DUAL-LOCO: Preserving Privacy Between Features in Distributed Estimation

Christina Heinze[†]   **Brian McWilliams**[*]   **Nicolai Meinshausen**[†]
Seminar for Statistics, ETH Zürich[†]   Disney Research[*]
Zürich, Switzerland
{heinze, meinshausen}@stat.math.ethz.ch
brian@disneyresearch.com

Abstract

We present DUAL-LOCO, a communication-efficient algorithm for distributed statistical estimation. DUAL-LOCO assumes that the data is distributed according to the features rather than the samples. It requires only a single round of communication where low-dimensional random projections are used to approximate the dependencies between features available to different workers. In particular, privacy between features can be preserved since no single worker has access to all coordinates. We show that DUAL-LOCO has bounded approximation error which only depends weakly on the number of workers. We compare DUAL-LOCO against a state-of-the-art distributed optimization method on a real world dataset and show that it obtains better speedups while retaining good accuracy.

1 Introduction

Many statistical estimation tasks amount to solving an optimization problem of the form

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} J(\boldsymbol{\beta}) := \sum_{i=1}^{n} f_i(\boldsymbol{\beta}^\top \mathbf{x}_i) + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 \tag{1}$$

where $\lambda > 0$ is the regularization parameter. The loss functions $f_i(\boldsymbol{\beta}^\top \mathbf{x}_i)$ depend on labels $y_i \in \mathbb{R}$ and linearly on the coefficients, $\boldsymbol{\beta}$ through a vector of covariates, $\mathbf{x}_i \in \mathbb{R}^p$. Furthermore, we assume all f_i to be convex and smooth with Lipschitz continuous gradients. Concretely, when $f_i(\boldsymbol{\beta}^\top \mathbf{x}_i) = (y_i - \boldsymbol{\beta}^\top \mathbf{x}_i)^2$, Eq. (1) corresponds to ridge regression; for logistic regression $f_i(\boldsymbol{\beta}^\top \mathbf{x}_i) = \log\left(1 + \exp\left(-y_i \boldsymbol{\beta}^\top \mathbf{x}_i\right)\right)$.

For large-scale problems, it is no longer practical to solve even relatively simple estimation tasks such as (1) on a single machine. To deal with this, approaches to distributed data analysis have been proposed that take advantage of many cores or computing nodes on a cluster. A common idea which links many of these methods is stochastic optimization. Typically, each of the workers only sees a small portion of the data points and performs incremental updates to a global parameter vector.

A fundamentally different approach to distributing learning is for each worker to only have access to a portion of the available features. Distributing according to the features could be a preferable alternative for several reasons. Firstly, **privacy**. Individual blocks of features could correspond to sensitive information (such as medical records or social security information) from a number of different sources which should be included in the predictive model but is not allowed to be communicated in an un-disguised form. Secondly, for **high-dimensional data**, where p is large relative to n, better scaling can be achieved. This setting is challenging, however, since most loss functions are not separable across coordinates. High-dimensional data is commonly encountered in the fields of

---

[*]Work done while BM was a postdoc at ETH.

bioinformatics, climate science and computer vision. Furthermore, for a variety of prediction tasks it is often beneficial to map input vectors into a higher dimensional feature space, e.g. using deep representation learning or considering higher-order interactions.

Our contribution. In this work we introduce DUAL-LOCO to solve problems of the form (1) in the distributed setting when each worker only has access to a subset of the *features*. DUAL-LOCO is an extension of the LOCO algorithm [1] which was recently proposed for solving distributed ridge regression in this setting. We propose an alternative formulation where each worker instead locally solves a *dual* optimization problem. DUAL-LOCO has a number of practical and theoretical improvements over the original algorithm:

- DUAL-LOCO is applicable to a wider variety of smooth, convex ℓ_2 penalized loss minimization problems encompassing many widely used regression and classification loss functions, including ridge regression, logistic regression and others.

- In §4 we provide a more intuitive and tighter theoretical result which crucially does not depend on specific details of the ridge regression model and has weaker dependence on the number of workers, K.

- We also show that *adding* (rather than concatenating) random features allows for an efficient implementation yet retains good approximation guarantees.

In §5 we report experimental results with a high-dimensional real world computer vision dataset. We compare DUAL-LOCO with COCOA$^+$, a recently proposed state-of-the-art algorithm for distributed dual coordinate ascent [2]. Our experiments show that DUAL-LOCO demonstrates better scaling with K than COCOA$^+$ while retaining a good approximation of the optimal solution. We provide an implementation of DUAL-LOCO in Apache Spark[1]. The portability of this framework ensures that DUAL-LOCO is able to be run in a variety of distributed computing environments.

2 Related work

2.1 Distributed estimation

Recently, several asynchronous stochastic gradient descent (SGD) methods [3, 4] have been proposed for solving problems of the form (1) in a parallel fashion in a multi-core, shared-memory environment and have been extended to the distributed setting. For such methods, large speedups are possible with asynchronous updates when the data is sparse. However, in some problem domains the data collected is dense with many correlated features. Furthermore, the $p \gg n$ setting can result in slow convergence. In the distributed setting, such methods can be impractical since the cost of communicating updates can dominate other computational considerations.

Jaggi et al. proposed a communication-efficient distributed dual coordinate ascent algorithm (CO-COA resp. COCOA$^+$) [5, 2]. Each worker makes multiple updates to its local dual variables before communicating the corresponding primal update. This allows for trading off communication and convergence speed. Notably they show that convergence is actually independent of the number of workers, thus COCOA$^+$ exhibits *strong scaling* with K.

Other recent work considers solving statistical estimation tasks using a single round of communication [6, 7]. However, all of these methods consider only distributing over the rows of the data where an i.i.d. assumption on the observations holds.

On the other hand, few approaches have considered distributing across the columns (features) of the data. This is a more challenging task for both estimation and optimization since the columns are typically assumed to have arbitrary dependencies and most commonly used loss functions are not separable over the features. Recently, LOCO was proposed to solve ridge regression when the data is distributed across the features [1]. LOCO requires a single round to communicate small matrices of randomly projected features which approximate the dependencies in the rest of the dataset (cf. Figure 1). Each worker then optimizes its own sub-problem independently and finally sends its portion of the solution vector back to the master where they are combined. LOCO makes no assumptions

---

[1]Software package and implementation details available at: `http://christinaheinze.github.io/loco-lib/`.

about the correlation structure between features. It is therefore able to perform well in challenging settings where the features are correlated between blocks and is particularly suited when $p \gg n$. Indeed, since the relative dimensionality of local problems decreases when splitting by columns, they are easier in a statistical sense. LOCO makes no assumptions about data sparsity so it is also able to obtain speedups when the data is dense.

2.2 Random projections for estimation and optimization

Random projections are low-dimensional embeddings $\mathbf{\Pi} : \mathbb{R}^\tau \to \mathbb{R}^{\tau_{subs}}$ which approximately preserve an entire subspace of vectors. They have been extensively used to construct efficient algorithms when the sample-size is large in a variety of domains such as: nearest neighbours [8], matrix factorization [9], least squares [10, 11] and recently in the context of optimization [12].

We concentrate on the Subsampled Randomized Hadamard Transform (SRHT), a structured random projection [13]. The SRHT consists of a projection matrix, $\mathbf{\Pi} = \sqrt{\tau/\tau_{subs}}\mathbf{DHS}$ [9] with the definitions: (i) $\mathbf{S} \in \mathbb{R}^{\tau \times \tau_{subs}}$ is a subsampling matrix. (ii) $\mathbf{D} \in \mathbb{R}^{\tau \times \tau}$ is a diagonal matrix whose entries are drawn independently from $\{-1, 1\}$. (iii) $\mathbf{H} \in \mathbb{R}^{\tau \times \tau}$ is a normalized Walsh-Hadamard matrix. The key benefit of the SRHT is that due to its recursive definition the product between $\mathbf{\Pi}^\top$ and $\mathbf{u} \in \mathbb{R}^\tau$ can be computed in $O\left(\tau \log \tau\right)$ time while never constructing $\mathbf{\Pi}$ explicitly.

Figure 1: Schematic for the distributed approximation of a large data set with random projections, used by DUAL-LOCO.

For moderately sized problems, random projections have been used to reduce the dimensionality of the data prior to performing regression [14, 15]. However after projection, the solution vector is in the compressed space and so interpretability of coefficients is lost. Furthermore, the projection of the low-dimensional solution back to the original high-dimensional space is in fact guaranteed to be a *bad* approximation of the optimum [16].

Dual random projections. Recently, [16, 17] studied the effect of random projections on the *dual* optimization problem. For the primal problem in Eq. (1), defining $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$, we have the corresponding dual

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} -\sum_{i=1}^{n} f_i^*(\alpha_i) - \frac{1}{2n\lambda} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \tag{2}$$

where f^* is the conjugate Fenchel dual of f. For example, for squared loss functions $f_i(u) = \frac{1}{2}(y_i - u)^2$, we have $f_i^*(\alpha) = \frac{1}{2}\alpha^2 + \alpha y_i$. For problems of this form, the dual variables can be directly mapped to the primal variables, such that for a vector $\boldsymbol{\alpha}^*$ which attains the maximum of (2), the optimal primal solution has the form $\boldsymbol{\beta}^*(\boldsymbol{\alpha}^*) = -\frac{1}{n\lambda}\mathbf{X}^\top \boldsymbol{\alpha}^*$.

Clearly, a similar dual problem to (2) can be defined in the projected space. Defining $\tilde{\mathbf{K}} = (\mathbf{X}\mathbf{\Pi})(\mathbf{X}\mathbf{\Pi})^\top$ we have

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} -\sum_{i=1}^{n} f_i^*(\alpha_i) - \frac{1}{2n\lambda} \boldsymbol{\alpha}^\top \tilde{\mathbf{K}} \boldsymbol{\alpha}. \tag{3}$$

Importantly, the vector of dual variables does not change dimension depending on whether the original problem (2) or the projected problem (3) is being solved. Under mild assumptions on the loss function, by mapping the solution to this new problem, $\tilde{\boldsymbol{\alpha}}$, back to the original space one obtains a vector $\tilde{\boldsymbol{\beta}}(\tilde{\boldsymbol{\alpha}}) = -\frac{1}{n\lambda}\mathbf{X}^\top \tilde{\boldsymbol{\alpha}}$, which is a *good* approximation to $\boldsymbol{\beta}^*$, the solution to the original problem (1) [16, 17].

2.3 Privacy-aware learning

When multiple parties want to use their data about a set of individuals jointly for statistical estimation without actually sharing their sensitive features due to privacy concerns, few approaches have
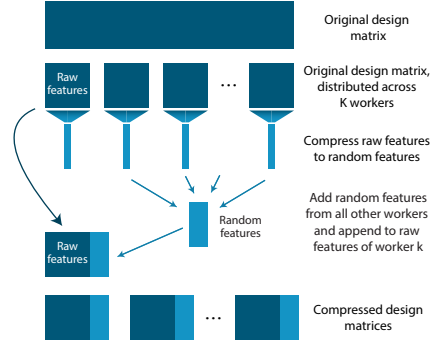
been proposed. The method in [18] collectively applies the same random projection on the rows of the local design matrices which are stored in the separate databases of the involved parties. Subsequently, these compressed matrices are shared publicly while the random projection matrix is kept private. Statistical aggregates such as correlation coefficients can be estimated as column-wise inner products are preserved. However, when applying this scheme to supervised learning, recovery of the true coefficients is problematic [19]. When the parties hold data about the same set of features from different individuals, the random projection is applied column-wise.

One shortcoming of this method is its susceptibility to privacy breaches if an attacker has background knowledge [20]. For instance, in a "known sample attack" the attacker collects the private raw features of more than τ observations where τ is the smallest number of features stored in one the databases. Together with the published compressed features, he can then infer the entries of the random projection matrix. This enables the computation of unbiased estimates of the raw features for any observation (cf. §6.1 in [18]). If the occurrence of such attacks is impossible in a given problem setting, this method may be a suitable option to preserve some degree of privacy.

A more rigorous approach is presented in [21] for the non-distributed setting. Specifically, the raw data is held by one party but it should not be exposed to the learner. The proposed method extends the idea of using a column-wise random projection by additionally perturbing the compressed data matrix with Gaussian noise. This two-stage procedure satisfies differential privacy, so the compressed and perturbed data matrix as well as the random projection matrix can be published. One remaining shortcoming is the lost interpretability of the features. In order to allow for further analyses, such as finding the nearest neighbors of a new input, this new observation has to be projected into the lower-dimensional space where the distances are computed.

3   The DUAL-LOCO algorithm

In this section we detail the DUAL-LOCO algorithm. DUAL-LOCO differs from the original LOCO algorithm in two important ways. **(i)** The random features from each worker are summed, rather than concatenated, to obtain a τ_{subs} dimensional approximation allowing for an efficient implementation in a large-scale distributed environment. **(ii)** Each worker solves a local *dual* problem similar to (3). This allows us to extend the theoretical guarantees to a larger class of estimation problems beyond ridge regression (§4).

We consider the case where p features are distributed across K different workers in non-overlapping subsets $\mathcal{P}_1, \ldots, \mathcal{P}_K$ of equal size[2], $\tau = p/K$.

Algorithm 1 DUAL-LOCO

Input: Data: $\mathbf{X}, Y$, no. workers: K
Parameters: τ_{subs}, λ
1: Partition $\{p\}$ into K subsets of equal size τ and distribute feature vectors in $\mathbf{X}$ accordingly over K workers.
2: **for each** worker $k \in \{1, \ldots K\}$ **in parallel do**
3:     Compute and send random features $\mathbf{X}_k \mathbf{\Pi}_k$.
4:     Receive random features and construct $\bar{\mathbf{X}}_k$.
5:     $\tilde{\boldsymbol{\alpha}}_k \leftarrow \texttt{LocalDualSolver}(\bar{\mathbf{X}}_k, Y, \lambda)$
6:     $\widehat{\boldsymbol{\beta}}_k = -\frac{1}{n\lambda} \mathbf{X}_k^\top \tilde{\boldsymbol{\alpha}}_k$
7:     Send $\widehat{\boldsymbol{\beta}}_k$ to driver.
8: **end for**

Output: Solution vector: $\widehat{\boldsymbol{\beta}} = \left[\widehat{\boldsymbol{\beta}}_1, \ldots, \widehat{\boldsymbol{\beta}}_K \right]$

Since most loss functions of interest are not separable across coordinates, a key challenge addressed by DUAL-LOCO is to define a local minimization problem for each worker to solve *independently* and *asynchronously* while still maintaining important dependencies between features in different blocks and keeping communication overhead low. Algorithm 1 details DUAL-LOCO in full.

We can rewrite (1) making explicit the contribution from block k. Letting $\mathbf{X}_k \in \mathbb{R}^{n \times \tau}$ be the sub-matrix whose columns correspond to the coordinates in $\mathcal{P}_k$ (the "raw" features of block k) and $\mathbf{X}_{(-k)} \in \mathbb{R}^{n \times (p-\tau)}$ be the remaining columns of $\mathbf{X}$, we have

$$J(\boldsymbol{\beta}) = \sum_{i=1}^n f_i \left(\mathbf{x}_{i,k}^\top \boldsymbol{\beta}_{\text{raw}} + \mathbf{x}_{i,(-k)}^\top \boldsymbol{\beta}_{(-k)} \right) + \lambda \left(\|\boldsymbol{\beta}_{\text{raw}}\|_2^2 + \|\boldsymbol{\beta}_{(-k)}\|_2^2 \right). \tag{4}$$

---

[2]This is for simplicity of notation only, in general the partitions can be of different sizes.

Where $\mathbf{x}_{i,k}$ and $\mathbf{x}_{i,(-k)}$ are the rows of $\mathbf{X}_k$ and $\mathbf{X}_{(-k)}$ respectively. We replace $\mathbf{X}_{(-k)}$ in each block with a low-dimensional randomized approximation which preserves its contribution to the loss function. This procedure is described in Figure 1.

In **Step 5**, these matrices of random features are communicated and worker k constructs the matrix

$$\bar{\mathbf{X}}_k \in \mathbb{R}^{n \times (\tau + \tau_{subs})} = \left[\mathbf{X}_k, \sum_{k' \neq k} \mathbf{X}_k \mathbf{\Pi}_k \right], \qquad (5)$$

which is the concatenation of worker k's raw features and the *sum* of the random features from all other workers. $\mathbf{\Pi}$ is the SRHT matrix introduced in §2.2.

As we prove in Lemma 2, summing $\mathbb{R}^\tau \to \mathbb{R}^{\tau_{subs}}$-dimensional random projections from $(K-1)$ blocks is equivalent to computing the $\mathbb{R}^{(p-\tau)} \to \mathbb{R}^{\tau_{subs}}$-dimensional random projection in one go. The latter operation is impractical for very large p and not applicable when the features are distributed. Therefore, summing the random features from each worker allows the dimensionality reduction to be distributed across workers. Additionally, the summed random feature representation can be computed and combined very efficiently.

Furthermore, only sanitized versions of the original features are communicated. This ensures that each worker only has exact information about its local, raw features. Thus, a user with permission to access the data on a specific node or database k is not able to infer the raw features stored on the other nodes without having further background knowledge. If this degree of data perturbation is not sufficient for the problem at hand (cf. discussion in §2.3), the approach from [21] can be used to extend DUAL-LOCO as follows. When adding Gaussian noise to the random features on each worker before communicating, the released perturbed random features satisfy differential privacy [21, 22]. This property guards against a much wider range of possible attacks.

For a single worker the local, approximate primal problem is then

$$\min_{\bar{\boldsymbol{\beta}} \in \mathbb{R}^{\tau + \tau_{subs}}} J_k(\bar{\boldsymbol{\beta}}) := \sum_{i=1}^n f_i(\bar{\boldsymbol{\beta}}^\top \bar{\mathbf{x}}_i) + \frac{\lambda}{2} \|\bar{\boldsymbol{\beta}}\|_2^2 \qquad (6)$$

where $\bar{\mathbf{x}}_i \in \mathbb{R}^{\tau + \tau_{subs}}$ is the ith row of $\bar{\mathbf{X}}_k$. The corresponding dual problem for each worker in the DUAL-LOCO algorithm is

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} -\sum_{i=1}^n f_i^*(\alpha_i) - \frac{1}{2n\lambda} \boldsymbol{\alpha}^\top \tilde{\mathbf{K}}_k \boldsymbol{\alpha}, \quad \tilde{\mathbf{K}}_k = \bar{\mathbf{X}}_k \bar{\mathbf{X}}_k^\top. \qquad (7)$$

The following steps in Algorithm 1 detail respectively how the solution to (7) and the final DUAL-LOCO estimates are obtained.

Step 6. `LocalDualSolver`. The `LocalDualSolver` computes the solution for (7), the local dual problem. The solver can be chosen to best suit the problem at hand. This will depend on the absolute size of n and $\tau + \tau_{subs}$ as well as on their ratio. For example, we could use SDCA [23] or Algorithm 1 from [16].

Step 7. Obtaining the global primal solution. Each worker maps its local dual solution to the primal solution corresponding only to the coordinates in $\mathcal{P}_k$. In this way, each worker returns coefficients corresponding only to its own raw features. The final primal solution vector is obtained by concatenating the K local solutions. Unlike LOCO, we no longer require to discard the coefficients corresponding to the random features for each worker. Consequently, computing estimates is more efficient (especially when $p \gg n$).

4   Approximation error of DUAL-LOCO

In this section we bound the recovery error between the DUAL-LOCO solution and the solution to Eq. (1).

Theorem 1 (DUAL-LOCO error bound). *Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ with rank, r. Assume that the loss $f(\cdot)$ is smooth and Lipschitz continuous. For a subsampling dimension $\tau_{subs} \geq c_1 pK$ where*

$0 \leq c_1 \leq 1/K^2$, let $\boldsymbol{\beta}^*$ be the solution to (1) and $\widehat{\boldsymbol{\beta}}$ be the estimate returned by Algorithm 1. We have with probability at least $1 - \xi$

$$\|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_2 \quad \leq \quad \frac{\varepsilon}{1-\varepsilon} \|\boldsymbol{\beta}^*\|_2 \,, where \quad \varepsilon = \sqrt{\frac{c_0 \log(2r/\delta)r}{c_1 p}} < 1. \tag{8}$$

Proof. By Lemma 4 and applying a union bound we can decompose the global optimization error in terms of the error due to each worker as $\|\boldsymbol{\beta}^* - \widehat{\boldsymbol{\beta}}\|_2 = \sqrt{\sum_{k=1}^{K} \|\boldsymbol{\beta}_k^* - \widehat{\boldsymbol{\beta}}_k\|_2^2} \leq \sqrt{K} \frac{\rho}{1-\rho} \|\boldsymbol{\beta}^*\|_2$, which holds with probability $1 - \xi = 1 - K\left(\delta + \frac{p-\tau}{e^r}\right)$. The final bound, (8) follows by setting $\rho = \sqrt{\frac{c_0 \log(2r/\delta)r}{\tau_{subs}}}$ and $\tau_{subs} \geq c_1 p K$ and noting that $\sqrt{K} \cdot \frac{\frac{\varepsilon}{\sqrt{K}}}{1 - \frac{\varepsilon}{\sqrt{K}}} \leq \frac{\varepsilon}{1-\varepsilon}$. $\qquad\square$

Theorem 1 guarantees that the solution to DUAL-LOCO will be close to the optimal solution obtained by a single worker with access to all of the data. Our result relies on the data having rank $r \ll p$. In practice, this assumption is often fulfilled, in particular when the data is high dimensional. For a large enough projection dimension, the bound has only a weak dependence on K through the union bound used to determine ξ. The error is then mainly determined by the ratio between the rank and the random projection dimension. When the rank of $\mathbf{X}$ increases for a fixed p, we need a larger projection dimension to accurately capture its spectrum. On the other hand, the failure probability increases with p and decreases with r. However, this countering effect is negligible as typically $\log(p - \tau) \ll r$.

5 Implementation and experiments

In this section we demonstrates the empirical performance of DUAL-LOCO in a large, distributed classification task. We implemented DUAL-LOCO in the Apache Spark framework[1].

Cross validation. In most practical cases, the regularization parameter λ is unknown and has to be determined via v-fold cross validation (CV). The chosen algorithm is usually run entirely once for each fold and each of l values of λ, leading to a runtime that is approximately $v \cdot l$ as large as the runtime of a single run[3]. In this context, DUAL-LOCO has the advantage that steps 3 and 4 in Algorithm 1 are independent of λ. Therefore, these steps only need to be performed *once per fold*. In step 5, we then estimate $\tilde{\boldsymbol{\alpha}}_k$ for each value in the provided sequence for λ. Thus, the runtime of DUAL-LOCO will increase by much less than $v \cdot l$ compared to the runtime of a single run. The performance of each value for λ is then not only averaged over the random split of the training data set into v parts but also over the randomness introduced by the random projections which are computed and communicated once per fold. The procedure is provided in full detail in Algorithm 2 in Appendix C.

Competing methods. For the classification example, the loss function is the hinge loss. Although the problem is non-smooth, and therefore not covered by our theory, we still obtain good results suggesting that Theorem 1 can be generalized to non-smooth losses. Alternatively, for classification the smoothed hinge or logistic losses could be used. For the regression problem we use the squared error loss and modify CoCoA$^+$ accordingly. As the `LocalDualSolver` we use SDCA [23]. We ran DUAL-LOCO as well as CoCoA$^+$ on a high-performance cluster[4].

Kaggle Dogs vs Cats dataset. This is a binary classification task consisting of $25,000$ images of dogs and cats[5]. We resize all images to 430×430 pixels and use OVERFEAT [24] – a pre-trained convolutional neural network – to extract $p = 200,704$ *fully dense* feature vectors from the 19th layer of the network for each image. We train on $n_{train} = 20,000$ images and test on the remaining $n_{test} = 5,000$. The size of the training data is 37GB with over 4 billion non-zero elements. All results we report in the following are averaged over five repetitions and by "runtime" we refer to wall clock time.

---

[3]"Approximately" since the cross validation procedure also requires time for testing. For a single run we only count the time it takes to estimate the parameters.

[4]CoCoA$^+$ Spark implementation available from: https://github.com/gingsmith/cocoa.

[5]https://www.kaggle.com/c/dogs-vs-cats

Figure 2 shows the median normalized training and test prediction MSE of DUAL-LOCO and CoCoA$^+$ for different numbers of workers[6]. For DUAL-LOCO, we also vary the size of the random feature representation and choose $\tau_{subs} = \{0.005, 0.01, 0.02\} \times (p-\tau)$. The corresponding errors are labeled with DUAL-LOCO 0.5, DUAL-LOCO 1 and DUAL-LOCO 2. Note that combinations of K and τ_{subs} that would result in $\tau < \tau_{subs}$ cannot be used, e.g. $K = 192$ and $\tau_{subs} = 0.01 \times (p-\tau)$. We ran CoCoA$^+$ until a duality gap of 10^{-2} was attained so that the number of iterations varies for different numbers of workers[7]. Notably, for $K = 48$ more iterations were needed than in the other cases which is reflected in the very low training error in this case. The fraction of local points to be processed per round was set to 10%. We determined the regularization parameter λ via 5-fold cross validation.
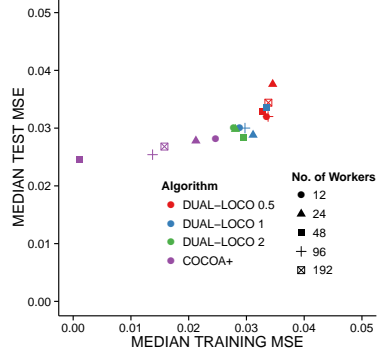

Figure 2: Median normalized training and test MSE based on 5 repetitions.

While the differences in training errors between DUAL-LOCO and CoCoA$^+$ are notable, the differences between the test errors are minor as long as the random feature representation is large enough. Choosing τ_{subs} to be only 0.5% of $p - \tau$ seems to be slightly too small for this data set. When setting τ_{subs} to be 1% of $p - \tau$ the largest difference between the test errors of DUAL-LOCO and CoCoA$^+$ is 0.9%. The averaged mean squared prediction errors and their standard deviations are collected in Table 1 in Appendix C.

Next, we compare the wall clock time needed to find the regularization parameter λ via 5-fold cross validation. For CoCoA$^+$, using the number of iterations needed to attain a duality gap of 10^{-2} would lead to runtimes of more than 24 hours for $K \in \{48, 96, 192\}$ when comparing $l = 20$ possible values for λ. One might argue that using a duality gap of 10^{-1} is sufficient for the cross validation runs which would speed up the model selection procedure significantly as much fewer iterations would be required. Therefore, for $K \geq 48$ we use a duality gap of 10^{-1} during cross validation and a duality gap of 10^{-2} for learning the parameters, once λ has been determined. Figure 3a shows the runtimes when $l = 20$ possible values for λ are compared; The absolute runtime of CoCoA$^+$ for a single run is smaller for $K = 12$ and $K = 24$ and larger for $K \in \{48, 96, 192\}$, so using more workers increased the amount of wall clock time necessary for job completion. The total runtime, including cross validation and a single run to learn the parameters with the determined value for λ, is always smaller for DUAL-LOCO, except when $K = 12$.

Figure 3 shows the relative speedup of DUAL-LOCO and CoCoA$^+$ when increasing K. The speedup is computed by dividing the runtime for $K = 12$ by the runtime achieved for the corresponding $K = \{24, 48, 96, 192\}$. A speedup value smaller than 1 implies an *increase* in runtime. When considering a single run, we run CoCoA$^+$ in two different settings: **(i)** We use the number of iterations that are needed to obtain a duality gap of 10^{-2} which varies for different number of workers[7]. Here, the speedup is smaller than 1 for all K. **(ii)** We fix the number of outer iterations to a constant number. As K increases, the number of inner iterations decreases, making it easier for CoCoA$^+$ to achieve a speedup. We found that although CoCoA$^+$ attains a speedup of 1.17 when increasing K from 12 to 48 (equivalent to a decrease in runtime of 14%), CoCoA$^+$ suffers a 24% increase in runtime when increasing K from 12 to 192.

For DUAL-LOCO 0.5 and DUAL-LOCO 1 we observe significant speedups as K increases. As we split the design matrix by features the number of observations n remains constant for different number of workers. At the same time, the dimensionality of each worker's local problem decreases with K leading to shorter runtimes. In case of DUAL-LOCO 2, the communication costs dominate the costs of computing the random projection and of the `LocalDualSolver`, resulting in much smaller speedups.

Although CoCoA$^+$ was demonstrated to obtain speedups for low-dimensional data sets [2] it is plausible that the same performance cannot be expected on a very high-dimensional data set. This il-

---

[6]In practice, this choice will depend on the available resources in addition to the size of the data set.
[7]For K ranging from 12 to 192, the number of iterations needed were 77, 207, 4338, 1966, resp. 3199.
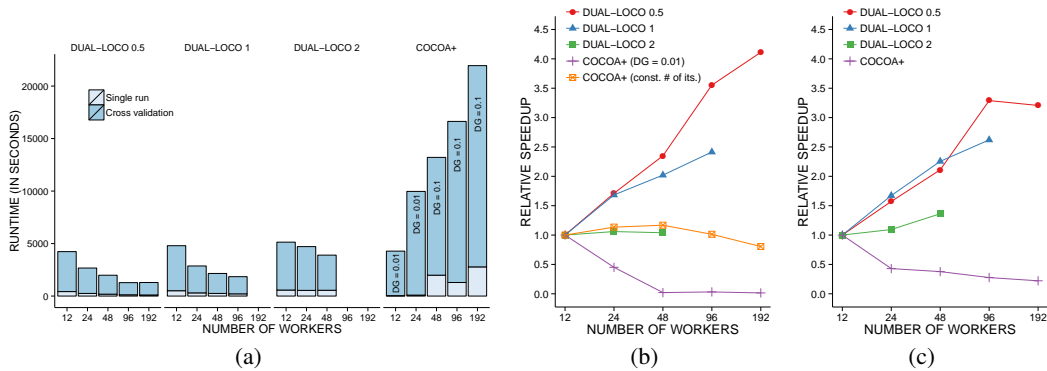
Figure 3: (a) Total wall clock time including 5-fold CV over $l = 20$ values for λ. For COCOA$^+$, we use a duality gap (DG) of 10^{-1} for the CV runs when $K \geq 48$. Relative speedup for (b) a single run and (c) 5-fold CV over $l = 20$ values for λ.

lustrates that in such a high-dimensional setting splitting the design matrix according to the columns instead of the rows is more suitable.

6 Conclusions and further work

We have presented DUAL-LOCO which considers the challenging and rarely studied problem of statistical estimation when data is distributed across features rather than samples. DUAL-LOCO generalizes LOCO to a wider variety of loss functions for regression and classification. We show that the estimated coefficients are close to the optimal coefficients that could be learned by a single worker with access to the entire dataset. The resulting bound is more intuitive and tighter than previous bounds, notably with a very weak dependence on the number of workers. We have demonstrated that DUAL-LOCO is able to recover accurate solutions for large-scale estimation tasks whilst also achieving better scaling than a state-of-the-art competitor, COCOA$^+$, as K increases. Additionally, we have shown that DUAL-LOCO allows for fast model selection using cross-validation.

The dual formulation is convenient for ℓ_2 penalized problems but other penalties are not as straightforward. Similarly, the theory only holds for smooth loss functions. However, as demonstrated empirically DUAL-LOCO also performs well with a non-smooth loss function.

As n grows very large, the random feature matrices may become too large to communicate efficiently even when the projection dimension is very small. For these situations, there are a few simple extensions we aim to explore in future work. One possibility is to first perform row-wise random projections (c.f. [25]) to further reduce the communication requirement. Another option is to distribute $\mathbf{X}$ according to rows and columns.

Contrary to stochastic optimization methods, the communication of DUAL-LOCO is limited to a single round. For fixed n, p and τ_{subs}, the amount of communication is deterministic and can be fixed ahead of time. This can be beneficial in settings where there are additional constraints on communication (for example when different blocks of features are distributed *a priori* across different physical locations).

In future work, we would like to formalize the utility of DUAL-LOCO in the context of privacy-aware learning. When the random features are perturbed by noise in order to satisfy differential privacy, we aim to analyze the resulting loss of precision in the statistical estimation.

References

[1] Christina Heinze, Brian McWilliams, Nicolai Meinshausen, and Gabriel Krummenacher. Loco: Distributing ridge regression with random projections. *arXiv preprint arXiv:1406.3469*, 2014.

[2] Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of The 32nd International Conference on Machine Learning*, 2015.

[3] Feng Niu, Benjamin Recht, Christopher Ré, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.

[4] John C. Duchi, Michael I. Jordan, and H. Brendan McMahan. Estimation, optimization, and parallelism when data is sparse. In *Advances in Neural Information Processing Systems*, 2013.

[5] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[6] Yuchen Zhang, John C Duchi, and Martin J Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *arXiv preprint arXiv:1305.5029*, 2013.

[7] Qiang Liu and Alex T. Ihler. Distributed estimation, information loss and exponential families. In *Advances in Neural Information Processing Systems*, pages 1098–1106, 2014.

[8] Nir Ailon and Bernard Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[9] Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the Subsampled Randomized Hadamard Transform. 2012. arXiv:1204.0062v4 [cs.DS].

[10] Paramveer Dhillon, Yichao Lu, Dean P. Foster, and Lyle Ungar. New Subsampling Algorithms for Fast Least Squares Regression. In *Advances in Neural Information Processing Systems*, 2013.

[11] Brian McWilliams, Gabriel Krummenacher, Mario Lucic, and Joachim M Buhmann. Fast and robust least squares estimation in corrupted linear models. In *Advances in Neural Information Processing Systems*, pages 415–423, 2014.

[12] Mert Pilanci and Martin J Wainwright. Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence. *arXiv preprint arXiv:1505.02250*, 2015.

[13] Joel A. Tropp. Improved analysis of the subsampled randomized Hadamard transform. November 2010. arXiv:1011.1595v4 [math.NA].

[14] Ata Kabán. New bounds on compressive linear least squares regression. In *Artificial Intelligence and Statistics*, 2014.

[15] Yichao Lu, Paramveer Dhillon, Dean P. Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in Neural Information Processing Systems 26*, pages 369–377, 2013.

[16] Lijun Zhang, Mehrdad Mahdavi, Rong Jin, Tianbao Yang, and Shenghuo Zhu. Recovering optimal solution by dual random projection. *arXiv preprint arXiv:1211.3046*, 2012.

[17] Lijun Zhang, Mehrdad Mahdavi, Rong Jin, Tianbao Yang, and Shenghuo Zhu. Random projections for classification: A recovery approach. *IEEE Transactions on Information Theory*, 2014.

[18] Kun Liu, Hillol Kargupta, and Jessica Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. on Knowl. and Data Eng.*, 18(1):92–106, January 2006.

[19] Mert Pilanci and Martin J. Wainwright. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *arXiv preprint arXiv:1411.0347*, 2014.

[20] E. Onur Turgay, Thomas B. Pedersen, Yücel Saygin, Erkay Savas, and Albert Levi. Disclosure risks of distance preserving data transformations. In Bertram Ludäscher and Nikos Mamoulis, editors, *Scientific and Statistical Database Management*, volume 5069 of *Lecture Notes in Computer Science*, pages 79–94. Springer Berlin Heidelberg, 2008.

[21] Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. Privacy via the johnson-lindenstrauss transform. *Journal of Privacy and Confidentiality*, 5(1):39–71, 2013.

[22] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 410–419. IEEE, 2012.

[23] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[24] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Over-feat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*. CBLS, April 2014.

[25] Michael W. Mahoney. Randomized algorithms for matrices and data. April 2011. arXiv:1104.5557v3 [cs.DS].

[26] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. April 2010. arXiv:1004.4389v7 [math.PR].

Supplementary Information for DUAL-LOCO: Distributing Statistical Estimation Using Random Projections

A   Supplementary Results

Here we introduce two lemmas. The first describes the random projection construction which we use in the distributed setting.

Lemma 2 (Summing random features). *Consider the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{V} \in \mathbb{R}^{p \times r}$ have orthonormal columns and $\mathbf{D} \in \mathbb{R}^{r \times r}$ is diagonal; $r = rank(\mathbf{X})$. c_0 is a fixed positive constant. In addition to the raw features, let $\bar{\mathbf{X}}_k \in \mathbb{R}^{n \times (\tau + \tau_{subs})}$ contain random features which result from summing the $K - 1$ random projections from the other workers. Furthermore, assume without loss of generality that the problem is permuted so that the raw features of worker k's problem are the first τ columns of $\mathbf{X}$ and $\bar{\mathbf{X}}_k$. Finally, let*

$$\Theta_S = \begin{bmatrix} \mathbf{I}_\tau & 0 \\ 0 & \mathbf{\Pi} \end{bmatrix} \in \mathbb{R}^{p \times (\tau + \tau_{subs})}$$

such that $\bar{\mathbf{X}}_k = \mathbf{X}\Theta_S$.

With probability at least $1 - \left(\delta + \frac{p - \tau}{e^r}\right)$

$$\|\mathbf{V}^\top \Theta_S \Theta_S^\top \mathbf{V} - \mathbf{V}^\top \mathbf{V}\|_2 \leq \sqrt{\frac{c_0 \log(2r/\delta)r}{\tau_{subs}}}.$$

Proof. See Appendix B. $\square$

Definition 1. *For ease of exposition, we shall rewrite the dual problems so that we consider minimizing convex objective functions. More formally, the original problem is then given by*

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ D(\boldsymbol{\alpha}) := \sum_{i=1}^n f_i^*(\alpha_i) + \frac{1}{2n\lambda} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \right\}. \tag{9}$$

The problem worker k solves is described by

$$\tilde{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \tilde{D}_k(\boldsymbol{\alpha}) := \sum_{i=1}^n f_i^*(\alpha_i) + \frac{1}{2n\lambda} \boldsymbol{\alpha}^\top \tilde{\mathbf{K}}_k \boldsymbol{\alpha} \right\}. \tag{10}$$

Recall that $\tilde{\mathbf{K}}_k = \bar{\mathbf{X}}_k \bar{\mathbf{X}}_k^\top$, where $\bar{\mathbf{X}}_k$ is the concatenation of the τ raw features and τ_{subs} random features for worker k.

To proceed we need the following result which relates the solution of the original problem to that of the approximate problem solved by worker k.

Lemma 3 (Adapted from Lemma 1 [17]). *Let $\boldsymbol{\alpha}^*$ and $\tilde{\boldsymbol{\alpha}}$ be as defined in Definition 1. We obtain*

$$\frac{1}{\lambda}(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*)^\top \left(\mathbf{K} - \tilde{\mathbf{K}}_k\right) \boldsymbol{\alpha}^* \geq \frac{1}{\lambda}(\tilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*)^\top \tilde{\mathbf{K}}_k (\tilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*). \tag{11}$$

Proof. See [17]. $\square$

For our main result, we rely heavily on the following variant of Theorem 1 in [17] which bounds the difference between the coefficients estimated by worker k, $\widehat{\boldsymbol{\beta}}_k$ and the corresponding coordinates of the optimal solution vector $\boldsymbol{\beta}_k^*$.

Lemma 4 (Local optimization error. Adapted from [17]). *For $\rho = \sqrt{\frac{c_0 \log(2r/\delta)r}{\tau_{subs}}}$ the following holds*

$$\|\widehat{\boldsymbol{\beta}}_k - \boldsymbol{\beta}_k^*\|_2 \leq \frac{\rho}{1 - \rho} \|\boldsymbol{\beta}^*\|_2$$

with probability at least $1 - \left(\delta + \frac{p - \tau}{e^r}\right)$.

The proof closely follows the proof of Theorem 1 in [17] which we restate here identifying the major differences.

Proof. Let the quantities $\tilde{D}_k(\boldsymbol{\alpha})$, $\tilde{\mathbf{K}}_k$, be as in Definition 1. For ease of notation, we shall omit the subscript k in $\tilde{D}_k(\boldsymbol{\alpha})$ and $\tilde{\mathbf{K}}_k$ in the following.

By the SVD we have $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$. So $\mathbf{K} = \mathbf{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}\mathbf{U}^\top$ and $\tilde{\mathbf{K}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\top$. We can make the following definitions

$$\gamma^* = \boldsymbol{\Sigma}\mathbf{U}^\top\boldsymbol{\alpha}^*, \qquad \tilde{\gamma} = \boldsymbol{\Sigma}\mathbf{U}^\top\tilde{\boldsymbol{\alpha}}.$$

Defining $\tilde{\mathbf{M}} = \mathbf{V}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{V}$ and plugging these into Lemma 3 we obtain

$$(\tilde{\gamma} - \gamma^*)^\top(\mathbf{I} - \tilde{\mathbf{M}})\gamma^* \geq (\tilde{\gamma} - \gamma^*)^\top\tilde{\mathbf{M}}(\tilde{\gamma} - \gamma^*). \tag{12}$$

We now bound the spectral norm of $\mathbf{I} - \tilde{\mathbf{M}}$ using Lemma 2. Recall that Lemma 2 bounds the difference between a matrix and its approximation by a *distributed* dimensionality reduction using the SRHT.

Using the Cauchy-Schwarz inequality we have for the l.h.s. of (12)

$$(\tilde{\gamma} - \gamma^*)^\top\left(\mathbf{I} - \tilde{\mathbf{M}}\right)\gamma^* \leq \rho\|\gamma^*\|_2\|\tilde{\gamma} - \gamma^*\|_2$$

For the r.h.s. of (12), we can write

$$\begin{aligned}
&(\tilde{\gamma} - \gamma^*)^\top\tilde{\mathbf{M}}(\tilde{\gamma} - \gamma^*) \\
&= \|\tilde{\gamma} - \gamma^*\|_2^2 - (\tilde{\gamma} - \gamma^*)^\top\left(\mathbf{I} - \tilde{\mathbf{M}}\right)(\tilde{\gamma} - \gamma^*) \\
&\geq \|\tilde{\gamma} - \gamma^*\|_2^2 - \rho\|\tilde{\gamma} - \gamma^*\|_2^2 \\
&= (1 - \rho)\|\tilde{\gamma} - \gamma^*\|_2^2.
\end{aligned}$$

Combining these two expressions and inequality (12) yields

$$\begin{aligned}
(1 - \rho)\|\tilde{\gamma} - \gamma^*\|_2^2 &\leq \rho\|\gamma^*\|_2\|\tilde{\gamma} - \gamma^*\|_2 \\
(1 - \rho)\|\tilde{\gamma} - \gamma^*\|_2 &\leq \rho\|\gamma^*\|_2.
\end{aligned} \tag{13}$$

From the definition of γ^* and $\tilde{\gamma}$ above and $\boldsymbol{\beta}^*$ and $\tilde{\boldsymbol{\beta}}$, respectively we have

$$\boldsymbol{\beta}^* = -\frac{1}{\lambda}\mathbf{V}\gamma^*, \qquad \tilde{\boldsymbol{\beta}} = -\frac{1}{\lambda}\mathbf{V}\tilde{\gamma}$$

so $\frac{1}{\lambda}\|\gamma^*\|_2 = \|\boldsymbol{\beta}^*\|_2$ and $\|\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_2 = \frac{1}{\lambda}\|\tilde{\gamma} - \gamma^*\|_2$ due to the orthonormality of $\mathbf{V}$. Plugging this into (13) and using the fact that $\|\boldsymbol{\beta}^* - \tilde{\boldsymbol{\beta}}\|_2 \geq \|\boldsymbol{\beta}_k^* - \hat{\boldsymbol{\beta}}_k\|_2$ we obtain the stated result. $\qquad\square$

B   Proof of Row Summing Lemma

Proof of Lemma 2 . Let $\mathbf{V}_k$ contain the first τ rows of $\mathbf{V}$ and let $\mathbf{V}_{(-k)}$ be the matrix containing the remaining rows. Decompose the matrix products as follows

$$\mathbf{V}^\top\mathbf{V} = \mathbf{V}_k^\top\mathbf{V}_k + \mathbf{V}_{(-k)}^\top\mathbf{V}_{(-k)}$$
$$\text{and}$$
$$\mathbf{V}^\top\boldsymbol{\Theta}_S\boldsymbol{\Theta}_S^\top\mathbf{V} = \mathbf{V}_k^\top\mathbf{V}_k + \tilde{\mathbf{V}}_k^\top\tilde{\mathbf{V}}_k$$

with $\tilde{\mathbf{V}}_k^\top = \mathbf{V}_{(-k)}^\top\boldsymbol{\Pi}$. Then

$$\begin{aligned}
&\|\mathbf{V}^\top\boldsymbol{\Theta}_S\boldsymbol{\Theta}_S^\top\mathbf{V} - \mathbf{V}^\top\mathbf{V}\|_2 \\
&= \|\mathbf{V}_k^\top\mathbf{V}_k + \tilde{\mathbf{V}}_k^\top\tilde{\mathbf{V}}_k - \mathbf{V}_k^\top\mathbf{V}_k - \mathbf{V}_{(-k)}^\top\mathbf{V}_{(-k)}\|_2 \\
&= \|\mathbf{V}_{(-k)}^\top\boldsymbol{\Pi}\boldsymbol{\Pi}^\top\mathbf{V}_{(-k)} - \mathbf{V}_{(-k)}^\top\mathbf{V}_{(-k)}\|_2.
\end{aligned}$$

Since $\boldsymbol{\Theta}_S$ is an orthogonal matrix, from Lemma 3.3 in [13] and Lemma 5, summing $(K - 1)$ independent SRHTs from τ to τ_{subs} is equivalent to applying a single SRHT from $p - \tau$ to τ_{subs}. Therefore we can simply apply Lemma 1 of [15] to the above to obtain the result. $\qquad\square$

Lemma 5 (Summed row sampling). *Let $\mathbf{W}$ be an $n \times p$ matrix with orthonormal columns. Let $\mathbf{W}_1, \ldots, \mathbf{W}_K$ be a balanced, random partitioning of the rows of $\mathbf{W}$ where each matrix $\mathbf{W}_k$ has exactly $\tau = n/K$ rows. Define the quantity $M := n \cdot \max_{j=1,\ldots n} \|e_j^\top \mathbf{W}\|_2^2$. For a positive parameter α, select the subsample size*

$$l \cdot K \geq \alpha M \log(p).$$

Let $\mathbf{S}_{T_k} \in \mathbb{R}^{l \times \tau}$ denote the operation of uniformly at random sampling a subset, T_k of the rows of $\mathbf{W}_k$ by sampling l coordinates from $\{1, 2, \ldots \tau\}$ without replacement. Now denote $\mathbf{SW}$ as the sum of the subsampled rows

$$\mathbf{SW} = \sum_{k=1}^{K} (\mathbf{S}_{T_k} \mathbf{W}_k).$$

Then

$$\sqrt{\frac{(1-\delta)l \cdot K}{n}} \leq \sigma_p(\mathbf{SW})$$

and

$$\sigma_1(\mathbf{SW}) \leq \sqrt{\frac{(1+\eta)l \cdot K}{n}}$$

with failure probability at most

$$p \cdot \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right]^{\alpha \log p} + p \cdot \left[\frac{e^{\eta}}{(1+\eta)^{1+\eta}} \right]^{\alpha \log p}$$

Proof. Define $\mathbf{w}_j^\top$ as the jth row of $\mathbf{W}$ and $M := n \cdot \max_j \|\mathbf{w}_j\|_2^2$. Suppose $K = 2$ and consider the matrix

$$
\begin{aligned}
\mathbf{G}_2 : & = (\mathbf{S}_1 \mathbf{W}_1 + \mathbf{S}_2 \mathbf{W}_2)^\top (\mathbf{S}_1 \mathbf{W}_1 + \mathbf{S}_2 \mathbf{W}_2) \\
& = (\mathbf{S}_1 \mathbf{W}_1)^\top (\mathbf{S}_1 \mathbf{W}_1) + (\mathbf{S}_2 \mathbf{W}_2)^\top (\mathbf{S}_2 \mathbf{W}_2) \\
& \quad + (\mathbf{S}_1 \mathbf{W}_1)^\top (\mathbf{S}_2 \mathbf{W}_2) + (\mathbf{S}_2 \mathbf{W}_2)^\top (\mathbf{S}_1 \mathbf{W}_1).
\end{aligned}
$$

In general, we can express $\mathbf{G} := (\mathbf{SW})^\top (\mathbf{SW})$ as

$$\mathbf{G} := \sum_{k=1}^{K} \sum_{j \in T_k} \left(\mathbf{w}_j \mathbf{w}_j^\top + \sum_{k' \neq k} \sum_{j' \in T_k'} \mathbf{w}_j \mathbf{w}_{j'}^\top \right).$$

By the orthonormality of $\mathbf{W}$, the cross terms cancel as $\mathbf{w}_j \mathbf{w}_{j'}^\top = \mathbf{0}$, yielding

$$\mathbf{G} := (\mathbf{SW})^\top (\mathbf{SW}) = \sum_{k=1}^{K} \sum_{j \in T_k} \mathbf{w}_j \mathbf{w}_j^\top.$$

We can consider $\mathbf{G}$ as a sum of $l \cdot K$ random matrices

$$\mathbf{X}_1^{(1)}, \ldots, \mathbf{X}_1^{(K)}, \ldots, \mathbf{X}_l^{(1)}, \ldots, \mathbf{X}_l^{(K)}$$

sampled uniformly at random without replacement from the family $\mathcal{X} := \{\mathbf{w}_i \mathbf{w}_i^\top : i = 1, \ldots, \tau \cdot K\}$.

To use the matrix Chernoff bound in Lemma 6, we require the quantities $\mu_{\min}$, $\mu_{\max}$ and B. Noticing that $\lambda_{\max}(\mathbf{w}_j \mathbf{w}_j^\top) = \|\mathbf{w}_j\|_2^2 \leq \frac{M}{n}$, we can set $B \leq M/n$.

Taking expectations with respect to the random partitioning ($\mathbb{E}_P$) and the subsampling within each partition ($\mathbb{E}_S$), using the fact that columns of $\mathbf{W}$ are orthonormal we obtain

$$\mathbb{E}\left[\mathbf{X}_1^{(k)}\right] = \mathbb{E}_P \mathbb{E}_S \mathbf{X}_1^{(k)} = \frac{1}{K} \frac{1}{\tau} \sum_{i=1}^{K\tau} \mathbf{w}_i \mathbf{w}_i^\top = \frac{1}{n} \mathbf{W}^\top \mathbf{W} = \frac{1}{n} \mathbf{I}$$

Recall that we take l samples in K blocks so we can define

$$\mu_{\min} = \frac{l \cdot K}{n} \qquad \text{and} \qquad \mu_{\max} = \frac{l \cdot K}{n}.$$

12

Plugging these values into Lemma 6, the lower and upper Chernoff bounds respectively yield

$$\mathbb{P}\left\{\lambda_{\min}\left(\mathbf{G}\right) \leq (1-\delta)\frac{l \cdot K}{n}\right\}$$

$$\leq p \cdot \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{l \cdot K/M} \quad \text{for } \delta \in [0,1), \text{ and}$$

$$\mathbb{P}\left\{\lambda_{\max}\left(\mathbf{G}\right) \geq (1+\delta)\frac{l \cdot K}{n}\right\}$$

$$\leq p \cdot \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{l \cdot K/M} \quad \text{for } \delta \geq 0.$$

Noting that $\lambda_{\min}(\mathbf{G}) = \sigma_p(\mathbf{G})^2$, similarly for $\lambda_{\max}$ and using the identity for $\mathbf{G}$ above obtains the desired result. $\qquad\square$

For ease of reference, we also restate the Matrix Chernoff bound from [13, 26] but defer its proof to the original papers.

Lemma 6 (Matrix Chernoff from [13]). *Let $\mathcal{X}$ be a finite set of positive-semidefinite matrices with dimension p, and suppose that*

$$\max_{\mathbf{A}\in\mathcal{X}} \lambda_{\max}(\mathbf{A}) \leq B$$

Sample $\{\mathbf{A}_1,\ldots,\mathbf{A}_l\}$ uniformly at random from $\mathcal{X}$ without replacement. Compute

$$\mu_{\min} = l \cdot \lambda_{\min}(\mathbb{E}\mathbf{X}_1) \quad \text{and} \quad \mu_{\max} = l \cdot \lambda_{\max}(\mathbb{E}\mathbf{X}_1)$$

Then

$$\mathbb{P}\left\{\lambda_{\min}\left(\sum_i \mathbf{A}_i\right) \leq (1-\delta)\mu_{\min}\right\}$$

$$\leq p \cdot \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu_{\min}/B} \quad \text{for } \delta \in [0,1), \text{ and}$$

$$\mathbb{P}\left\{\lambda_{\max}\left(\sum_i \mathbf{A}_i\right) \geq (1+\delta)\mu_{\max}\right\}$$

$$\leq p \cdot \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{\mu_{\max}/B} \quad \text{for } \delta \geq 0.$$

Algorithm	K	TEST MSE	TRAIN MSE
DUAL-LOCO 0.5	12	0.0343 (3.75e-03)	0.0344 (2.59e-03)
DUAL-LOCO 0.5	24	0.0368 (4.22e-03)	0.0344 (3.05e-03)
DUAL-LOCO 0.5	48	0.0328 (3.97e-03)	0.0332 (2.91e-03)
DUAL-LOCO 0.5	96	0.0326 (3.13e-03)	0.0340 (2.67e-03)
DUAL-LOCO 0.5	192	0.0345 (3.82e-03)	0.0345 (2.69e-03)
DUAL-LOCO 1	12	0.0310 (2.89e-03)	0.0295 (2.28e-03)
DUAL-LOCO 1	24	0.0303 (2.87e-03)	0.0307 (1.44e-03)
DUAL-LOCO 1	48	0.0328 (1.92e-03)	0.0329 (1.55e-03)
DUAL-LOCO 1	96	0.0299 (1.07e-03)	0.0299 (7.77e-04)
DUAL-LOCO 2	12	0.0291 (2.16e-03)	0.0280 (6.80e-04)
DUAL-LOCO 2	24	0.0306 (2.38e-03)	0.0279 (1.24e-03)
DUAL-LOCO 2	48	0.0285 (6.11e-04)	0.0293 (4.77e-04)
CoCoA$^+$	12	0.0282 (4.25e-18)	0.0246 (2.45e-18)
CoCoA$^+$	24	0.0278 (3.47e-18)	0.0212 (3.00e-18)
CoCoA$^+$	48	0.0246 (6.01e-18)	0.0011 (1.53e-19)
CoCoA$^+$	96	0.0254 (5.49e-18)	0.0137 (1.50e-18)
CoCoA$^+$	192	0.0268 (1.23e-17)	0.0158 (6.21e-18)

Table 1: Dogs vs Cats data: Normalized training and test MSE: mean and standard deviations (based on 5 repetitions).

C Supplementary Material for Section 5

Algorithm 2 DUAL-LOCO – cross validation

Input: Data: $\mathbf{X}, Y$, no. workers: K, no. folds: v
Parameters: $\tau_{subs}, \lambda_1, \dots \lambda_l$

1: Partition $\{p\}$ into K subsets of equal size τ and distribute feature vectors in $\mathbf{X}$ accordingly over K workers.
2: Partition $\{n\}$ into v folds of equal size.
3: **for each** fold f **do**
4:    Communicate indices of training and test points.
5:    **for each** worker $k \in \{1, \dots K\}$ **in parallel do**
6:       Compute and send $\mathbf{X}_{k,f}^{train} \mathbf{\Pi}_{k,f}$.
7:       Receive random features and construct $\bar{\mathbf{X}}_{k,f}^{train}$.
8:       **for each** $\lambda_j \in \{\lambda_1, \dots \lambda_l\}$ **do**
9:          $\tilde{\boldsymbol{\alpha}}_{k,f,\lambda_j} \leftarrow \texttt{LocalDualSolver}(\bar{\mathbf{X}}_{k,f}^{train}, Y_f^{train}, \lambda_j)$
10:          $\widehat{\boldsymbol{\beta}}_{k,f,\lambda_j} = -\frac{1}{n\lambda_j} \mathbf{X}_{k,f}^{train \top} \tilde{\boldsymbol{\alpha}}_{k,f,\lambda_j}$
11:          $\hat{Y}_{k,f,\lambda_j}^{test} = \mathbf{X}_{k,f}^{test} \widehat{\boldsymbol{\beta}}_{k,f,\lambda_j}$
12:          Send $\hat{Y}_{k,f,\lambda_j}^{test}$ to driver.
13:       **end for**
14:    **end for**
15:    **for each** $\lambda_j \in \{\lambda_1, \dots \lambda_l\}$ **do**
16:       Compute $\hat{Y}_{f,\lambda_j}^{test} = \sum_{k=1}^{K} \hat{Y}_{k,f,\lambda_j}^{test}$.
17:       Compute $\text{MSE}_{f,\lambda_j}^{test}$ with $\hat{Y}_{f,\lambda_j}^{test}$ and Y_f^{test}.
18:    **end for**
19: **end for**
20: **for each** $\lambda_j \in \{\lambda_1, \dots \lambda_l\}$ **do**
21:    Compute $\text{MSE}_{\lambda_j} = \frac{1}{v} \sum_{f=1}^{v} \text{MSE}_{f,\lambda_j}$.
22: **end for**

Output: Parameter λ_j attaining smallest MSE_{λ_j}