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

Ian E.H. Yen\textsuperscript{1}, Xiangru Huang\textsuperscript{2}, Wei Dai\textsuperscript{1}, Pradeep Ravikumar\textsuperscript{1}, Inderjit S. Dhillon\textsuperscript{2} and Eric Xing\textsuperscript{1}

\textsuperscript{1}Carnegie Mellon University. \textsuperscript{2}University of Texas at Austin

KDD, 2017
Outline

1. Problem Setting
   - Extreme Classification
   - Related Works

2. Algorithm
   - Separable Loss
   - Algorithm Diagram

3. Theory
   - Analysis of primal and dual sparsity

4. Experimental Results
Outline

1. Problem Setting
   - Extreme Classification
   - Related Works

2. Algorithm
   - Separable Loss
   - Algorithm Diagram

3. Theory
   - Analysis of primal and dual sparsity

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

$K$ is large (e.g. $10^3 \sim 10^6$).

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) $n_p = \frac{1}{K} \sum_{k=1}^{K} n_{kp} = \frac{1}{K} \sum_{k=1}^{K} (\sum_i y_{ik})$.

$n_p = \frac{N_{kp}}{K} \ll N$. 

Ian E.H. Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit S. Dhillon and Eric Xing (shortinst)
Goal: Learn a function \( h(x) : \mathbb{R}^D \rightarrow \mathbb{R}^K \) from \( D \) input features to \( K \) output scores that is consistent with labels \( y \in \{0, 1\}^K \).

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

$K$ is large (e.g. $10^3 \sim 10^6$).

Average number of positive labels (per sample)

$$k_p = \frac{1}{N} \sum_{i=1}^{N} \left( \sum_k y_{ik} \right).$$

Multiclass: $k_p = 1$; Multilabel: $k_p \ll K$.

Average number of positive samples (per class)

$$n_p = \frac{1}{K} \sum_{k=1}^{K} n_p^k = \frac{1}{K} \sum_{k=1}^{K} \left( \sum_i y_{ik} \right).$$

$n_p = Nk_p/K \ll N$
We consider **Linear Classifier**:

\[ h(x) := W^T x \quad \text{where} \quad W \in \mathbb{R}^{D \times K}. \]
We consider Linear Classifier:

\[ h(x) := W^T x \text{ where } W \in \mathbb{R}^{D \times K}. \]

Challenge: When \( K \) is large, training of simple linear model requires \( O(NDK) \) cost.
Outline

1 Problem Setting
   - Extreme Classification
   - Related Works

2 Algorithm
   - Separable Loss
   - Algorithm Diagram

3 Theory
   - Analysis of primal and dual sparsity

4 Experimental Results
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 $< 30 \text{ min}$ on largest dataset with 100 cores.
Outline

1. Problem Setting
   - Extreme Classification
   - Related Works

2. Algorithm
   - Separable Loss
   - Algorithm Diagram

3. Theory
   - Analysis of primal and dual sparsity

4. Experimental Results
We consider the *classwise-separable hinge loss*

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

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

\[
\min_{W \in \mathbb{R}^{D \times K}} \sum_{i=1}^{N} \sum_{k=1}^{K} \ell(w_k^T x_i, y_{ik}) = \sum_{k=1}^{K} \left( \sum_{i=1}^{N} \ell(w_k^T x_i, y_{ik}) \right)
\]
We consider the classwise-separable hinge loss

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

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

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

To obtain sparse iterates, we add \( \ell_1 \)-penalty on \( W \) and add bias per class \( w_{0k} \). The dual problem of the \( \ell_1-\ell_2 \)-regularized problem is:

\[
\min_{\alpha_k \in \mathbb{R}^N} G(\alpha_k) := \frac{1}{2} \|w(\alpha_k)\|^2 - \sum_{i=1}^{N} \alpha_{ik}
\]

s.t. \( w(\alpha_k) = \text{prox}_{\lambda}(\hat{X}^T \alpha_k), \)

\( 0 \leq \alpha_{ik} \leq 1. \)
Outline

1. Problem Setting
   - Extreme Classification
   - Related Works

2. Algorithm
   - Separable Loss
   - Algorithm Diagram

3. Theory
   - Analysis of primal and dual sparsity

4. Experimental Results
Primal-Dual-Sparse Active-set Method

\[ \nabla G(\alpha_k) = Xw_k \]

\[ \Delta \alpha_{A_k} = \arg\min_{\Delta \alpha_i = 0, \ i \notin A_k} G(\alpha + \Delta \alpha) \]

\[ w_k = \text{prox}_{\lambda \| \cdot \|_1}(X^T(\alpha_k)) \]

- \( O\left( \text{nnz}(w_k) \text{nnz}(x^j) + \text{nnz}(\alpha_k) \text{nnz}(x_i) \right) \) per iteration.
  - **Search** \( \text{nnz}(w_k) \text{nnz}(x^j) \)
  - **Update + Maintain** \( \text{nnz}(\alpha_k) \text{nnz}(x_i) \)

- Apply **Random Sparsification** on (already sparse) \( w_k \) before search.
- Update \( \alpha \) by Coordinate Descent within \( A_k \).
Due to the separable loss, the optimization can be embarrassingly parallelized with one-time communication.

The input $y_k$ and output $w_k$ 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.
1 Problem Setting
   - Extreme Classification
   - Related Works

2 Algorithm
   - Separable Loss
   - Algorithm Diagram

3 Theory
   - Analysis of primal and dual sparsity

4 Experimental Results
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.
Key Insight: The number of positive samples for each class

\[ n_p = \frac{NK}{K} \]

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

Step-1: bound \( \|w\|_1 \), and optimal \( \|\alpha^*\|_1 \) in terms of \( n_p \):

\[ \|w_k\|_1 \leq \frac{2n^k_p}{\lambda}, \quad \|\alpha^*_k\|_1 \leq 4n^k_p. \]
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 \( \| w \|_1 \), and optimal \( \| \alpha^* \|_1 \) in terms of \( n_p \):
  \[ \| w_k \|_1 \leq \frac{2n_p^k}{\lambda} , \quad \| \alpha^*_k \|_1 \leq 4n_p^k. \]

- **Step-2:** bound \( \text{nnz}(w) \), and \( \text{nnz}(\alpha) \) in terms of \( \| w \|_1 \) and \( \| \alpha^* \|_1 \):
  \[ \text{nnz}(\tilde{w}_k) \leq \frac{\| w_k \|_1^2}{\delta^2} , \quad \text{nnz}(\alpha^*_k) \leq t \leq \frac{4\| \alpha^*_k \|_1^2}{\epsilon} \]
  where \( \tilde{w} \) is Random-Sparsified version of \( w \) with \( \delta \)-approximation error in \( \nabla G(\alpha) \), and \( \epsilon \) is the desired precision of solution.
**Multilabel Classification**

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

### Table: Comparison of Different Methods

<table>
<thead>
<tr>
<th>Data</th>
<th>Metrics</th>
<th>FastXML</th>
<th>PfastreXML</th>
<th>SLEEC</th>
<th>PDSparse</th>
<th>DiSMEC</th>
<th>PPDSparse</th>
</tr>
</thead>
<tbody>
<tr>
<td>aloi.bin</td>
<td>$T_{train}$</td>
<td>1900.9s</td>
<td>1901.6s</td>
<td>16193s</td>
<td>139.8s</td>
<td>92.0s</td>
<td>7.05s</td>
</tr>
<tr>
<td></td>
<td>accuracy (%)</td>
<td>95.71</td>
<td>93.43</td>
<td>93.74</td>
<td>96.2</td>
<td>96.28</td>
<td>96.38</td>
</tr>
<tr>
<td></td>
<td>model size</td>
<td>1.3G</td>
<td>1.3G</td>
<td>3.7G</td>
<td>19M</td>
<td>16M</td>
<td>14M</td>
</tr>
<tr>
<td></td>
<td>$T_{test}/N_{test}$</td>
<td>5.05ms</td>
<td>5.10ms</td>
<td>28.00ms</td>
<td>0.064ms</td>
<td>0.02ms</td>
<td>0.0178ms</td>
</tr>
<tr>
<td>LSHTC1</td>
<td>$T_{train}$</td>
<td>1398.2s</td>
<td>1422.4s</td>
<td>5919.3s</td>
<td>196.6s</td>
<td>298.8s</td>
<td>45.8s</td>
</tr>
<tr>
<td></td>
<td>accuracy (%)</td>
<td>22.04</td>
<td>23.32</td>
<td>12.2</td>
<td>22.46</td>
<td>22.74</td>
<td>22.70</td>
</tr>
<tr>
<td></td>
<td>model size</td>
<td>937M</td>
<td>1.1G</td>
<td>631M</td>
<td>88M</td>
<td>142M</td>
<td>381M</td>
</tr>
<tr>
<td></td>
<td>$T_{test}/N_{test}$</td>
<td>5.73ms</td>
<td>8.81ms</td>
<td>14.66ms</td>
<td>0.40ms</td>
<td>3.7ms</td>
<td>6.94ms</td>
</tr>
<tr>
<td>Dmoz</td>
<td>$T_{train}$</td>
<td>6475.1s</td>
<td>6619.7s</td>
<td>47490s</td>
<td>2518.9s</td>
<td>1972.0s</td>
<td>170.60s</td>
</tr>
<tr>
<td></td>
<td>accuracy (%)</td>
<td>40.76</td>
<td>39.78</td>
<td>33.03</td>
<td>39.91</td>
<td>39.38</td>
<td>39.32</td>
</tr>
<tr>
<td></td>
<td>model size</td>
<td>3.5G</td>
<td>3.8G</td>
<td>1.5G</td>
<td>680M</td>
<td>369M</td>
<td>790M</td>
</tr>
<tr>
<td></td>
<td>$T_{test}/N_{test}$</td>
<td>3.29ms</td>
<td>3.20ms</td>
<td>40.43ms</td>
<td>1.87ms</td>
<td>4.58ms</td>
<td>6.58ms</td>
</tr>
</tbody>
</table>
Thank you

Xiangru Huang
xrhuang@cs.utexas.edu