PPDSparse: A Parallel Primal and Dual Sparse Method to Extreme Classification

Ian E.H. Yen[1], Xiangru Huang[2], Wei Dai[1], Pradeep Ravikumar[1], Inderjit S. Dhillon[2] and Eric Xing[1]

[1]Carnegie Mellon University. [2]University of Texas at Austin

KDD, 2017

Outline

Outline

Extreme Classification

- **Goal:** Learn a function $\boldsymbol{h}(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^K$ from D input features to K output scores that is consistent with labels $\boldsymbol{y} \in \{0, 1\}^K$.

Extreme Classification

- **Goal:** Learn a function $\boldsymbol{h}(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^K$ from D input features to K output scores that is consistent with labels $\boldsymbol{y} \in \{0, 1\}^K$.
- K is large (e.g. 10^3~10^6).

Extreme Classification

- **Goal:** Learn a function $\boldsymbol{h}(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}^K$ from D input features to K output scores that is consistent with labels $\boldsymbol{y} \in \{0, 1\}^K$.
- K is large (e.g. 10^3~10^6).
- Average number of positive labels (per sample)
  $k_p = \frac{1}{N} \sum_{i=1}^{N} (\sum_k y_{ik})$.
  Multiclass: $k_p = 1$; Multilabel: $k_p \ll K$.
- Average number of positive samples (per class)
  $n_p = \frac{1}{K} \sum_{k=1}^{K} n_p^k = \frac{1}{K} \sum_{k=1}^{K} (\sum_i y_{ik})$.
- $n_p = N k_p / K \ll N$

Extreme Classification

- We consider Linear Classifier:

$$\boldsymbol{h}(\boldsymbol{x}) := W^T \boldsymbol{x} \quad \textit{where} \quad W \in \mathbb{R}^{D \times K}.$$

Extreme Classification

- We consider Linear Classifier:

$$\boldsymbol{h}(\boldsymbol{x}) := W^T \boldsymbol{x} \quad \textit{where} \quad W \in \mathbb{R}^{D \times K}.$$

- **Challenge:** When K is large, training of simple linear model requires $O(NDK)$ cost.

Wikipedia Dataset: $N \approx 10^6$, $D \approx 10^6$, $K \approx 10^6$

Outline

Approaches

- **Approach 1** Structural, i.e. Low-rank or Tree-hierarchy Good accuracy when assumption holds. Lower accuracy when assumptions not hold.

Approaches

- **Approach 1** Structural, i.e. Low-rank or Tree-hierarchy Good accuracy when assumption holds. Lower accuracy when assumptions not hold.
- **Approach 2** Parallelized one-vs-all Good accuracy, slow, parallelizable. Need days on largest dataset with 100 cores.

Approaches

- **Approach 1** Structural, i.e. Low-rank or Tree-hierarchy Good accuracy when assumption holds. Lower accuracy when assumptions not hold.
- **Approach 2** Parallelized one-vs-all Good accuracy, slow, parallelizable. Need days on largest dataset with 100 cores.
- **Approach 3** Primal-Dual Sparse Good accuracy, fast, not parallelizable, memory issue $O(DK)$. Need days on largest dataset.

Approaches

- **Approach 1** Structural, i.e. Low-rank or Tree-hierarchy Good accuracy when assumption holds. Lower accuracy when assumptions not hold.
- **Approach 2** Parallelized one-vs-all Good accuracy, slow, parallelizable. Need days on largest dataset with 100 cores.
- **Approach 3** Primal-Dual Sparse Good accuracy, fast, not parallelizable, memory issue $O(DK)$. Need days on largest dataset.
- **This paper** Parallel PD-Sparse Good accuracy, fast, parallelizable. Need only $<$ 30 min on largest dataset with 100 cores.

Outline

- We consider the *classwise-separable hinge loss*

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

- Minimizing a separable loss is equivalent to One-versus-all:

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

- We consider the *classwise-separable hinge loss*

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

- Minimizing a separable loss is equivalent to One-versus-all:

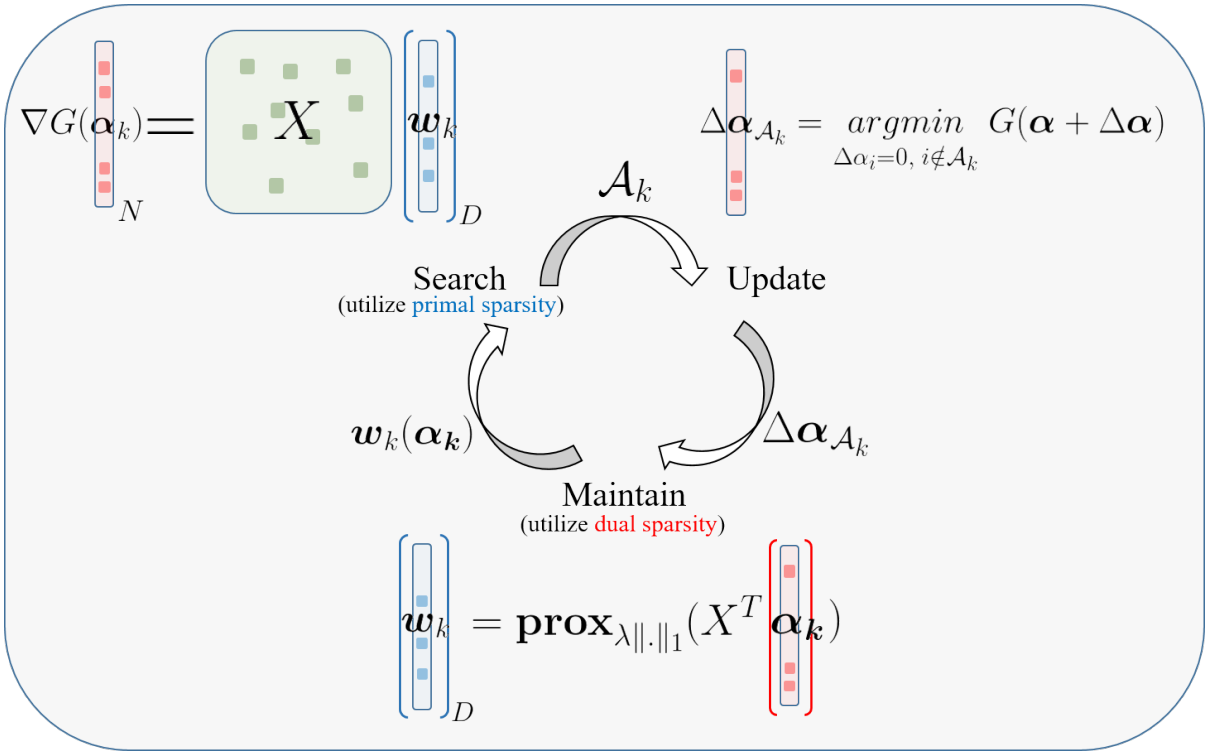
$$\min_{W \in \mathbb{R}^{D \times K}} \sum_{i=1}^{N} \sum_{k=1}^{K} \ell(\boldsymbol{w}_k^T x_i, y_{ik}) = \sum_{k=1}^{K} \left(\sum_{i=1}^{N} \ell(\boldsymbol{w}_k^T x_i, y_{ik}) \right)$$

- To obtain sparse iterates, we add ℓ_1-penalty on W and add bias per class w_{0k}. The dual problem of the ℓ_1-ℓ_2-regularized problem is:

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

Outline

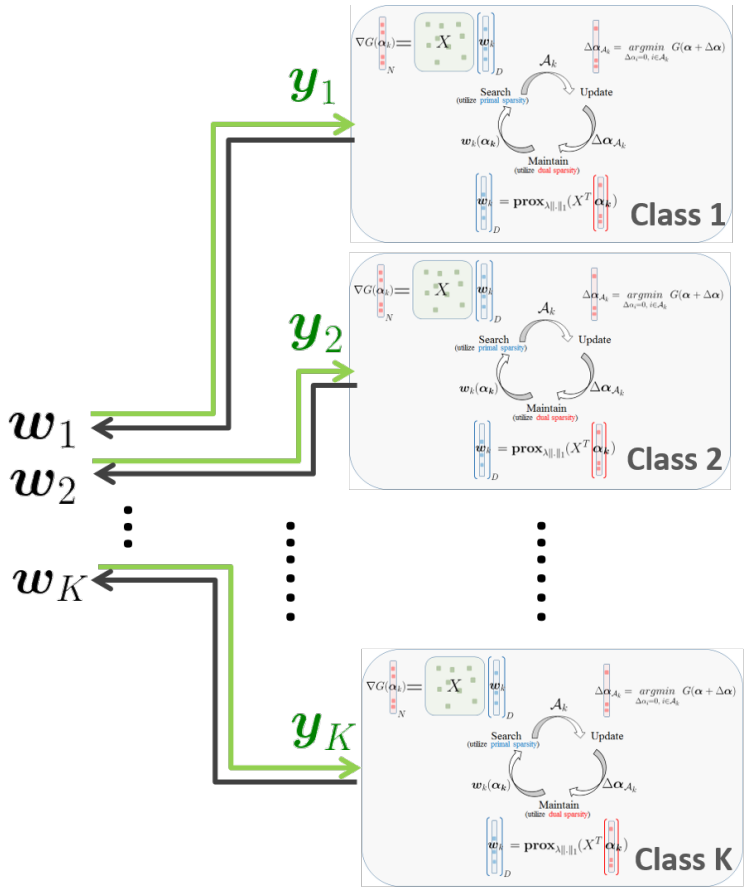
Primal-Dual-Sparse Active-set Method



- $O\left(\underbrace{nnz(\boldsymbol{w}_k)nnz(\boldsymbol{x}^j)}_{Search} + \underbrace{nnz(\boldsymbol{\alpha}_k)nnz(\boldsymbol{x}_i)}_{Update+Maintain}\right)$ per iteration.
- Apply Random Sparsification on (already sparse) $\boldsymbol{w}_k$ before search.
- Update $\boldsymbol{\alpha}$ by Coordinate Descent within $\mathcal{A}_k$.

Parallel & Primal-Dual Sparse Method

- Due to the separable loss, the optimization can be embarrassingly parallelized with one-time communication.
- The input $\boldsymbol{y}_k$ and **output** $\boldsymbol{w}_k$ of each sub-problem are sparse.
- Can be implemented in a distributed, shared-memory, or two-level parallelization setting.
- Space: $O(nnz(X) + D)$.
- Nearly linear speedup even with thousands of cores.



Outline

Theory: Primal and Dual Sparsity

- **Key Insight:** The number of positive samples for each class

$$n_p = \frac{Nk_p}{K}$$

is small. The following results hold if class-wise bias w_{k0} are added.

Theory: Primal and Dual Sparsity

- **Key Insight:** The number of positive samples for each class

$$n_p = \frac{Nk_p}{K}$$

  is small. The following results hold if class-wise bias w_{k0} are added.
- **Step-1:** bound $\|\boldsymbol{w}\|_1$, and optimal $\|\boldsymbol{\alpha}^*\|_1$ in terms of $\mathbf{n_p}$:

$$\|\boldsymbol{w}_k\|_1 \leq \frac{2n_p^k}{\lambda} \quad , \qquad \|\boldsymbol{\alpha}_k^*\|_1 \leq 4n_p^k.$$

Theory: Primal and Dual Sparsity

- **Key Insight:** The number of positive samples for each class

$$n_p = \frac{Nk_p}{K}$$

  is small. The following results hold if class-wise bias w_{k0} are added.

- **Step-1:** bound $\|\boldsymbol{w}\|_1$, and optimal $\|\boldsymbol{\alpha}^*\|_1$ in terms of $\mathbf{n_p}$:

$$\|\boldsymbol{w}_k\|_1 \leq \frac{2n_p^k}{\lambda} \quad , \qquad \|\boldsymbol{\alpha}_k^*\|_1 \leq 4n_p^k.$$

- **Step-2:** bound $nnz(\boldsymbol{w})$, and $nnz(\boldsymbol{\alpha})$ in terms of $\|\boldsymbol{w}\|_1$ and $\|\boldsymbol{\alpha}^*\|_1$:

$$nnz(\tilde{\boldsymbol{w}}_k) \leq \frac{\|\boldsymbol{w}_k\|_1^2}{\delta^2} \quad , \qquad nnz(\boldsymbol{\alpha}_k^t) \leq t \leq \frac{4\|\boldsymbol{\alpha}_k^*\|_1^2}{\epsilon}$$

  where $\tilde{\boldsymbol{w}}$ is Random-Sparsified version of $\boldsymbol{w}$ with δ-approximation error in $\nabla G(\boldsymbol{\alpha})$, and ϵ is the desired precision of solution.

Multilabel Classification

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

Multiclass Classification

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							

Thank you

Xiangru Huang
xrhuang@cs.utexas.edu