Project Report: Implementation and Analysis of CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning

(Group No. 38)
Bidya Sarkar(18111011), *Nirjhar Roy*(18111409)
Avijit Roy(18111404), *Manish Mazumder*(18111038)
Supervised By : *Prof. Piyush Rai*

DISCLAIMER:

The work carried out in the project has not been re-used from any another course project at IITK or elsewhere, or any other project implemented elsewhere.

***Abstract*— In this project we explore the idea of Extreme Multi-Label learning (XML). In XML data sets are typically of huge dimensions (can even exceed one million). Tree based methods can divide the problem into small scale sub problems and reduce the complexity of computation at subsequent nodes. However, current tree based methods before CRAFTML does not exploit the idea of randomization in trees. We in this project use the idea of random forest to efficiently do the computation by reducing the complexity of computation at each label of tree. And as tree based methods inherently support parallelized implementation, it takes less computation time.**

I. INTRODUCTION AND PROBLEM DESCRIPTION

Multi-Label classification is a very useful practical problem which can be used to recommend product or items, used in image or text classification, bioinformatics and many more. But most of the standard algorithms face scalability issues or performance issues due to huge data dimensions and huge datasets.

There are three possible ways that have been tried to address these issues.

- Using optimization tricks. (e.g., Primal-Dual conversion, parallization in supercomputers).
- Reducing the data-dimensionality and using a low dimensional representation of the data.
- Hierarchically partitioning the data into small lower complexity problems using tree based or similar approach.

Tree based approach has several advantage, it can exploit the inherent structural property of a data (like Wikipedia article). It also reduces the data complexity. So, less computation required.

CRAFTML is an algorithm which tries to solve the problem of eXtreme Multi-label Learning. The main objective in Extreme multi-label learning is build a classifier that can annotate the the test points with appropriate labels using a subset of huge labels provided. In this project we solve the problem of handling huge dimensional data by taking a random projection of the data and then randomly subsampling the data. It uses random projection insted of random selection, which preserves more information.

CRAFTML uses a low complexity data clustering method to split the data in each node which avoids the resolution of a multi-objective optimization problem at each node. Moreover, with the use of parallel approach, the training time decreases.

II. PRIOR WORKS

Due to increase interest in multi-label classification in past decade, many interesting algorithms have been proposed. Several numerical experimentations have highlighted the performances of multi-label k-nearest neighbours(ML-kNN) and of the multi-label random forest. However, they cannot scale up dimensionality orders (10^5 to 10^7) of the eXtreme Multi-label learning. As a result, three different algorithms strategies are developed to handle the scaling issue : Optimization tricks and parallelization, Dimensionality reduction and Hierarchical decomposition.

A. *Optimization tricks and parallelization*

(PDSparse, PPDSparse, DISMEC). PDSparse resorts to a sparse linear model which is regularized with an elastic net to minimize the ranking loss measure. It exploits a primal-dual conversation of the optimization problem and the sparsity to scale up to XML data. Recently it has been extended to parallel model called PPDSparse. DISMEC is purely based on regularized one-vs-rest large margin linear model. By exploiting the independence between the submodels associated to each label, computation are accelarated with a parallelization of the training set. The meomory consumption is thus reduced which leads to a sparse model.

B. *Dimensionality Reduction*

Dimensionality reduction gives a synthesized representation of data with lower number of variables(feature, labels, or both). In XML one of the very first dimensionality reduction technique used was LEML and WSABIE. They create low rank reduced version of data but they miss information brought by the long tail distribution of the labels. To overcome this difficulty SLEEC and AnnexML train a set of local low-rank embeddings on a partition of a feature space to

cover a global high rank embedding of good quality. While in SLEEC the partition is deduced from an unsupervised clustering, in AnnexML the partition aims at gathering the same class item.

C. *Tree-based Methods*

Tree-based methods transform the large-scale problem into series of small-scale sub-problems by hierarchically partitioning the label set. These subsets are nodes of the tree. The whole set associated to the root is partitioned into a fixed number k of subsets which are associated to k child nodes. The decomposition repeats untill a stop condition is reached. In each node two types of optimization problems can be raised : (i) Computing a partition for a given criterion, and (ii) defining a condition or building a classifier on the feature space to decide to which subsets of the partition an instance is assigned. For prediction an input follows path from root to leaf or several leaves. For an instance tree, a local classifier predicts on trained leaf instances.

There are two pioneering approaches RF-PCT and HOMER presented in past decade have highlighted the interest of tree-based strategy. XML literature has recent tree-based strategy : LPSR and FastXML respectively based on a single k-ary instance tree and a forest of binary trees and PLT based on a label tree.

These approaches have obtained competitive results in numerical experiments. But there are still lot of spaces to improve. There are two approaches: using very fast partitioning strategy and exploiting and tree feature/label randomization. Moreover, tree diversity, implemented with random feature selection, contributed to success of random forests and RF-PCT in multilabel learning. To address these shortcomings, a novel tree-based approach called CRAFTML is introduced.

III. IMPLEMENTATION

1) Decision tree forest of four decision trees.
2) Each decision tree in forest differs in the below parameters:
   a) number of children of a node.
   b) feature sampled of data.
   c) number of data point sampled.
3) Stopping criteria for decision tree generation:
   a) If number of datapoint in a node < 200.
   b) All datapoint in a node have same feature.
   c) All datapoint in a node have same label.
4) Each node will execute two methods to generate the decision tree
   a) trainTree.
   b) trainNodeClassifier.
5) At each node data is randomly feature sampled (x% of features considered) followed by row sampled randomly.
6) Each node will execute Kmeans on label of datapoint and cluster the datapoints which is followed by creation of child nodes of a node.
7) Train data and validation data ratio is kept at 9:1
8) For the used data sets, label vector is multilabel binary vector.
9) Accuracy calculation is performed in form of "precision@k", where k value is taken as 1,3,5.

Algorithm 1 trainTree

Input: Training set with a feature matrix X and a label matrix Y.
Initialize node v
v.isLeaf $\leftarrow$testStopCondition(X, Y)
if v.isLeaf $=$ false **then**
  v.classif $\leftarrow$ trainNodeClassifier(X, Y)
  $(X_{child_i}, Y_{child_i})_{i=0,..,k-1} \leftarrow$ split$(v.\text{classif}, X, Y)$
  for i **from** 0 **to** $k-1$ **do**
    $v.child_i \leftarrow$trainTree$(X_{child_i}, Y_{child_i})$
  end for
else
  $v.\widehat{y} \leftarrow$computeMeanLabelVector(Y)
end if
Output: node v

Algorithm 2 trainNodeClassifier

Input: feature matrix (X_v) and label matrix (Y_v) of the instance set of the node v.
$X_s, Y_s \leftarrow$ sampleRows(X_v, Y_v, n_s)
$X'_s \leftarrow X_s P_x$ # random feature projection
$Y'_s \leftarrow Y_s P_y$ # random label projection
$c \leftarrow k$-means(Y'_s, k) # $c \in \{0, ..., k-1\}^{\min(n_v, n_s)}$
for i **from** 0 **to** $k-1$ **do**
  $(\text{classif})_{i,.} \leftarrow$ computeCentroid$(\{(X'_s)_{j,.} | c_j = i\})$
end for
Output: Classifier classif $(\in \mathbb{R}^{k \times d'_x})$.

c is a vector where the j^{th} component c_j denotes the cluster index of the j^{th} instance associated to $(X'_s)_{j,.}$ and $(Y'_s)_{j,.}$.

IV. TOOLS AND SOFTWARES USED

1) Python IDE : Jupyter notebook, Spyder.
2) Library (Python) : numpy, scipy, sklearn, matplotlib.
3) Python inbuilt methods :
   - numpy(array, isclose , argsort, intersect1d)
   - scipy.sparse
   - sklearn.cluster.KMeans
   - sklearn.random_projection. SparseRandomProjection
   - sklearn.metrics.pairwise.pairwise_distances
4) Dataset used:
   http://manikvarma.org/downloads/XC/XMLRepository.html
   a) Mediamill(#datapoint = 30993, #feature = 120, #labels = 101).
   b) Bibtex (#datapoint = 4880, #feature = 1836, #labels = 159).
   c) Delicious (#datapoint = 12920, #feature = 500, #labels = 983).
   d) wiki10 (#datapoint = 14146, #feature = 101938, #labels = 30938).

V. RESULTS AND OBSERVATIONS

We have applied different set of values for hyper parameters like number of row sampled, number of feature sampled and number of clusters on validation data set and get the accuracy of the model. We have chosen the hyper parameter values for which the validation accuracy becomes consistent . We have tested the test data set with that values and get the accuracy as mentioned in the table. We also tried to get the test data accuracy with the set of values applied for validation data accuracy.

We have run the experiments in Intel Core i5(5^{th} gen), 2.6 gHz, 8GB RAM system.

For the last dataset(wiki10) we have used part of the data to train the model using Intel Core i7(3.4 gHz),16 GB RAM system.

A. Accuracy Analysis:

Mediamill Dataset		
p@value	Actual Accuracy	TEST data achived accuracy
p@1	85.86%	81.22%
p@2	69.01%	64.84%
p@3	54.65%	50.74%

Bibtex Dataset		
p@value	Actual Accuracy	TEST data achived accuracy
p@1	65.15%	61.83%
p@3	39.83%	27.49%
p@3	28.99%	20.58%

Delicious Dataset		
p@value	Actual Accuracy	TEST data achived accuracy
p@1	70.26%	62.85%
p@2	63.98%	56.43%
p@3	59.00%	52.17%

wiki10 Dataset(31k)		
p@value	Actual Accuracy	TEST data achived accuracy
p@1	85.19%	37.26%
p@3	73.17%	26.34%
p@5	63.27%	20.62%

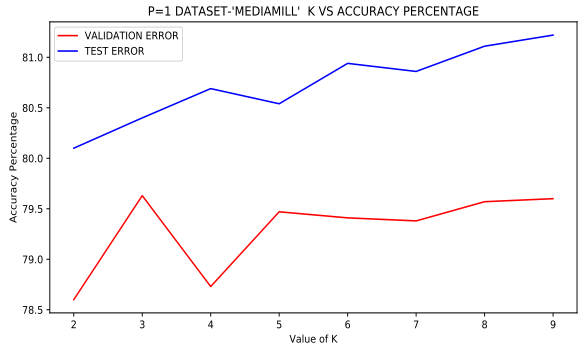
B. Time Analysis:

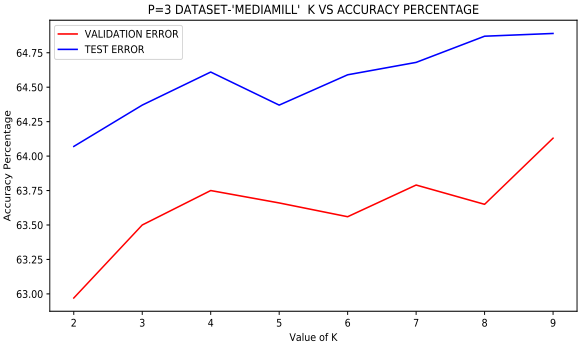
Mediamill Dataset (k=9, train : validation = 9 : 1)			
Tree Id	Time(in sec.)	Sampled Rows	Sampled Features
1	206.6995	60%	70%
2	206.6995	60%	60%
3	123.4935	60%	70%
4	124.2034	60%	60%

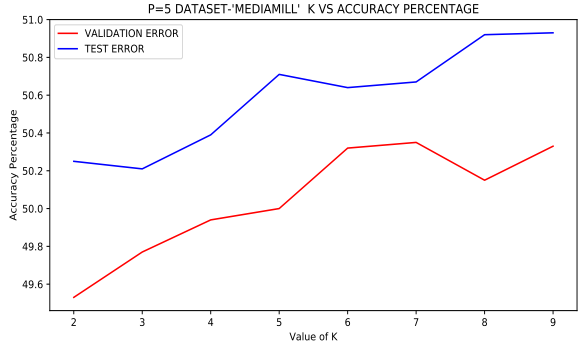
Bibtex Dataset (k=9, train : validation = 6 : 4)			
Tree Id	Time(in sec.)	Sampled Rows	Sampled Features
1	22.1493	60%	70%
2	16.6686	60%	60%
3	22.5723	60%	70%
4	22.5789	60%	60%

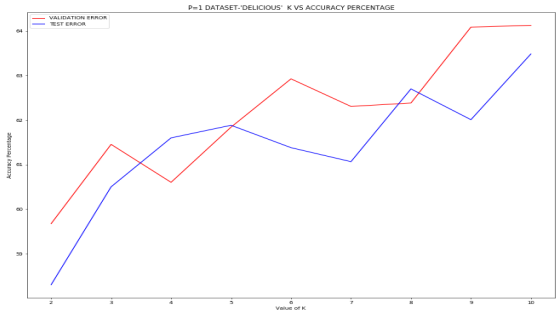
Delicious Dataset (k=9, train : validation = 9 : 1)			
Tree Id	Time(in sec.)	Sampled Rows	Sampled Features
1	93.1794	60%	70%
2	81.5424	60%	60%
3	73.7701	60%	70%
4	70.0399	60%	60%

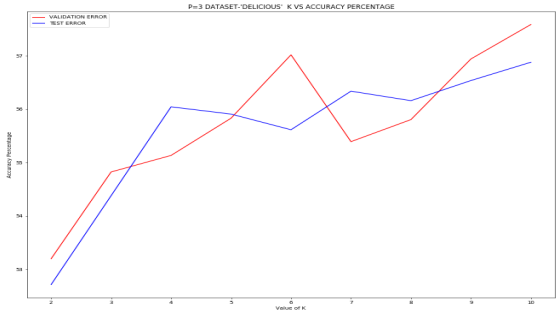
wiki10(31k) (k=9, train : validation = 9 : 1)			
Tree Id	Time(in sec.)	Sampled Rows	Sampled Features
1	8191.7580	60%	70%
2	8114.4623	60%	60%
3	8465.1402	60%	70%
4	8049.6639	60%	60%

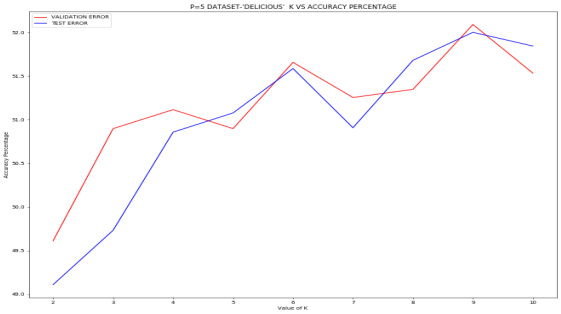


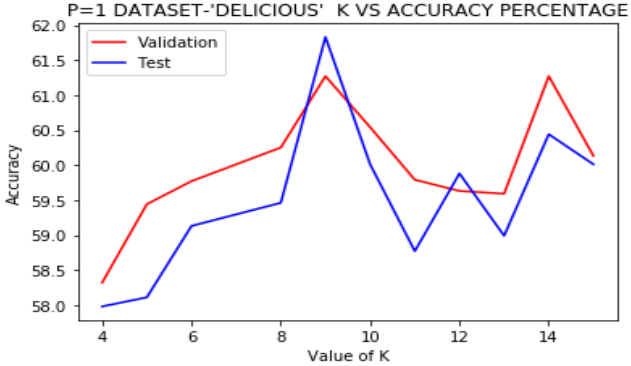


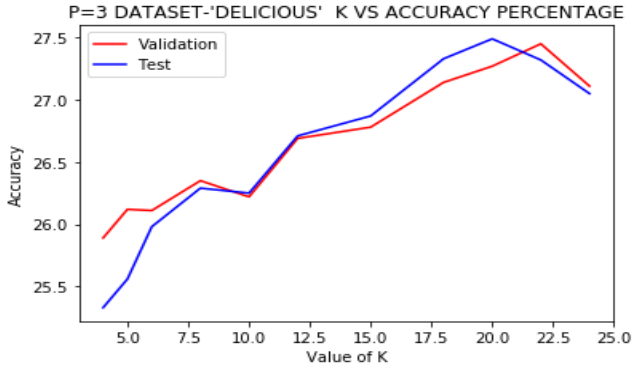


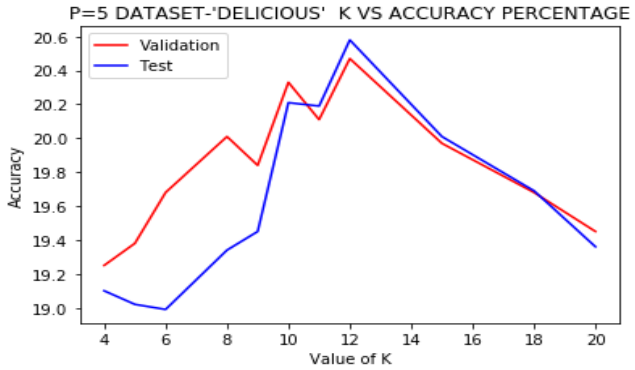












VI. CONCLUSIONS

- We have performed validation with different set of values of hyper-parameters like number of clusters, number of row sampled, number of feature sampled and get the accuracy values.
- Studied the behaviour of model with different values.
- Also perform the testing on test data and get the accuracy of the mode.
- We have seen that, with closer cluster number for all trees in decision forest keeping all parameters constant, increases the accuracy.

VII. ACKNOWLEDGMENT

We are very much thankful to Prof. Piyush Rai for his help and supervision. And we are also thankful to the Project Mentor and Teaching Assistants for their support throughout the project.

REFERENCES

[1] Class Notes provided.
[2] https://www.wikipedia.org/
[3] https://stackoverflow.com/
[4] Wissam Siblini, Pascale Kuntz , Frank Meyer . CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning
[5] Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, Manik Varma, Extreme Multi-label Learning with Label Features for Warm-start Tagging, Ranking Recommendation.

Github Repository: https://github.com/manishmazumder/CRAFTML.git