quad \leq \quad \varepsilon^{(t)} - \frac{a}{2}\frac{1}{b+\varepsilon_0}\varepsilon^{(t)}\varepsilon^{(t+1)} \tag{27}$$

Deviding both sides by $\varepsilon^{(t)}\varepsilon^{(t+1)}$ yields

$$\frac{1}{\varepsilon^{(t+1)}} \quad \geq \quad \frac{1}{\varepsilon^{(t)}} - \frac{a}{2}\frac{1}{b+\varepsilon_0} \tag{28}$$

Applying this bound recursively and plugging in the definition of a, b concludes the proof. □

A.3.2. STRONGLY-CONVEX g_i.

Lemma 6. *(strongly convex g_i) Let f be $\frac{1}{\tau}$-smooth and g_i μ-strongly convex with $\mu > 0$. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded above by σ_{sup}. Then, we can bound the suboptimality $\varepsilon^{(t)} := F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star)$ as*

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \left(1 - \xi(1-\eta)\frac{\mu\tau}{c_A\sigma_{sup} + \mu\tau}\right)^{|S_t|}\varepsilon^{(0)}$$

where $c_A = \max_k \|A_{[k]}\|^2$ and $|S_t|$ denotes the cardinality of the set S_t which counts the number of successful updates up to iteration t.

Proof. For μ-strongly convex g_i with $\mu > 0$ we can choose $\kappa = \hat{\kappa}^{(t)}$ such that $R^{(t)} \leq 0$ in Lemma 3. That is

$$\hat{\kappa}^{(t)} = \frac{\mu\tau}{c_A\sigma_t + \mu\tau}$$

since

$$\begin{aligned} R^{(t)} &= \sigma_t(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})^\top \tilde{H}(\boldsymbol{\alpha}^{(t)})(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}) - \frac{\mu(1-\kappa)}{\kappa}\|\boldsymbol{\alpha}^{(t)} - \mathbf{u}^{(t)}\|_2^2 \\ &= \sigma_t \sum_k (A_{[k]}(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})_{[k]})^\top \nabla^2 f(A\boldsymbol{\alpha})(A_{[k]}(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})_{[k]}) - \frac{\mu(1-\kappa)}{\kappa}\|\boldsymbol{\alpha}^{(t)} - \mathbf{u}^{(t)}\|_2^2 \\ &\leq \frac{\sigma_t}{\tau}\sum_k \|A_{[k]}(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})_{[k]}\|_2^2 - \frac{\mu(1-\kappa)}{\kappa}\|\boldsymbol{\alpha}^{(t)} - \mathbf{u}^{(t)}\|_2^2 \\ &\leq \left(\frac{c_A\sigma_t}{\tau} - \frac{\mu(1-\kappa)}{\kappa}\right)\|\boldsymbol{\alpha}^{(t)} - \mathbf{u}^{(t)}\|_2^2 \end{aligned}$$

where $c_A = \max_k \|A_{[k]}\|^2$.

Hence, by Lemma 4, for successful updates $(\Delta\boldsymbol{\alpha}, \sigma_t)$, the function decrease at iteration t can be lower bounded as

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \geq \xi(1-\eta)\left[\frac{\mu\tau}{c_A\sigma_t + \mu\tau}\right]\mathcal{G}(\boldsymbol{\alpha}^{(t)}).$$

While, by construction of Algorithm 1, the iterate remains unchanged over unsuccessful iterations. Let us denote the suboptimality at iteration t as $\varepsilon^{(t)} = F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star)$. Then, using the fact that the duality gap always upper bounds the suboptimality, i.e., $\mathcal{G}(\boldsymbol{\alpha}^{(t)}) \geq F(\boldsymbol{\alpha}^{(t)})$ we get the following recursion:

$$\varepsilon^{(t+1)} \leq \left(1 - \xi(1-\eta)\left[\frac{\mu\tau}{c_A\sigma_t + \mu\tau}\right]\right)\varepsilon^{(t)}.$$

Using that fact that the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded by σ_{sup} we can establish the following rate of convergence

$$\varepsilon^{(t)} \leq \left(1 - \xi(1-\eta)\left[\frac{\mu\tau}{c_A\sigma_{\text{sup}} + \mu\tau}\right]\right)^{|S_t|}\varepsilon^{(0)}$$

□

A.4. Bound on number of successful steps

From Lemma 6 and Lemma 5 in the previous section the following two lemmas follow immediately:

Lemma 7 (Number of successful iterations)**.** *Let f be $\frac{1}{\tau}$ smooth and g_i μ-strongly convex. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded above by σ_{sup}. Then, Algorithm 1 achieves a suboptimality $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \varepsilon$ after*

$$\frac{1}{\log(C_2^{-1})}\log\left(\frac{\varepsilon_0}{\varepsilon}\right) \tag{29}$$

successful iterations where $C_2 \in (0,1)$ is a constant defined as $C_2 = 1 - \xi(1-\eta)\frac{\mu\tau}{c_A\sigma_{\text{sup}} + \mu\tau}$ and ε_0 denotes the initial suboptimaliy $\varepsilon_0 = F(\boldsymbol{\alpha}^{(0)}) - F(\boldsymbol{\alpha}^\star)$.

Lemma 8 (Number of successful iterations)**.** *Let be f $\frac{1}{\tau}$-smooth and g_i convex functions with L-bounded support. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded above by σ_{sup}. Then, Algorithm 1 achieves a suboptimality $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \varepsilon$ after*

$$C_1\frac{1}{\varepsilon} \tag{30}$$

successful iterations where $C_1 = \frac{2\left[\frac{4}{\tau}L^2R^2\sigma_{sup} + \varepsilon_0\right]}{\xi(1-\eta)}$

A.5. Bound on number of unsuccessful steps

We assume there exists a $\sigma_{\text{sup}} < \infty$ such that $\sigma_t \leq \sigma_{\text{sup}}$ for all $t \geq 0$ (we will later in Section A.7 show the existence of such a bound). As a consequence, the algorithm may only take a limited number of consecutive unsuccessful steps and hence the total number of unsuccessful iterations is at most a problem dependent constant times the number of successful iterations plus some additive term depending on the initialization σ_0. The following Lemma is motivated by (Cartis et al., 2011b, Corollary 5.5):

Lemma 9 (Number of unsuccessful iterations)**.** *Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded above by σ_{sup}. Then, for any fixed $T \geq 0$, it holds that*

$$|U_T| \leq \frac{1}{\log(\gamma)} \log\left(\frac{\sigma_{sup}}{\sigma_0}\right) + |S_T| \tag{31}$$

Proof. Since it holds that $\xi \leq \frac{1}{\zeta}$ we have

$$\sigma_{t+1} = \gamma\sigma_t \;\; \forall t \in U_T$$
$$\sigma_{t+1} \geq \frac{1}{\gamma}\sigma_t \;\; \forall t \in S_T$$

Thus, we deduce inductively

$$\sigma_0 \gamma^{-|S_T|}\gamma^{|U_T|} \leq \sigma_T$$

since σ_T is bounded by σ_{sup} we have

$$\log\left(\frac{1}{\gamma}\right)|S_T| + \log(\gamma)|U_T| \leq \log\left(\frac{\sigma_{\text{sub}}}{\sigma_0}\right)$$

Recall $\gamma > 1$ and hence

$$|U_T| \leq \frac{1}{\log(\gamma)} \log\left(\frac{\sigma_{\text{sup}}}{\sigma_0}\right) + |S_T| \tag{32}$$

□

Remark 1. *Assume we start with a save value $\sigma_0 \geq \sigma_{sup}$, then (32) simplifies to $|U_T| \leq 1 + |S_T|$*

A.6. Final convergence result

In order to prove the final convergence results it remains to combine Lemma 6 and Lemma 5 with Lemma 9 in order to bound the total number of iterations $T = |S_T| + |U_T|$ required in Algorithm 1 to reach a required suboptimality. Doing so results in the following two corollaries:

Corollary 10 (Number of successful and unsuccessful iterations)**.** *Let f be $\frac{1}{\tau}$-smooth and g_i μ-strongly-convex. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded above by σ_{sup}. Then Algorithm 1 reaches an accuracy $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \varepsilon$ within a total number of*

$$\frac{1}{\log(\gamma)} \log\left(\frac{\sigma_{sup}}{\sigma_0}\right) + \frac{2}{\log(C_2^{-1})} \log\left(\frac{\varepsilon_0}{\varepsilon}\right) \tag{33}$$

steps, where $C_2 \in (0,1)$ is a constant defined as $C_2 = 1 - \xi(1-\eta)\frac{\mu\tau}{c_A\sigma_{\text{sup}}+\mu\tau}$.

Proof. Combining Lemma 9 and Lemma 7 this result follows immediately. □

Corollary 11 (Number of successful and unsuccessful iterations)**.** *Let f be $\frac{1}{\tau}$-smooth and g_i have L-bounded support. Assume the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded above by σ_{sup}. Then, Algorithm 1 reaches an accuracy $F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \varepsilon$ within a total number of*

$$\frac{1}{\log(\gamma)} \log\left(\frac{\sigma_{sup}}{\sigma_0}\right) + 2C_1\frac{1}{\varepsilon} \tag{34}$$

steps, where $C_1 > 0$ is a constant defined as $C_1 = \frac{2\left[\frac{4}{\tau}L^2R^2\sigma_{sup}+\varepsilon_0\right]}{\xi(1-\eta)}$.

Remark 2. *The first term in (34) is the price we potentially pay for a bad initialization σ_0 (the number of unsuccessful steps before the first successful step happens). If we choose a safe initial value $\sigma_0 \geq \sigma_{sup}$ this first term can be upper-bounded by 1.*

A.7. Boundness of the sequence $\{\sigma_t\}_{t\geq 0}$

So far we have assumed that there exists a $\sigma_{\text{sup}} < \infty$ that bounds the sequence $\{\sigma_t\}_{t\geq 0}$ generated by Algorithm 1. In this section we will explicitly give such an upper bound under different conditions on f. To achieve this we show that if σ is large enough the auxiliary model builds a global upper bound on the objective function and Algorithm 1 must yield a successful step.

A.7.1. QUASI-SELF-CONCORDANT f

Let us assume the function f is quasi self-concordant (Bach et al., 2010). This assumption is not very restrictive and fulfilled by most prominent machine learning applications including logistic regression. We will first state the definition of quasi-self concordance together with some useful properties and then explicitly state an upper bound σ_t.

Definition 2 (multivariate quasi self-concordant functions)**.** f is quasi self-concordant if $\forall \mathbf{w}, \mathbf{v} \in \mathbb{R}^n$ the function $\phi(t) = f(\mathbf{w} + t\mathbf{v})$ satisfies $|\phi'''(t)| \leq M_f \|\mathbf{v}\|_2 \phi''(t)$ for some $M_f \geq 0$.

Proposition 12. *(Bach et al., 2010, Proposition 1) Let f be a quasi self-concordant function with constant M_f, then*

$$f(\mathbf{w}+\mathbf{v}) \leq f(\mathbf{w}) + \mathbf{v}^\top \nabla f(\mathbf{w}) + \frac{\mathbf{v}^\top \nabla^2 f(\mathbf{w}) \mathbf{v}}{M_f^2 \|\mathbf{v}\|_2}(e^{M_f\|\mathbf{v}\|_2} - M_f\|\mathbf{v}\|_2 - 1)$$

and

$$\nabla^2 f(\mathbf{w}+\mathbf{v}) \preceq e^{M_f\|\mathbf{v}\|_2} \nabla^2 f(\mathbf{w})$$

Lemma 13 (safe bound on σ_t)**.** *Assume f be a quasi self-concordant function. Then, for every iteration $t \geq 0$ of Algorithm 1 we have $\sigma_t \leq \sigma_{sup}$, where*

$$\sigma_{sup} := 2K\gamma \left[\frac{e^{M_f\|\Delta\boldsymbol{\alpha}\|_2} - M_f\|\Delta\boldsymbol{\alpha}\|_2 - 1}{M_f^2\|\Delta\boldsymbol{\alpha}\|_2} \right] \tag{35}$$

Proof. To prove Lemma 13 we show that for $\sigma_t \geq \frac{\sigma_{\text{sup}}}{\gamma}$ the model forms an upper bound on the objective, i.e. $\mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)}) \geq F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \quad \forall \Delta\boldsymbol{\alpha}$. Hence, for every η-approximate update $\Delta\boldsymbol{\alpha}$ we have $\rho_t \geq 1 \geq \xi$ and hence a successful step. In order to establish this bound on σ_t we use Proposition 12 which yields

$$\begin{aligned}
&f(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \\
&\quad \leq f(A\boldsymbol{\alpha}^{(t)}) + \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A\Delta\boldsymbol{\alpha} + (A\Delta\boldsymbol{\alpha})^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A\Delta\boldsymbol{\alpha} \left[\frac{e^{M_f\|\Delta\boldsymbol{\alpha}\|_2} - M_f\|\Delta\boldsymbol{\alpha}\|_2 - 1}{M_f^2\|\Delta\boldsymbol{\alpha}\|_2} \right] \\
&\quad \overset{(i)}{\leq} f(A\boldsymbol{\alpha}^{(t)}) + \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A\Delta\boldsymbol{\alpha} + K\sum_k (A_{[k]}\Delta\boldsymbol{\alpha}_{[k]})^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)}) A_{[k]}\Delta\boldsymbol{\alpha}_{[k]} \left[\frac{e^{M_f\|\Delta\boldsymbol{\alpha}\|_2} - M_f\|\Delta\boldsymbol{\alpha}\|_2 - 1}{M_f^2\|\Delta\boldsymbol{\alpha}\|_2} \right] \\
&\quad = f(A\boldsymbol{\alpha}^{(t)}) + \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A\Delta\boldsymbol{\alpha} + K\sum_k \Delta\boldsymbol{\alpha}_{[k]}{}^\top \tilde{H}(\boldsymbol{\alpha}^{(t)}) \Delta\boldsymbol{\alpha}_{[k]} \left[\frac{e^{M_f\|\Delta\boldsymbol{\alpha}\|_2} - M_f\|\Delta\boldsymbol{\alpha}\|_2 - 1}{M_f^2\|\Delta\boldsymbol{\alpha}\|_2} \right].
\end{aligned}$$

In (i) we used Jensen's inequality for convex functions, i.e., $f(\frac{1}{n}\sum_i x_i) \leq \frac{1}{n}\sum_i f(x_i)$.
Hence,

$$\begin{aligned}
F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) &= f(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) + \sum_i g_i((\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha})_i) \\
&\leq \mathcal{M}_{\sigma_t = \frac{1}{\gamma}\sigma_{\text{sup}}}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)})
\end{aligned}$$

since any $\sigma > \frac{1}{\gamma}\sigma_{\text{sup}}$ is guaranteed to yield a successful step and in every iteration σ is at most increased by γ, hence, σ_{sup} provides an upper bound on $\{\sigma_t\}_{t\geq 0}$

□

A.7.2. f WITH LIPSCHITZ CONTINUOUS HESSIAN

In this section we will show that the theoretical bound on σ_{sup} derived in the previous section can be refined when posing stronger assumptions of f. Therefore, let us pose the following two widely used assumptions on f:

Assumption 1. *Assume the Hessian of $f(A\boldsymbol{\alpha})$ and $\tilde{H}(\boldsymbol{\alpha})$ agree along this direction of the step, i.e.,*

$$\|(A^\top \nabla^2 f(\mathbf{v}) A - \tilde{H}(\boldsymbol{\alpha}))\Delta\boldsymbol{\alpha}_k\| \leq C\|\Delta\boldsymbol{\alpha}\| \quad \forall t > 0 \ \textit{ and some } C > 0$$

Assumption 2. *Assume the Hessian of f is globally Lipschitz continuous, i.e.,*

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^n \ \textit{ and some } L > 0$$

Under these conditions we can show that the sequence $\{\sigma_t\}_{t\geq 0}$ is bounded by σ_{sup} as defined in the following lemma:

Lemma 14. *Assume f satisfies Assumptions 1 and 2, then, the sequence $\{\sigma_t\}_{t\geq 0}$ in Algorithm 1 is bounded above by*

$$\sigma_{sup} = \frac{\gamma}{2\|\tilde{H}(\boldsymbol{\alpha}^{(t)})\|}\left(L + C + 1\right). \tag{36}$$

Proof. In order to prove Lemma 14 we show that from $\sigma \geq \frac{1}{2\|\tilde{H}(\boldsymbol{\alpha})\|}\left(L + C + 1\right)$ the Algorithm must yield a successful step. This can be achieved by proving that for $\sigma = \sigma_{\text{sup}}$ the model globally upper bounds the objective function, i.e., $F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)}) < 0$ and hence $\rho_t \geq 1 > \xi$. Therefore we bound $F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)})$ as

$$\begin{aligned}
F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)}) \quad &\leq \quad f(A(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha})) - \left[f(A\boldsymbol{\alpha}^{(t)}) + \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A\Delta\boldsymbol{\alpha} + \frac{\sigma}{2}\Delta\boldsymbol{\alpha}^\top H(\boldsymbol{\alpha}^{(t)})\Delta\boldsymbol{\alpha}\right] \\
&= \quad f(A\boldsymbol{\alpha}^{(t)}) + \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A\Delta\boldsymbol{\alpha} + \frac{1}{2}(A\Delta\boldsymbol{\alpha})^\top \nabla^2 f(A\boldsymbol{\beta})A\boldsymbol{\alpha}^{(t)} \\
&\quad\quad - \left[f(A\boldsymbol{\alpha}^{(t)}) + \nabla f(A\boldsymbol{\alpha}^{(t)})^\top A\Delta\boldsymbol{\alpha} - \frac{\sigma}{2}\Delta\boldsymbol{\alpha}^\top \tilde{H}(\boldsymbol{\alpha}^{(t)})\Delta\boldsymbol{\alpha}\right] \\
&= \quad \frac{1}{2}\Delta\boldsymbol{\alpha}^\top \left[A^\top \nabla^2 f(A\boldsymbol{\beta}^{(t)})A - \sigma\tilde{H}(\boldsymbol{\alpha}^{(t)})\right]\Delta\boldsymbol{\alpha}
\end{aligned}$$

which holds for some $\boldsymbol{\beta}^{(t)}$ on the line segment $(\boldsymbol{\alpha}^{(t)}, \boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha})$. We continue by using Assumption 1 and Assumption 2 which yields

$$\begin{aligned}
F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) - \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)}) \quad &= \quad \frac{1}{2}\Delta\boldsymbol{\alpha}^\top \left[A^\top \nabla^2 f(A\boldsymbol{\beta}^{(t)})A - \sigma_t\tilde{H}(\boldsymbol{\alpha}^{(t)}) \pm A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)})A \pm \tilde{H}(\boldsymbol{\alpha}^{(t)})\right]\Delta\boldsymbol{\alpha} \\
&\leq \quad \frac{1}{2}\|A^\top \nabla^2 f(A\boldsymbol{\beta}^{(t)})A - A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)})A\|\|\Delta\boldsymbol{\alpha}\|^2 \\
&\quad\quad + \frac{1}{2}\|(A^\top \nabla^2 f(A\boldsymbol{\alpha}^{(t)})A - \tilde{H}(\boldsymbol{\alpha}^{(t)}))\Delta\boldsymbol{\alpha}\|\|\Delta\boldsymbol{\alpha}\| + \frac{1}{2}(1 - \sigma_t)\|\tilde{H}(\boldsymbol{\alpha}^{(t)})\|\|\Delta\boldsymbol{\alpha}\|^2 \\
&\leq \quad \frac{L}{2}\|\Delta\boldsymbol{\alpha}\|^2 + \frac{C}{2}\|\Delta\boldsymbol{\alpha}\|^3 + \frac{1}{2}(1 - \sigma_t)\|\tilde{H}(\boldsymbol{\alpha}^{(t)})\|\|\Delta\boldsymbol{\alpha}\|^2 \\
&\leq \quad \max(\|\Delta\boldsymbol{\alpha}\|^3, \|\Delta\boldsymbol{\alpha}\|^2)\left[\frac{L}{2} + \frac{C}{2} + \frac{1}{2}(1 - \sigma_t)\|\tilde{H}(\boldsymbol{\alpha}^{(t)})\|\right].
\end{aligned}$$

We can conclude the proof by noting that the RHS is negative for $\sigma_t \geq \frac{\sigma_{\text{sup}}}{\gamma}$ as defined in (36) and this guarantees $F(\boldsymbol{\alpha}^{(t)} + \Delta\boldsymbol{\alpha}) \leq \mathcal{M}_{\sigma_t}(\Delta\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t)})$ and hence a successful step for any $\xi \in (0, 1)$. □