PPDSparse: A Parallel Primal-Dual Sparse Method for Extreme Classification

Ian E.H. Yen[†] Xiangru Huang[‡] Wei Dai[†,*] Pradeep Ravikumar[†] Inderjit Dhillon[‡] Eric Xing[†,*]

[†]Carnegie Mellon University [‡]University of Texas at Austin [*]Petuum Inc.

{eyan, pradeepr}@cs.cmu.edu, {xrhuang,inderjit}@cs.utexas.edu, {wei.dai, eric.xing}@petuum.com

ABSTRACT

Extreme Classification comprises multi-class or multi-label prediction where there is a large number of classes, and is increasingly relevant to many real-world applications such as text and image tagging. In this setting, standard classification methods, with complexity linear in the number of classes, become intractable, while enforcing structural constraints among classes (such as low-rank or tree-structure) to reduce complexity often sacrifices accuracy for efficiency. The recent *PD-Sparse* method addresses this via an algorithm that is sub-linear in the number of variables, by exploiting *primal-dual* sparsity inherent in a specific loss function, namely the max-margin loss. In this work, we extend PD-Sparse to be efficiently parallelized in large-scale distributed settings. By introducing separable loss functions, we can scale out the training, with network communication and space efficiency comparable to those in one-versus-all approaches while maintaining an overall complexity sub-linear in the number of classes. On several large-scale benchmarks our proposed method achieves accuracy competitive to the state-of-the-art while reducing the training time from days to tens of minutes compared with existing parallel or sparse methods on a cluster of 100 cores.

KEYWORDS

Extreme Classification, Primal Dual Method, Sparse Optimization.

1 INTRODUCTION

Extreme Classification comprises Multiclass and Multilabel prediction with a large number of classes, and has become increasingly prevalent in real-world applications such as text and image tagging, where the number of tags can easily go beyond tens or even hundreds of thousands.

In such a setting, standard approaches such as one-versus-all and one-versus-one become intractable due to their high training and prediction complexity (that is at least linear) w.r.t. the number of classes. On a benchmark data set with hundreds of thousand of classes, the training time of the one-versus-all approach could take a few months [11, 34], with a model taking hundreds of gigabytes to store.

Structural Constraints. Many approaches have since been proposed to address this space and computational complexity by imposing structural constraints among the sub-models for each class, such as embedding or hierarchy based, among others. One of the most commonly used structural constraint is based on *low-rank embedding* [5, 17, 36], which projects parameters of different classes to a low-dimensional subspace, and thus reduces the effective number of classes. However, such a low-rank assumption could often be violated in real-world data, for instance data with a power-law label distribution, and thus leads to lower accuracy [1, 34]. One remedy to this caveat is to find *local embeddings* instead of a *global embedding*, which entails a much weaker structural constraint, and leads to higher accuracy [2]. However, to find such local embeddings, one needs to find *nearest neighbors*, which either leads to a high complexity w.r.t. the number of samples, or introduces additional dependency on the quality of clustering, which could be unstable for high dimensional data. Thus despite its superior accuracy on some data sets, the local embedding algorithm [2] could be unstable, and moreover has many more tuning parameters than other methods.

Another commonly used structural constraint is based on a tree hierarchy [7, 14, 24], which finds a tree or a forest that filters a fraction of classes as candidates at each tree node visited. This leads to a *logarithmic* prediction time w.r.t. the number of classes. However, finding an optimal tree that maximizes performance measure is computationally hard; note that a greedy partitioning method could potentially have large cumulative error due to cascading effects. As a result, one often needs to train a large number of trees to get reasonable performance. Moreover, in practice, a good hierarchy might not exist in the first place, so that the tree-based approach often has to trade accuracy for efficiency.

Primal and Dual Sparsity. As [34] pointed out, when the number of classes become large, the collection of model parameters is underdetermined, so that a simple structural constraint of sparsity might be practically useful. For the specific max-margin loss, [34] showed that there exists a naturally sparse solution to the Extreme Classification problem with sub-linear number of non-zeros in both primal and dual variables, which can be exploited to develop an estimation algorithm with sub-linear dependency on the number of classes. Since primal-dual sparsity is naturally satisfied in the Extreme Classification setting, the estimation algorithm often leads to higher accuracy on problems with larger number of classes. However, the max-margin loss employed in [34] has several disadvantages:

KDD '17, August 13-17, 2017, Halifax, NS, Canada

DOI: 10.1145/3097983.3098083

(i) it is not separable w.r.t. classes and thus requires optimizing parameters of all classes together, which can lead to a high memory consumption that prohibits its application to larger problems, (ii) the non-separability w.r.t. classes prevents a simple parallelization scheme that the one-versus-all methods enjoy, (iii) the max-margin loss focuses on the margin between the most confusing classes, which as a criterion is sensitive to *mis-labeling*, such as missing positive labels.

A recent work [1] shows that an one-versus-all based approach with weight truncation (for model compression) can reduce the training time substantially via parallelization, albeit the approach has no theoretical guarantee on the resulting model quality. This leads to the question: can we develop a method that enjoys both the *parallelizability*, *small memory footprint* of the one-versus-all technique and the sub-linear complexity of *primal-dual sparse* method?

In this work, we propose a *greedy algorithm* that enjoys both the low runtime complexity inherent in the primal-dual sparse approach *and* one-versus-all's simple parallelization training with small memory footprint. Our specific contributions are as follows:

- We show that the loss optimized by the common one-versus-all technique also enjoys a similar *primal-dual sparse* structure presented in [34] when a class-wise bias is added.
- We propose a *greedy active-set algorithm* that optimizes parameters of each class separately *without communication* and thus enjoys both parallelizability and primal-dual sparsity.
- We extend the analysis in [34] in two ways: (i) we bound the number of non-zero dual variables in terms of the *number of positive samples* instead of the *number of confusing samples* that could be growing linearly with the total number of samples for the separable loss considered; (ii) the bound holds not only for the optimal solution but also for any descent iterates during the optimization, which leads to a more realistic analysis for the sub-linear complexity of the algorithm.
- In our experiments on several benchmark data sets with hundreds of thousands of classes, our proposed method achieves accuracy competitive with state-of-the-art. On a cluster of 100 cores, our method is orders of magnitude faster than the existing parallel one-versus-all methods and the sequential PD-Sparse.

2 PROBLEM SETUP AND BACKGROUND

We consider the standard *Empirical Risk Minimization (ERM)* framework for Multilabel/Multiclass classification, where we are given a collection of training samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ with $\boldsymbol{x}_i \in \mathbb{R}^D$ denoting D-dimensional feature vectors that encode relevant information about the instances, and $\boldsymbol{y}_i \in \{-1, 1\}^K$ denoting label vectors such that $y_{ik} = 1$ if k is a correct label for the i-th sample and $y_{ik} = -1$ otherwise. We denote $\mathcal{P}(\boldsymbol{y}) = \{k | y_k = 1\}$ as the set of positive labels and $\mathcal{N}(\boldsymbol{y}) = \{k | y_k = -1\}$ as the set of negative ones for each sample.

Extreme Classification refers to cases when K is extremely large, sometimes as large as N. On the other hand, the number of positive labels per sample $|\mathcal{P}(\boldsymbol{y}_i)|$ is usually pretty small. In Multiclass classification, we have $|\mathcal{P}(\boldsymbol{y}_i)| = 1$, and in most Mulitlabel prediction problems, $|\mathcal{P}(\boldsymbol{y}_i)|$ is typically in the order of tens or less than 10.

In this work we consider classifiers $h : \mathbb{R}^D \to \{-1, 1\}^K$ of the form

$$h_W(\boldsymbol{x}) = sign(W^T \boldsymbol{x} + \boldsymbol{b})$$

where W is a $D \times K$ matrix. Each column of W is denoted as $\boldsymbol{w}_k$, which corresponds to the model parameters of k-th class. For such linear classifiers, given a loss function $L : \mathbb{R}^K \times \{-1, 1\}^K$, we are interested in the ERM approach to classification that solves the following problem

$$W^* \in arg \min_{W \in \mathbb{R}^{D \times K}} \sum_{i=1}^{N} L(W^T \boldsymbol{x}_i, \boldsymbol{y}_i). \tag{1}$$

2.1 Primal & Dual Sparsity

The estimation problem (1) is said to have *primal sparsity* if the solution W^* has a small number of non-zeros $nnz(W^*)$. Interestingly, for specific loss functions, the optimization problem could also have *dual sparsity*.

[34] specifically considered the max-margin loss [8, 9]:

$$L(\boldsymbol{z}, \boldsymbol{y}) = \max_{k_n \in \mathcal{N}(\boldsymbol{y})} \max_{k_p \in \mathcal{P}(\boldsymbol{y})} (1 + z_{k_n} - z_{k_p})_+. \tag{2}$$

For a given $(\boldsymbol{z}, \boldsymbol{y})$, define the set of *active labels* as those labels k that attain the maximum of (2). In other words, a label is active if it is *most confusing*; note that a negative label is more confusing if it gets a higher score, while a positive label is more confusing if its score is lower. Letting k_a denote the average number of active labels ranging over the training dataset, the optimization problem in (1) with the max-margin loss (2) is said to be *dual sparse* if k_a is small relative to K, which entails that only few of the dual variables are non-zeros at a minimizer of the ERM problem. [34] showed that this is indeed typically the case in extreme classification settings, and proposed a method they called *PD-Sparse* that exploits this fact.

[34] further show that, under the loss (2), if one has k_a average number of active labels, there exists a minimizer of the ERM problem (1) with $nnz(W^*) \leq Nk_a$, which can moreover be the unique solution by augmenting the objective (1) with an arbitrarily small ℓ_1-penalty. In other words, the *dual sparsity* (small k_a) implies *primal sparsity* (small $nnz(W^*)$) if

$$k_w := \frac{nnz(W^*)}{D} \leq \frac{Nk_a}{D} \ll K,$$

This results in a model of both primal and dual sparse structure as long as $DK \gg Nk_a$. Note this result is significant since the sparse solution is *not enforced* but *identified*. In other words, it is not a trade-off between accuracy and sparsity. Instead, when $DK \gg Nk_a$, there are many more parameters than constraints in the problem (1), and there exits a very sparse solution among the many possible solutions.

3 FORMULATION: PRIMAL-DUAL SPARSITY WITH SEPARABILITY

A significant disadvantage of the loss (2) is that it is not separable w.r.t. the class parameters $\boldsymbol{w}_1, ..., \boldsymbol{w}_K$, and therefore it requires training all parameters W together. This incurs a much larger

memory consumption than the *one-versus-all* approach even in the presence of sparsity. This prohibits *PD-Sparse* from using a simple parallelization scheme that assigns the training of different classes to different cores—a scheme that could enjoy nearly linear speedup to even a thousand of cores in [1].

To achieve parallelizability and space efficiency, we consider the following *class-separable hinge loss*

$$L(\boldsymbol{z}, \boldsymbol{y}) := \sum_{k=1}^{K} \ell(z_k, y_k) = \sum_{k=1}^{K} \max(1 - y_k z_k, 0) \tag{3}$$

One-versus-all method can be interpreted as minimizing (3) since

$$\min_{W \in \mathbb{R}^{D \times K}} \sum_{i=1}^{N} \sum_{k=1}^{K} \ell(\boldsymbol{w}_k^T x_i, y_{ik}) = \sum_{k=1}^{K} \left(\sum_{i=1}^{N} \ell(\boldsymbol{w}_k^T x_i, y_{ik}) \right) \tag{4}$$

The goal of this section is to show that (4) also permits a *primal-dual sparse* structure similar to [34] when a *bias* term is added to the parameters of each class. We first note that showing (3) has dual sparsity is more difficult, since unlike (2), it penalizes each class separately, so there could potentially be many more active labels for each sample as we discuss below. We thus take two different approaches to show the desired dual sparsity structure.

3.1 Dual Sparsity: Active Labels

We first employ an approach similar to [34], and try to establish dual sparsity in terms of the number of active labels of each sample i. In our case, the set of active labels are characterized as

$$C_i := \{k \mid y_{ik} \langle \boldsymbol{w}_k, \boldsymbol{x}_i \rangle \leq 1\},$$

that is, labels of either wrong predictions or confidence scores ≤ 1 for a particular sample i. This is related to the set of support vectors of each class $\mathcal{S}_k := \{i \mid y_{ik} \langle \boldsymbol{w}_k, \boldsymbol{x}_i \rangle \leq 1\}$. Denote $k_a := \frac{1}{N} \sum_{i=1}^{N} |C_i|$ and $n_a := \frac{1}{K} \sum_{k=1}^{K} |\mathcal{S}_k|$ as the average number of active labels and support vectors. We have

$$N k_a = K n_a.$$

Note when the number of active labels k_a is constant, we have $n_a = N k_a / K$ decreases with K, which results in the *dual sparsity* when $K \gg k_a$.

The following theorem then shows that the dual sparsity also implies a primal-sparse solution when $n_a \ll D$.

Theorem 3.1. *Let $\lambda > 0$ be an arbitrarily small constant and*

$$\boldsymbol{w}^* \in \underset{\boldsymbol{w} \in \mathbb{R}^D}{argmin} \ \lambda \|\boldsymbol{w}\|_1 + \sum_{i=1}^{N} \ell(\boldsymbol{w}^T \boldsymbol{x}_i, y_i). \tag{5}$$

Then for $\{\boldsymbol{x}_i\}_{i=1}^N$ drawn from a continuous distribution *we have*

$$d_k := nnz(\boldsymbol{w}^*) \leq n_a. \tag{6}$$

where $n_a := \{i \mid y_i \langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle \leq 1\}$ is the number of support vectors.

Proof. Let X be an $N \times D$ feature matrix with rows $\{\boldsymbol{x}_i\}_{i=1}^N$. Any optimal solution of (5) satisfies

$$\lambda \boldsymbol{\rho}^* + \sum_{i=1}^{N} \alpha_i^* \boldsymbol{x}_i = \lambda \boldsymbol{\rho}^* + X^T \boldsymbol{\alpha}^* = 0 \tag{7}$$

for some $\boldsymbol{\rho}^* \in \partial(\|\boldsymbol{w}^*\|_1)$ and $\alpha_i^* \in \partial_{z_i} \ell(z_i, y_i)$ with $z_i := \langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b$. Then since hinge loss (and square hinge loss) has $\alpha_i^* \neq 0$ only when $y_i z_i^* \leq 1$, there are at most n_a non-zeros α_i^* in the linear system (7). On the other hand, the subgradient $\boldsymbol{\rho}^*$ of $\|\boldsymbol{w}^*\|_1$ satisfies

$$\boldsymbol{\rho}^* = \begin{cases} 1\,, & w_j^* > 0 \\ -1\,, & w_j^* < 0 \\ \nu\,, & \nu \in [-1, 1], w_j^* = 0. \end{cases}$$

Let $\mathcal{B} := \{j \mid w_j^* \neq 0\}$ be the indexes with non-zeros weights. Then consider equations given by the non-zeros α_is and $|\mathcal{B}|$ rows in (7)

$$-\lambda \ sign([\boldsymbol{w}^*]_{\mathcal{B}}) = X^T \boldsymbol{\alpha}^*.$$

It has $nnz(\boldsymbol{w}^*)$ equations and n_a variables, which, for a feature matrix at *general position*, can be satisfied only if

$$nnz(\boldsymbol{w}^*) = |\mathcal{B}| \leq n_a.$$

A feature matrix X is always at *general position* if its rows $\{\boldsymbol{x}\}_{i=1}^N$ are drawn from a continuous distribution [28]. □

Note the Theorem 3.1 implies that when the number of active labels k_a is constant, both the number of non-zeros in the primal variables d_k and dual variables n_a are proportional to $\frac{N k_a}{K}$. However, there are several drawbacks with Theorem 3.1. First, the loss (3) does not *force* k_a to be small as the max-margin loss (2) does, so in our case k_a could actually increase linearly with K. Second, Theorem 3.1 only analyzes the optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{w}^*)$, which cannot however guarantee the sparsity of $\boldsymbol{w}$ during the intermediate iterates of the training algorithm. Finally, the result (6) holds only for a purely ℓ_1-regularized objective, which is known to yield a non-smooth dual objective that can be hard to optimize in a coordinate-wise fashion. In next section, we thus employ a different approach to resolve these caveats.

3.2 Dual Sparsity: Positive Examples

In this section, we take a more realistic approach which bounds the ℓ_1-norm of primal and dual variables by the *number of positive examples* of each class, which decreases as a function of K even under the separable loss (3). Denote n_p as the average number of positive samples per class, and k_p as the average number of positive labels per sample. We have

$$n_p := \left(\frac{k_p}{K} \right) N \ll N. \tag{8}$$

when K is large. Note unlike the number of active labels k_a, the number of positive labels k_p is a constant that does not grow with K in most of Extreme Classification problems. Therefore, the number of positive samples per class n_p is decreasing with K since $n_p = N k_p / K$.

Now consider a model that incorporates the bias term, where we have an additional feature $x_{i,0} = 1$ for all samples, and an additional weight $w_{k,0}$ for all classes. For ease of optimization in the dual, we consider the following ℓ_1-ℓ_2 (Elastic-Net) regularized objective :

$$\min_{\boldsymbol{w}_k \in \mathbb{R}^D} \ F(\boldsymbol{w}_k) := \lambda \sum_{j=1}^{D} |w_{jk}| + \frac{1}{2} \|\boldsymbol{w}_k\|^2 + \sum_{i=1}^{N} \ell(\boldsymbol{w}_k^T \boldsymbol{x}_i, y_{ik}), \tag{9}$$

which has a dual objective of the form

$$\begin{aligned} \min_{\boldsymbol{\alpha}_k \in \mathbb{R}^N} \quad & G(\boldsymbol{\alpha}_k) := \frac{1}{2}\|\boldsymbol{w}(\boldsymbol{\alpha}_k)\|^2 - \sum_{i=1}^{N} \alpha_{ik} \\ s.t. \quad & \boldsymbol{w}(\boldsymbol{\alpha}_k) = \mathbf{prox}_\lambda(\hat{X}^T \boldsymbol{\alpha}_k), \\ & 0 \le \alpha_{ik} \le 1. \end{aligned} \tag{10}$$

where $\hat{X}$ is $N \times D$ matrix with rows $\{y_{ik}\boldsymbol{x}_i\}_{i=1}^N$, and $\mathbf{prox}_\lambda(.)$ is the proximal operator of the function $\lambda \sum_{j=1}^D |w_j|$. Note we do not penalize bias term in the ℓ_1 regularization, so $w_0 = [\hat{X}^T \boldsymbol{\alpha}_k]_0$. Then the following two theorems bound the ℓ_1-norm of primal and dual parameters respectively.

Theorem 3.2 (ℓ_1-norm of primal variables). *Let $\hat{\boldsymbol{w}} = (\hat{w}_0 = -1, \hat{\boldsymbol{w}}_{-0} = \mathbf{0})$ be a trival solution. For any $\boldsymbol{w}_k$ with $F(\boldsymbol{w}_k) \le F(\hat{\boldsymbol{w}})$, we have*

$$\|\boldsymbol{w}_k\|_1 \le \frac{2n_p^k}{\lambda}. \tag{11}$$

where n_p^k is the number of positive samples of k-th class.

Proof. $\hat{\boldsymbol{w}}$ satisfies

$$F(\hat{\boldsymbol{w}}) = \frac{1}{2} + n_p^k \le 2n_p^k$$

since $\ell(-1, y_{ik}) = 0$ for $y_{ik} = -1$ and $\ell(0, y_{ik}) = 1$ for $y_{ik} = 1$. Then for any $\boldsymbol{w}_k$ of better objective than $\hat{\boldsymbol{w}}$,

$$\lambda\|\boldsymbol{w}_k\|_1 \le F(\boldsymbol{w}_k) \le F(\hat{\boldsymbol{w}}) \le 2n_p^k$$

which yields the result. □

Theorem 3.3 (ℓ_1-norm of dual variables). *Any optimal solution $\boldsymbol{\alpha}_k$ of (10) satisfies*

$$\|\boldsymbol{\alpha}_k^*\|_1 \le 4n_p^k.$$

Proof. Let $\boldsymbol{w}$, $\boldsymbol{\alpha}$ be the primal and dual variables for a particular class. By strong duality, the optimal dual objective (in its maximization form) equals to the optimal primal objective

$$\max_{\boldsymbol{\alpha}} -G(\boldsymbol{\alpha}) = \min_{w} F(\boldsymbol{w})$$

and therefore, any optimal solution $(\boldsymbol{w}^*, \boldsymbol{\alpha}^*)$ satisfies

$$-\frac{1}{2}\|\boldsymbol{w}^*\|^2 + \sum_i \alpha_i^* = \frac{1}{2}\|\boldsymbol{w}^*\|^2 + \lambda \sum_{j=1}^{D} |w_j^*| + \sum_{i=1}^{N} \ell(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle, y_{ik}),$$

which yields the bound

$$\begin{aligned} \|\boldsymbol{\alpha}^*\|_1 &= \|\boldsymbol{w}^*\|^2 + \lambda \sum_{j=1}^{D} |w_j^*| + \sum_{i=1}^{N} \ell(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle, y_{ik}) \\ &\le 2\left(\frac{1}{2}\|\boldsymbol{w}^*\|^2 + \lambda \sum_{j=1}^{D} |w_j^*| + \sum_{i=1}^{N} \ell(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle, y_{ik})\right) \le 4n_p^k \end{aligned}$$

where the last inequality is due to the existence of $\hat{\boldsymbol{w}} = (\hat{w}_0 = -1, \hat{\boldsymbol{w}}_{-0} = 0)$ with primal objective $F(\hat{\boldsymbol{w}}) \le 2n_p^k$. □

Theorem 3.2 and 3.3 successfully bound the ℓ_1-norm of primal, dual variables by $n_p = O(N/K)$. Although a small ℓ_1-norm does not imply small number of non-zeros in general, we show in the next section that the *complexity* of our algorithm is determined by the ℓ_1 norm of primal and dual variables. In particular, in Section 4.2, we propose a random sparsification procedure that finds a sparse approximation to $\boldsymbol{w}_k$ in order to perform efficient greedy search of active coordinates. The procedure is guaranteed to find a sparse solution with number of non-zeros proportional to $\|\boldsymbol{w}_k\|_1^2$. Further, Section 4.3 gives an iteration complexity, and thus a bound on the active size, proportional to $\|\boldsymbol{\alpha}^*\|_1^2$, where $\boldsymbol{\alpha}^*$ is an optimal solution of (10).

4 ALGORITHM

In this section, we propose a greedy algorithm that alternates between the search of potential support vectors and the optimization over a small set of active samples. Note we have an objective (9) for each class k independent of other classes, so in the following we simply use $\boldsymbol{\alpha}$ to denote $\boldsymbol{\alpha}_k$ and $\boldsymbol{w}$ to denote $\boldsymbol{w}_k$ for a particular class k being solved. The algorithm will be performed for each class independently *without communication*, and thus it is inherently easy to parallelize.

4.1 A Greedy Active-Set Method

Instead of searching for the *confusing labels* of each sample as in [34], we propose a greedy algorithm that optimizes (10) by looking for *active samples* of each class. Note the domain of our objective (10) has box constraints $0 \le \alpha_i \le 1$ instead of simplex constraints as in [34], so a Frank-Wolfe-based algorithm used in [34] does not lead to sparse iterates. Here we propose a *greedy active-set coordinate descent* algorithm that alternates between the greedy search of novel active variables and the optimization over an active set.

The objective (10) has gradient of the form

$$\nabla G(\boldsymbol{\alpha}) = \hat{X}^T \boldsymbol{w}(\boldsymbol{\alpha}) - 1, \tag{12}$$

which can be evaluated in time $O(nnz(\boldsymbol{w})\bar{n})$ if the vector $\boldsymbol{w}(\boldsymbol{\alpha})$ is maintained whenever any dual coordinate α_i is changed, where $\bar{n}$ is an upper bound on the number of non-zero in each column of X. On the other hand, when $\boldsymbol{\alpha}$ is changed by $\Delta\boldsymbol{\alpha}$, the maintenance of

$$\boldsymbol{w}(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) = \mathbf{prox}_{\lambda\|.\|_1}(\hat{X}^T \boldsymbol{\alpha} + \hat{X}^T \Delta\boldsymbol{\alpha}) \tag{13}$$

requires a cost of $O(nnz(\Delta\boldsymbol{\alpha})\bar{d})$ where $\bar{d}$ is an upper bound on the number of non-zeros for each row of $\hat{X}$. Note one can exploit the sparsity of both $\hat{X}$ and $\boldsymbol{w}(\boldsymbol{\alpha})$ simultaneously when computing the gradients of all coordinates together as

$$\nabla G(\boldsymbol{\alpha}) = \sum_{j=0}^{D} w_j \hat{X}_{:,j}. \tag{14}$$

Therefore, an efficient algorithm exploits (14) to compute gradients of all coordinate simultaneously while updates only a small number of them to ensure a small $nnz(\Delta\alpha)$. This suggests a greedy strategy that optimizes only coordinates α_i leading to the most progress. The resulting algorithm is summarized in Algorithm 1, where we perform an inner minimization over active set via Randomized Dual Coordinate Descent (Algorithm 2) [13]. The cost of one epoch of Algorithm 2 over the active set is $O(|\mathcal{A}|\bar{d})$. Therefore, each iteration of Algorithm 1 has an overall cost of $O(nnz(\boldsymbol{w})\bar{n} + |\mathcal{A}|\bar{d})$. Note this is a cost sublinear to the size of data matrix $nnz(X)$ if $nnz(\boldsymbol{w}) \ll D$ and $nnz(\boldsymbol{\alpha}) \ll N$. In section 4.2 and 4.3, we give bounds on $nnz(\boldsymbol{w})$ and $|\mathcal{A}|$ that decreases with number of classes K.

Algorithm 1 Greedy Active-Set Algorithm

0. $\boldsymbol{\alpha} = 0$, $\boldsymbol{b} = \mathbf{1}$, $\mathcal{A} = \{i | y_{ik} = 1\}$ and $H_{ii} = \|\boldsymbol{x}_i\|^2, i \in [N]$.
for t=1.....T **do**
1. Compute $\nabla G(\boldsymbol{\alpha})$ via random sparsification (Algorithm 3).
2. $\mathcal{A} \leftarrow$ Pick κ variables $\notin \mathcal{A}$ of largest $-\nabla_{\alpha_i} G(\boldsymbol{\alpha})$.
3. Minimize (10) w.r.t. coordinates in $\mathcal{A}$ via Algorithm 2.
4. Eliminate $\{i \mid \alpha_i = 0 \ \& \ y_{ik} \neq 1\}$ from $\mathcal{A}$.
end for

Algorithm 2 Coordinate Descent for Acitve Subproblem

for s=1.....S **do**
1. Draw $i \in \mathcal{A}$ uniformly at random.
2. Compute $\nabla_i G(\boldsymbol{\alpha}) = y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - 1$.
3. $\Delta\alpha_i \leftarrow \min(\max(\alpha_i - \nabla_i G(\boldsymbol{\alpha})/H_{ii}, 0), U) - \alpha_i$
4. Maintain (13) with update $\Delta\alpha_i$.
end for

4.2 Sublinear-Time Search via Random Sparsification

Given an ℓ_1 norm bounded by (11), we show that the maximum negative gradient found in Step 2 of Algorithm 1 can be approximated with δ precision when replacing $\boldsymbol{w}(\boldsymbol{\alpha})$ with its random sparsified version $\tilde{\boldsymbol{w}}$ obtained from Algorithm 3, where the number of nonzeros in $\tilde{\boldsymbol{w}}$ is bounded by the square of ℓ_1-norm as stated in the following theorem.

Theorem 4.1. *Running the Random Sparsification procededure 3 for* $R = \lceil \frac{2\|\boldsymbol{w}\|_1^2}{\delta^2} \rceil$ *iterations gives a* $\tilde{\boldsymbol{w}}$ *satisfying*

$$nnz(\tilde{\boldsymbol{w}}) \leq \left(\frac{4n_p^2}{\lambda^2} \right) \frac{1}{\delta^2}, \tag{15}$$

with

$$\mathbb{E}[\min_i \ y_i \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] - \min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \leq \delta, \tag{16}$$

Proof. Since the function $f(\boldsymbol{z}) = \min_i \ z_i$ is 1-Lipschitz-continuous, we have

$$\min_i \ y_i \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \leq |\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle|.$$

Taking expectation over $\tilde{\boldsymbol{w}}$ on both sides, we have

$$\begin{aligned} \mathbb{E}[\min_i \ y_i \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] - \min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle &\leq \mathbb{E}[|\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle|] \\ &\leq \sqrt{\mathbb{E}[|\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle|^2]} \end{aligned} \tag{17}$$

from Jensen's inequality. Since $\mathbb{E}[\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle$ by the construction of $\tilde{\boldsymbol{w}}$ in Algorithm 3, the RHS of (17) corresponds to the square root of variance

$$\mathbb{E}[\min_i \langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle] \leq \sqrt{Var[\langle \tilde{\boldsymbol{w}}, \boldsymbol{x}_i \rangle]} = \frac{\|\boldsymbol{w}\|_1}{\sqrt{R}}.$$

The conclusion follows by noticing that $nnz(\tilde{\boldsymbol{w}}) \leq R$ and $\|\boldsymbol{w}\|_1$ satisfies (11). □

Note (16) implies that the greedy coordinate $\hat{i}$ found by approximate search and the coordinate found by exact search $i^* = arg\min_i \ y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - 1$ satisfy

$$\mathbb{E}[\nabla_{\hat{i}} G(\boldsymbol{\alpha})] \leq \nabla_{i^*} G(\boldsymbol{\alpha}) + 2\delta \tag{18}$$

Therefore, by replacing $\boldsymbol{w}$ with $\tilde{\boldsymbol{w}}$, we reduce the cost of gradient computation from the worst-case $O(nnz(\boldsymbol{w})\bar{n}) = O(D\bar{n})$ to $O(nnz(\tilde{\boldsymbol{w}})\bar{n})$ with a 2δ approximation error. In the next section, we will show that setting $\delta = O(N\hat{\epsilon})$ suffices for the global convergence of our Greedy Algorithm 1 to $\frac{1}{N}(G(\boldsymbol{\alpha}) - G^*) \leq \hat{\epsilon}$ for some $\hat{\epsilon} \in (0, 1)$. Therefore, we have

$$nnz(\tilde{\boldsymbol{w}}) = O(\frac{n_p^2}{\lambda^2 N^2 \hat{\epsilon}^2}) = O(\frac{k_p^2}{\lambda^2 K^2 \hat{\epsilon}^2}).$$

which could be much less than D in the Extreme Classification setting.

4.3 Convergence Analysis

In this section, we give an iteration complexity of Algorithm 1 that depends on the ℓ_1 norm of the optimal solution $\boldsymbol{\alpha}^*$. For simplicity of the analysis, we assume that a normalized feature matrix with $\|\boldsymbol{x}_i\| \leq 1$.

Theorem 4.2. *Let* $\boldsymbol{\alpha}^*$ *be an optimal solution of* (10). *The iterates* $\{\boldsymbol{\alpha}^t\}_{t=1}^{\infty}$ *given by Algorithm 1 with Random Sparsification tolerance* $\delta \leq \frac{\epsilon}{4\|\boldsymbol{\alpha}^*\|1}$ *has* $\mathbb{E}[G(\boldsymbol{\alpha}^t)] - G(\boldsymbol{\alpha}^*) \leq \epsilon$ *for any iterate*

$$t \geq \frac{4\|\boldsymbol{\alpha}^*\|_1^2}{\epsilon} + \frac{G(\mathbf{0}) - G^*}{\|\boldsymbol{\alpha}^*\|_1^2}.$$

Proof. Our dual objective (10) is of the form

$$g(\boldsymbol{\alpha}) + h(\boldsymbol{\alpha})$$

where $h(\boldsymbol{\alpha}) := \sum_{i=1}^N h_i(\alpha_i)$ and

$$h_i(\alpha) = \begin{cases} 0, & 0 \leq \alpha \leq 1 \\ \infty, & o.w. \end{cases}$$

and $g(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}(\boldsymbol{\alpha})\|^2 - \sum_{i=1}^N \alpha_i$ is smooth with Lipschitz-continuous gradient $\nabla g(\boldsymbol{\alpha})$. To see this, let $B_\alpha = \{j | w_j(\boldsymbol{\alpha}) \neq 0\}$. The generalized Hessian of $g(\boldsymbol{\alpha})$ is

$$\nabla^2 g(\boldsymbol{\alpha}) = X_{:,B_\alpha} X_{:,B_\alpha}^T$$

which has diagonal elements $\|\boldsymbol{x}_{i,B_\alpha}\|^2$ bounded by $\|\boldsymbol{x}_i\|^2 \leq 1$ and thus a spectral norm bounded by N. Then for any coordinate i, we have the following descent amount since the second derivative of the smooth part of objective is bounded by $\|\boldsymbol{x}_i\|^2 \leq 1$.

$$\min_\eta \ G(\boldsymbol{\alpha} + \eta \boldsymbol{e}_i) - G(\boldsymbol{\alpha}) \leq \min_\eta \ \nabla_i G * \eta + \frac{1}{2}\eta^2 + h_i(\alpha_i + \eta) \tag{19}$$

where $\boldsymbol{e}_i$ is an indicator vector. Note for $i \in \mathcal{A}$, the minimizer of RHS of (19) is 0 since the previous iteration already minimizes our

Algorithm 3 Random Sparsification

INPUT: a vector $\boldsymbol{w} \in \mathbb{R}^D$.
0. $\tilde{\boldsymbol{w}}^0 = 0$.
for $r = 1...R$ **do**
1. Draw $j \in [D]$ with probability $|w_j|/\|\boldsymbol{w}\|_1$.
2. $\tilde{\boldsymbol{w}}^{(r)} \rightarrow \tilde{\boldsymbol{w}}^{(r-1)} + sign(w_j)\boldsymbol{e}_j$
end for
OUTPUT: $\tilde{\boldsymbol{w}} := \frac{\|\boldsymbol{w}\|_1}{R} \tilde{\boldsymbol{w}}^{(R)}$

objective w.r.t. the active set $\mathcal{A}$. And for $i \notin \mathcal{A}$, the minimizer of the RHS of (19) is $-[-\nabla_i G(\boldsymbol{\alpha})]_+^2/2$, which corresponds to the selection criteria at Step 2 of Algorithm 1. Therefore, at each iteration, the coordinate $\hat{\imath}$ and i^* found by the approximate and exact greedy search respectively satisfy

$$\begin{aligned}
&\mathbb{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \le \mathbb{E}[\min_{\eta}\ G(\boldsymbol{\alpha}+\eta \boldsymbol{e}_{\hat{\imath}})] - G(\boldsymbol{\alpha}) \\
&\le \min_{\eta}\ \nabla_{i^*} G * \eta + \frac{1}{2}\eta^2 + h_i(\alpha_i+\eta) + 2\delta\eta \\
&\le \min_{\boldsymbol{\eta}}\ \langle \nabla G, \boldsymbol{\eta}\rangle + \frac{1}{2}\|\boldsymbol{\eta}\|_1^2 + h(\boldsymbol{\alpha}+\boldsymbol{\eta}) + 2\delta\sum_i \eta_i
\end{aligned}$$

where the first inequality is because the update $\Delta\boldsymbol{\alpha}$ is obtained by minimizing objective w.r.t. a working set $\mathcal{A}^{r+1}$ containing $\hat{\imath}$, and the second, third inequalities follow from (18) and the fact that a linear objective subject to ℓ_1 ball has minimizer at the corner respectively. Then we use convexity to obtain a global estimate of descent amount relative to the suboptimality $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*)$:

$$\mathbb{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \tag{20}$$

$$\le \min_{\boldsymbol{\eta}}\ \langle \nabla G, \boldsymbol{\eta}\rangle + \frac{1}{2}\|\boldsymbol{\eta}\|_1^2 + h(\boldsymbol{\alpha}+\boldsymbol{\eta}) + 2\delta\sum_i \eta_i \tag{21}$$

$$\le \min_{q\in[0,1]}\ q(\nabla G, \boldsymbol{\alpha}^* - \boldsymbol{\alpha}) + \frac{q^2}{2}\|\boldsymbol{\alpha}^*\|_1^2 + 2q\delta\|\boldsymbol{\alpha}^*\|_1 \tag{22}$$

$$\le \min_{q\in[0,1]}\ -q(G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*)) + \frac{q^2}{2}\|\boldsymbol{\alpha}^*\|_1^2 + 2q\delta\|\boldsymbol{\alpha}^*\|_1 \tag{23}$$

where the last ineqaulity is from convexity, and the second inequality is from a restriction of optimization space to $\boldsymbol{\eta} = q(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})$ and the fact that for $i \in \mathcal{A}$ the minimizer of (21) has $\eta_i = 0$. Then choosing $\delta \le \frac{G(\boldsymbol{\alpha})-G(\boldsymbol{\alpha}^*)}{4\|\boldsymbol{\alpha}^*\|_1}$ and minimizing the RHS w.r.t. q leads to

$$\mathbb{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \le -\frac{(G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*))^2}{4\|\boldsymbol{\alpha}^*\|_1^2} \tag{24}$$

for iterates with $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*) \le 2\|\boldsymbol{\alpha}^*\|_1^2$ and has

$$\mathbb{E}[G(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha})] - G(\boldsymbol{\alpha}) \le -\|\boldsymbol{\alpha}^*\|_1^2/2 \tag{25}$$

for iterates with $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*) > 2\|\boldsymbol{\alpha}^*\|_1^2$. Note the constant descent amount (25) happens only in the beginning iterates when $G(\boldsymbol{\alpha}) - G(\boldsymbol{\alpha}^*)$ and can happen at most $2(G(\mathbf{0}) - G(\boldsymbol{\alpha}^*))/\|\boldsymbol{\alpha}^*\|_1^2$ times. Considering those iterates of case (24), we have recursive relation $\Delta G^{t+1} - \Delta G^t \le -\frac{(\Delta G^t)^2}{4\|\boldsymbol{\alpha}^*\|_1^2}$ where $\Delta G^t := \mathbb{E}[G(\boldsymbol{\alpha}^t) \mid \boldsymbol{\alpha}^{t-1}] - G(\boldsymbol{\alpha}^*)$. The recursion leads to the conclusion by, for example, Theorem 1 of [26]. □

Theorem 4.2 is significant when combined with Theorem 3.3, which gives us an iteration complexity of

$$t = \frac{4\|\boldsymbol{\alpha}\|_1^2}{\epsilon} \le \frac{64 n_p^2}{\epsilon},$$

and also a bound on the active size $|\mathcal{A}| \le \kappa t \le \kappa\frac{64n_p^2}{\epsilon}$ that depends only on the number of positive examples. Considering the average case where $n_p = Nk_p/K$, for achieving $\frac{1}{N}(G(\boldsymbol{\alpha}) - G^*) \le \hat{\epsilon} \in (0,1)$, we have

$$|\mathcal{A}| = O(\frac{Nk_p^2}{K^2\hat{\epsilon}}).$$

Then the complexity for running Algorithm 1 on all classes is:

$$K * O(nnz(\tilde{\boldsymbol{w}})\bar{n} + |\mathcal{A}|\bar{d}) = O\left(\frac{k_p^2\bar{n}}{K\lambda^2\hat{\epsilon}^2} + nnz(X)\frac{k_p^2}{K\hat{\epsilon}}\right),$$

times the number of iterations, which is a cost proportional to the factor k_p^2/K.

5 PRACTICAL ISSUES

5.1 Parallelization Architecture

The dual objective (10) comprises independent sub-objectives for different classes, and which moreover only differ by the set of labels $\{y_{ik}\}_{i=1}^N$. Under a shared-memory architecture each thread can share the same copy of data matrix X. In our implementation, we employ a two-layer parallelization with 10 computational nodes and 10 threads per node. Dynamic load balancing is performed by one of the thread which assigns new labels to a thread once it finishes its current job. Usually one can enjoy a nearly linear speedup as long as the number of parallel jobs is an order of magnitude less than K. More sophisticated communication and load balancing can also be achieved through distributed shared memory [10, 29] and scheduler [19].

5.2 Loss and Regularization Variants

In practice, we found replacing the hinge loss in (9) with a square hinge loss

$$\ell(z, y) := \frac{1}{2}\max(1 - yz, 0)^2$$

yields a better performance in our empirical evaluation. One can also introduce an additional tuning parameter τ for the ℓ_2 regularization in (9), although in our experiments we found $\tau = 1$ to be the optimal choice out of $\{0.1, 1, 10\}$ for most of the data sets.

5.3 Two-Level Sparsification

The sparsification of $\boldsymbol{w}$ described in section 4.2 reduces the cost of computing (14) from $O(nnz(X))$ to $O(nnz(\tilde{w})\bar{n})$. For some data sets, each column of X is dense ($\bar{n} \approx N$), but most of entries have small magnitude which contribute little to $\nabla G(\boldsymbol{\alpha})$. A trick to further speed up the greedy search is to consider only entries in X with magnitude larger than a threshold. Then by sorting each column of X in descending order of magnitude, one can stop traversing the sorted list once processed all entries of magnitude larger than the threshold. This trick improves the efficiency of the proposed method signficantly for a number of text data sets in our experiments.

6 EXPERIMENTS

In this section, we compare our proposed algorithm with state-of-the-art approaches on multiclass and multilabel problems chosen based on the experimental results shown in the Extreme Classification Repository [1] and [1, 34]. The compared methods are:

[1] https://manikvarma.github.io/downloads/XC/XMLRepository.html

Table 1: Results and statistics for large-scale Multilabel data sets, N_{train}= number of training samples, N_{test}= number of testing samples, T_{train} = training time, T_{test} = testing time, K = number of classes, D = number of features. P@k = top-k accuracy. DiSMEC and PPDSparse are parallelized with 100 cores. We highlight the best result for each metric, except that for T_{train} we highlight best results among single-core solvers (left four) and parallel solvers. For all experiments, we set a memory limit to be 100G. Experiments that exceeded limits are marked *Memory Limit Exceeded* (MLE).

Data	Metrics	FastXML	PfastreXML	SLEEC	PDSparse	DiSMEC	PPDSparse
Amazon-670K	T_{train}	**5624s**	6559s	20904s		174135s	**921.9s**
N_{train}=490449	P@1 (%)	33.12	32.87	35.62		43.00	**43.04**
N_{test}=153025	P@3 (%)	28.98	29.52	31.65	MLE	38.23	**38.24**
D=135909	P@5 (%)	26.11	26.82	28.85		34.93	**34.94**
K=670091	model size	**4.0G**	6.3G	6.6G		8.1G	5.3G
	T_{test}/N_{test}	**1.41ms**	1.98ms	6.94ms		148ms	20ms
WikiLSHTC-325K	T_{train}	**19160s**	20070s	39000s	94343s	271407s	**353s**
N_{train}=1778351	P@1 (%)	50.01	57.17	58.34	60.70	64.00	**64.13**
N_{test}=587084	P@3 (%)	32.83	37.03	36.7	39.62	**42.31**	42.10
D=1617899	P@5 (%)	24.13	27.19	26.45	29.20	**31.40**	31.14
K=325056	model size	14G	16G	650M	**547M**	8.1G	4.9G
	T_{test}/N_{test}	**1.02ms**	1.47ms	4.85ms	3.89ms	65ms	290ms
Delicious-200K	T_{train}	8832.46s	8807.51s	**4838.7s**	5137.4s	38814s	**2869s**
N_{train}=196606	P@1 (%)	**48.85**	26.66	47.78	37.69	44.71	45.05
N_{test}=100095	P@3 (%)	**42.84**	23.56	42.05	30.16	38.08	38.34
D=782585	P@5 (%)	**39.83**	23.21	39.29	27.01	34.7	34.90
K=205443	model size	1.3G	20G	2.1G	**3.8M**	18G	9.4G
	T_{test}/N_{test}	1.28ms	7.40ms	2.685ms	**0.432ms**	311.4ms	275ms
AmazonCat-13K	T_{train}	11535s	13985s	119840s	**2789s**	11828s	**122.8s**
N_{train}=1186239	P@1 (%)	**94.02**	86.06	90.56	87.43	92.72	92.72
N_{test}=306782	P@3 (%)	**79.93**	76.24	76.96	70.48	78.11	78.14
D=203882	P@5 (%)	**64.90**	63.65	62.63	56.70	63.40	63.41
K=13330	model size	9.7G	11G	12G	**15M**	2.1G	355M
	T_{test}/N_{test}	1.21ms	1.34ms	13.36ms	0.87ms	**0.20ms**	1.82ms

- **FastXML** [24]: An efficient and scalable tree-based algorithm. We adopted parameter setting suggested by the solver.
- **PfastreXML** [14]: An efficient and scalable tree ensemble based method improving upon FastXML by minimizing propensity-scored loss at each tree node, which leads to better performance on tail labels. Since all other methods are not adjusted based on the propensity, in our experiment we still measure performance via traditional top-k accuracy.
- **SLEEC** [2]: A non-linear solver that 1) partitions training sample into clusters and 2) compute local embeddings that preserves nearest neighbor structure within each cluster. Because of this composition of components, its performance highly relies on its parameter setting. We adopted settings suggested by the authors for each data set.
- **PDSparse** [34]: A Primal-Dual sparse method that minimizes a max-margin loss with ℓ_1-ℓ_2 regularization and enjoys sublinear complexity w.r.t. the number of classes.
- **DiSMEC** [1]: A distributed and parallelized method that learns one-versus-all classifiers with heuristic model compression (weight truncation). We run this method with 100 cores using the same parallelization framework to our solver.
- **PPDsparse**: The proposed method with 100 cores (10 machines with 10 cores on each machine).

All compared solvers are available from Extreme Classification Repository[1]. Other solvers not compared in this paper are i) *one-vs-all logistic regression*, *one-vs-all SVM*, *multiclass SVM*, *one-vs-all ℓ_1-regularized logistic regression*, implemented in *LibLinear* [12], ii) *Vowpal-Wabbit* [7], iii) *LEML* [36], iv) *RobustXML* [33], v) *PLT* [15], vi) *LPSR-NB* [30]. All of these have been shown less competitive in a number of previous papers [1, 34].

Experiments are conducted on four large-scale multilabel data sets, four medium-scale multilabel data sets and three large-scale multiclass data sets. We adopt data sets used by [1, 14, 24, 34] and also that from the Extreme Classification Repository [1]. Large-scale multilabel data sets are *WikiLSHTC-325K*, *Delicious-200K*, *Amazon-670K* and *AmazonCat-13K*. Medium-scale Multilabel data sets are *Mediamill*, *Bibtex*, *RCV1-2K* and *EURLex-4K*. They can be found at Extreme Classification Repository[1]. Multiclass data sets *LSHTC1*, *Dmoz* and *aloi.bin* are available [2] from authors of [34].

The data statistics and results are shown in Table 1, 2 and 3. For all experiments, we select our hyperparameter λ from $\{0.01, 0.1, 1\}$ and τ from $\{0.1, 1, 10\}$ to maximize the heldout performance. However, we observed that for most of data sets, $\tau = 1$, $\lambda = 0.01$ consistently gives the best performance. For large-scale multilabel datasets

[2]http://www.cs.utexas.edu/~xrhuang/PDSparse/

Table 2: Results and statistics for small Multilabel data sets, N_{train}= number of training samples, N_{test}= number of testing samples, T_{train} = training time, T_{test} = testing time, K = number of classes, D = number of features. P@k = top-k accuracy. DiSMEC and PPDSparse are parallelized with 100 cores. We highlight the best result for each metric, except that for T_{train} we highlight best results among single-core solvers (left four) and parallel solvers.

Data	Metrics	FastXML	PfastreXML	SLEEC	PDSparse	DiSMEC	PPDSparse
Mediamill	T_{train}	276.4s	293.2s	9504s	**23.8s**	**12.15s**	34.1s
N_{train}=30993	P@1 (%)	84.27	84.08	**87.37**	83.64	84.83	84.42
N_{test}=12914	P@3 (%)	67.34	67.45	**72.60**	66.13	67.17	67.26
D=120	P@5 (%)	53.06	53.23	**58.39**	50.90	52.80	52.78
K=101	model size	87M	88M	104M	**20K**	412K	412K
	T_{test}/N_{test}	0.27ms	0.37ms	4.95ms	**0.004ms**	0.142ms	0.078ms
Bibtex	T_{train}	21.68s	21.47s	296.86s	**7.71s**	**0.203s**	0.232s
N_{train}=4880	P@1 (%)	63.66	63.18	**64.77**	62.36	63.69	63.69
N_{test}=2515	P@3 (%)	39.42	**39.67**	38.97	36.50	38.80	39.43
D=1836	P@5 (%)	28.60	**29.47**	28.50	26.50	28.30	28.67
K=159	model size	34M	37M	5.2M	**20K**	2.1M	2.5M
	T_{test}/N_{test}	0.64ms	0.73ms	0.70ms	**0.007ms**	0.28ms	0.094ms
RCV1-2K	T_{train}	4874.4s	4947.2s	85212s	**709.5s**	641.1s	**35.0s**
N_{train}=623847	P@1 (%)	91.14	89.79	**91.36**	90.02	90.52	91.08
N_{test}=155962	P@3 (%)	73.35	72.65	**73.38**	71.92	72.31	72.93
D=47236	P@5 (%)	**52.69**	52.23	52.50	51.23	51.25	52.10
K=2456	model size	3.9G	4.1G	1.1G	**1.6M**	209M	23M
	T_{test}/N_{test}	0.87ms	1.08ms	53.95ms	**0.066ms**	1.72ms	0.338ms
EURLex-4K	T_{train}	**315.9s**	324.4s	4543.4s	773.2s	76.07s	**9.95s**
N_{train}=15539	P@1 (%)	70.86	70.33	**79.15**	75.90	70.61	74.61
N_{test}=3809	P@3 (%)	59.06	58.61	**64.09**	61.16	57.56	59.56
D=5000	P@5 (%)	49.58	49.69	**52.09**	50.83	47.33	48.43
K=3993	model size	384M	455M	121M	25M	15M	**9.5M**
	T_{test}/N_{test}	3.65ms	5.43ms	3.67ms	**0.73ms**	2.26ms	1.5ms

Table 3: Results and statistics for Multiclass data sets, N_{train}= number of training samples, N_{test}= number of testing samples, T_{train} = training time, T_{test} = testing time, K = number of classes, D = number of features. DiSMEC and PPDSparse are parallelized with 100 cores. We highlight the best result for each metric, except that for T_{train} we highlight best results among single-core solvers (left four) and parallel solvers.

Data	Metrics	FastXML	PfastreXML	SLEEC	PDSparse	DiSMEC	PPDSparse
aloi.bin	T_{train}	1900.9s	1901.6s	16193s	**139.8s**	92.0s	**7.05s**
N_{train}=100000	accuracy (%)	95.71	93.43	93.74	96.2	96.28	**96.38**
N_{test}=8000	model size	1.3G	1.3G	3.7G	19M	16M	**14M**
D=636911	T_{test}/N_{test}	5.05ms	5.10ms	28.00ms	0.064ms	0.02ms	**0.0178ms**
K=1000							
LSHTC1	T_{train}	1398.2s	1422.4s	5919.3s	**196.6s**	298.8s	**45.8s**
N_{train}=88806	accuracy (%)	22.04	**23.32**	12.2	22.46	22.74	22.70
N_{test}=5000	model size	937M	1.1G	631M	**88M**	142M	381M
D=347255	T_{test}/N_{test}	5.73ms	8.81ms	14.66ms	**0.40ms**	3.7ms	6.94ms
K=12294							
Dmoz	T_{train}	6475.1s	6619.7s	47490s	**2518.9s**	1972.0s	**170.60s**
N_{train}=345068	accuracy (%)	**40.76**	39.78	33.03	39.91	39.38	39.32
N_{test}=38340	model size	3.5G	3.8G	1.5G	**680M**	369M	790M
D=833484	T_{test}/N_{test}	3.29ms	3.20ms	40.43ms	**1.87ms**	4.58ms	6.58ms
K=11947							

(Table 1), we use tf-idf features with sample-wise normalization as suggested by the author of [1].

Our experimental results confirmed several comments from previous work [1, 34]: 1) Among single-core solvers (PDSparse, FastXML, PfastreXML, SLEEC), PDSparse can achieve orders of

magnitude speed up in terms of training time without significantly downgrading performance compared to the direct one-vs-all approach (except on Delicious-200k, a data set of missing labels) . 2) Given enough computing resources, DiSMEC is able to achieve significantly higher accuracy than other approaches on some data sets. However, their training on the largest data sets typically take few days even with 100 cores.

From Table 1-3, we illustrate how our proposed PPDSparse method combines the strength from both PDSparse and DiSMEC. By adopting one-versus-all loss, PPDSparse can achieve accuracy as good as DiSMEC on most of data sets and resolve drawbacks of the max-margin loss used by PDSparse in three ways: (i) PPDSparse reduces the memory requirement of PDSparse by orders of magnitude due to the separation of training of each class, which clears the MLE issue of PDSparse on Amazon-670K, (ii) By embarassingly parallelized to 100 cores, PPDSparse is orders of magnitude faster than both PDSparse and parallel 1-vs-all (DiSMEC), (iii) the performance of PPDSparse is less sensitive to data set of mislabeling. On Delicious-200K, a data set of missing positive labels, PPDsparse improves accuracy of PDSparse significantly.

The training of PPDSparse is consistently faster than tree-based and local embedding methods by orders of magnitude while maintaining a competitive accuracy on most of data sets. On Amazon-670K and WikiLSHTC-325K, PPDSparse (and DisMEC) enjoy a significant increase in accuracy compared to tree-based approaches. On the other hand, the prediction speed of Primal Dual sparse approaches (PPDSparse, PDSparse) are slower than tree-based methods (FastXML, PfastreXML) on problems of more than 10^5 classes, while being comparable on medium-sized data sets of $10^3 - 10^4$ classes (Table 3), presumably because tree-based methods enjoy logarithmic-time prediction w.r.t. the number of classes.

7 ACKNOWLEDGEMENT

I.D. is supported by NSF via CCF-1320746, IIS-1546452, and CCF-1564000. W.D. is supported by ONR via N000141410684, and NSF via CCF-1629559. P.R. is supported by ARO via W911NF-12-1-0390 and NSF via IIS-1149803, IIS-1320894, IIS-1447574, and DMS-1264033, and NIH via R01 GM117594-01 as part of the Joint DMS/NIGMS Initiative to Support Research at the Interface of the Biological and Mathematical Sciences.

REFERENCES

[1] R. Babbar and B. Schölkopf. 2017. DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017)*.

[2] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*. 730–738.

[3] Koby C. and Yoram S. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research* (2002).

[4] Jie Chen and others. 2016. Efficient one-vs-one kernel ridge regression for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2454–2458.

[5] Y.N. Chen and H.T. Lin. 2012. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*.

[6] Anna Choromanska, Alekh Agarwal, and John Langford. 2013. Extreme multi class classification. In *NIPS Workshop: eXtreme Classification, submitted*.

[7] Anna E Choromanska and John Langford. 2015. Logarithmic time online multiclass prediction. In *Advances in Neural Information Processing Systems*. 55–63.

[8] K. Crammer and Y. Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR* (2001).

[9] Koby Crammer and Yoram Singer. 2003. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research* 3, Feb (2003), 1025–1058.

[10] Wei Dai, Abhimanu Kumar, Jinliang Wei, Qirong Ho, Garth Gibson, and Eric P. Xing. 2015. Analysis of High-Performance Distributed ML at Scale through Parameter Server Consistency Models. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

[11] J. Deng, A.C. Berg, K. Li, and Li F-F. What does classifying more than 10,000 image categories tell us? In *ECCV 2010*.

[12] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research* 9 (2008), 1871–1874.

[13] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. 2008. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of 25th international conference on Machine learning*.

[14] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

[15] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and E Hüllermeier. 2016. Extreme f-measure maximization using sparse probability estimates. In *Proceedings of the 33nd International Conference on Machine Learning*. 1435–1444.

[16] Sham Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2009. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. *Unpublished Manuscript, http://ttic. uchicago. edu/shai/papers/KakadeShalevTewari09. pdf* (2009).

[17] A. Kapoor, R. Viswanathan, and P. Jain. Multilabel classification using bayesian compressed sensing. In *NIPS 2012*.

[18] S Sathiya Keerthi and others. 2008. A sequential dual method for large scale multi-class linear SVMs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 408–416.

[19] Jin Kyu Kim and others. 2016. STRADS: a distributed framework for scheduled model parallel machine learning. In *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, 5.

[20] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. 2013. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *ICML 2013 International Conference on Machine Learning*. 53–61.

[21] P. Langley. 2000. Crafting Papers on Machine Learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, Pat Langley (Ed.). Morgan Kaufmann, Stanford, CA, 1207–1216.

[22] O. Meshi, M. Mahdavi, and A. Schwing. Smooth and Strong: MAP Inference with Linear Convergence. In *NIPS 2015*.

[23] Ioannis Partalas and others. 2015. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581* (2015).

[24] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.

[25] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In *Advances in neural information processing systems*. 1177–1184.

[26] P. Richtárik and M. Takáč. 2014. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming* (2014).

[27] Shai Shalev-Shwartz and Tong Zhang. 2013. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research* (2013).

[28] Ryan J Tibshirani and others. 2013. The lasso problem and uniqueness. *Electronic Journal of Statistics* 7 (2013), 1456–1490.

[29] Jinliang Wei, Wei Dai, Aurick Qiao, Qirong Ho, Henggang Cui, Gregory R Ganger, Phillip B Gibbons, Garth A Gibson, and Eric P Xing. 2015. Managed communication and consistency for fast data-parallel iterative analytics. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*. ACM, 381–394.

[30] Jason Weston, Ameesh Makadia, and Hector Yee. 2013. Label Partitioning For Sublinear Ranking.. In *ICML (2)*. 181–189.

[31] Lingfei Wu and Andreas Stathopoulos. 2015. A preconditioned hybrid svd method for accurately computing singular triplets of large matrices. *SIAM Journal on Scientific Computing* 37, 5 (2015), S365–S388.

[32] Lingfei Wu, Ian EH Yen, Jie Chen, and Rui Yan. 2016. Revisiting random binning features: Fast convergence and strong parallelizability. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[33] Chang Xu, Dacheng Tao, and Chao Xu. 2016. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA August*. 13–17.

[34] I.E.H. Yen, X. Huang, K. Zhong, P. Ravikumar, and I.S Dhillon. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *Proceedings of the 33nd International Conference on Machine Learning*.

[35] I.E.H. Yen, T.W. Lin, S.D. Lin, P. Ravikumar, and I.S. Dhillon. Sparse random feature algorithm as coordinate descent in Hilbert space. In *NIPS 2014*.

[36] H.F. Yu, P. Jain, P. Kar, and I. S Dhillon. 2013. Large-scale multi-label learning with missing labels. *arXiv:1307.5101* (2013).

[37] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B* (2005).