CS771 Assignment 2, Group 21

Abhyuday Pandey	Ayush Kumar	Deepesh Lall	Srinjay Kumar	Umang Malik
170039	170195	170236	170722	170765

1 Description of PD Sparse[1]

Given feature vectors $(\mathbf{x}_i)_{i=1}^N$ and label indicator variables $\mathbf{y}_i$. Denote $\mathcal{P}_i = \{k \in [K] \mid y_{ik} = 1\}$ and $\mathcal{N}_i = \{k \in [K] \mid y_{ik} = 0\}$.
Denote $X := (\mathbf{x}_i^T)_{i=1}^N$ as the $N \times D$ matrix and $Y := (\mathbf{y}_i^T)_{i=1}^N$ as the $N \times K$ label matrix, the aim is to learn a classifier $h : \mathbb{R}^D \to [K]$

$$h(\mathbf{x}) := \arg\max_k \langle \mathbf{w}_k, \mathbf{x} \rangle \tag{1}$$

parametrized by a $D \times K$ matrix, i.e. $W = (\mathbf{w}_k)_{k=1}^K$. Consider, the following minimization problem (l_2 regularized),

$$\frac{1}{2}\sum_{k=1}^{K} ||\mathbf{w}_k||_2^2 + C\sum_{i=1}^{N} L(W^T\mathbf{x}_i, y_i) \tag{2}$$

Consider the following problem with l_1 penalty,

$$\lambda\sum_{k=1}^{K} ||\mathbf{w}_k||_1 + C\sum_{i=1}^{N} L(W^T\mathbf{x}_i, \mathbf{y}_i) \tag{3}$$

(3) has superior sparsity, but it is non-smooth and non-separable. Moreover, greedy coordinate-wise optimization would be non-convergent. A mix of (2) and (3) can solve the problem. Consider the minimization problem,

$$\frac{1}{2}\sum_{k=1}^{K} ||\mathbf{w}_k||_2^2 + \lambda\sum_{k=1}^{K} ||\mathbf{w}_k||_1 + C\sum_{i=1}^{N} L(W^T\mathbf{x}_i, \mathbf{y}_i) \tag{4}$$

The dual representation of the same problem is,

$$G(\alpha) = \min_\alpha \frac{1}{2}\sum_{k=1}^{K} ||\mathbf{w}_k(\alpha_k)||^2 + \sum_{i=1}^{N} \mathbf{e}_i^T\alpha^i \; s.t. \; \alpha^{\mathbf{i}} \in \mathcal{C}_{\mathbf{i}}, \forall i \in [N] \tag{5}$$

$$\nabla_{\alpha^i} G(\alpha) = W^T\mathbf{x}_i - \mathbf{e}_i \tag{6}$$

$$k_n^* = \arg\max_{k_n \in \mathcal{N}_i} \langle \mathbf{w}_k^t, \mathbf{x}_i \rangle - 1 \tag{7}$$

The research aims at optimizing (4) by taking λ and C as hyper-parameters. The algorithm for performing descent is Fully-Corrective Block-Coordinate Frank Wolfe (FC BCFW) for the dual problem. The complexity is sublinear and works for $10^3 - 10^6$ classes and performs reasonably better than other approaches.

In BCFW strategy, only the active set of variables is updated at a time. The pseudocode is summarized below,

Time Complexity
The gradient can be calculated in $O(nnz(x_i)nnz(w_j))$. Updating a coordinate of a point itself requires $O(nnz(x_i)K)$ time, which is significantly higher than evaluating the gradient.

Preprint. Under review.

Algorithm 1 FC BCFW

Initialize $\alpha^0 \leftarrow 0$
Initialize $A^0 \leftarrow \phi$
for $i \in [1..T]$ **do**
    Randomly select i from 1 ... N
    Find most violating label $k^* \in N_i$ ▷ Using (7)
    $A_i^{t+\frac{1}{2}} = A_i^t \bigcup \{k^*\}$
    Solving subproblem w.r.t. active set $A_i^{t+\frac{1}{2}}$
    $A_i^{t+\frac{1}{2}} = A_i^{t+\frac{1}{2}} \setminus \{k | \alpha_{ik} = 0, k \notin P_i\}$
end for

2 Advantages and Disadvantages of PDSparse

Advantages-

- Unlike OvA linear classifiers other than itself, PDSparse provides competitive prediction and training time compared to tree based approaches.
- Standard multiclass solvers suffer from complexity growing linearly with K unlike PDSparse which has sublinear complexity.
- Model size is 5x to 10x smaller than FastXML (tree based),VW (tree based), LEML (low-rank based) and SLEEC (piecewise low-rank based).

Disadvantages-

- The problem (4) does not satisfy Primal and Dual sparsity ($nnz(W^*) \leq nnz(A^*)$) of ERM for any $\lambda > 0$. However, empirically the solution obtained from (4) is close in sparsity to that from (3).
- The classifier is based on OvA linear classification, a linearly inseparable data may provide hurdles. However, this is quite less likely in sparse vectors.
- The number of classes is equal to the number of objects, which is again problematic as the model size increases with number of objects. However, in this case only 3400 objects are present.
- The method is largely dependent on the seed value as observed in our experiments. Many a times the accuracy is incompetent even for a set of seed values.

3 Results on PDSparse (without any changes)

Random seed value to replicate the experiment = 1572288642 (which is CPU time by default). λ and C also default.

$k \longrightarrow$	1	3	5
prec@k	0.8417	0.723556	0.621598
mprec@k	0.00964171	0.0146676	0.016019

4 Other methods and Modifications

We tried the following methods :

- PDSparse
- PARABEL[2] is a tree-based method which deploys linear classification at its leaf nodes. Since, the last step is a linear classification with only a few (upto 500 labels as set by us) we need not perform PD Sparse at this step, we can manage with L2 regularized L1 loss SVC. The following results were obtained. A concern in this method is its model size (4x times bulkier) and its prediction time (20x larger). This method works well when number of labels are very large (not in this case).

Modifications done:

- We tried to decrease the depth of the PARABEL tree. This comes as a tradeoff, while tree performs faster classification than the accompanying linear classifier, the latter provides lightweight models. However, we could not decrease it beyond 20 MB (which is still atleast 2x larger than PDSparse). The time was increased but not very significantly. The prediction was slightly better. We dropped out this method as the model size was too big.
- To overcome the seed value problem (discussed in disadvantages) we devised the following algorithm. The above algorithm increased the training time (by roughly 1.5x) but the resulting models were very accurate. With this modification, we beat PARABEL in terms of accuracy as well.

Algorithm 2 Start-PDSparse

```
Initialize m                                              ▷ m ≈ 3
bestSeed ← 0
Perform FC BCFW for m iters using bestSeed
acc ← accuracy on heldout data
for i ∈ [1..T] do
    Randomly take a seed value s
    Perform FC BCFW for m iters using s
    acc_i ← accuracy on heldout data
    if acc_i > acc then
        acc ← acc_i
        bestSeed ← s
    end if
end for
Run PDSparse with seed s
```

- **Hyperparameter tuning:** We tuned λ and C in equation (4). The obtained values of $\lambda = 0.08$ and $C = 6$.
- To reduce the time taken to pass arguments to the prediction algorithm, we wrapped our C++ class using Cython[3].

References

References to papers and applications we used for studying and developing our prediction code.

[1] PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. Ian E.H. Yen*, Xiangru Huang*, Kai Zhong, Pradeep Ravikumar and Inderjit S. Dhillon. (* equally contributed) In International Conference on Machine Learning (ICML), 2016. [pdf]

[2] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal and M. Varma, Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising in WWW, 2018. [pdf]

[3] [Cython]