FastXML: A Fast, Accurate and Stable Tree-classifier for eXtreme Multi-label Learning

Prabhu & Varma
KDD 2014

February 5, 2019

Outline

1. Introduction
2. Training
3. Testing

Motivation

- Objective in eXtreme Multi-Label (XML) classification is to learn a classifier that can automatically tag a data point with the most relevant subset of labels from a large label set

FastXML Overview

- FastXML learns a hierarchy, not over the label space as is traditionally done in the multi-class setting, but rather over the feature space
- The intuition is that only a small number of labels are present, or active, in each region of feature space.
- Efficient prediction can be made by determining the region in which a test point lies by traversing the learnt feature space hierarchy and then focusing exclusively on the set of labels active in the region

FastXML Overview

- FastXML learns an ensemble of trees
- FastXML defines the set of labels active in a region to be the union of the labels of all training points present in that region
- Predictions are made by returning the ranked list of most frequently occurring active labels in all the leaf nodes in the ensemble containing the test point

Outline

Learning to Partition a Node

- Training FastXML consists of recursively partitioning a parents feature space between its children
- Such node partitions should be learnt by optimizing a global measure of performance such as the ranking predictions induced by the leaf nodes

Learning to Partition a Node

- Data $\{(x_i, y_i)_{i=1}^{N}\}$ with D dimensional feature vectors x_i and L dimensional binary label vectors $y_i \in 0, 1^L$
- Discounted Cumulative Gain (DCG) at k of a ranked vector r given ground truth label vector y with binary levels of relevance:

$$\mathcal{L}_{DCG@k}(r, y) = \sum_{l=1}^{k} \frac{y_{rl}}{log(1 + l)} \tag{1}$$

- Unlike precision, DCG is sensitive to both the ranking and relevance of predictions.

Learning to Partition a Node

FastXML partitions the current node's feature space by learning a linear separator w:

$$
\begin{aligned}
\min \quad & \|\mathbf{w}\|_1 + \sum_i C_\delta(\delta_i) \log(1 + e^{-\delta_i \mathbf{w}^\top \mathbf{x}_i}) \\
& - C_r \sum_i \tfrac{1}{2}(1 + \delta_i) \mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^+, \mathbf{y}_i) \\
& - C_r \sum_i \tfrac{1}{2}(1 - \delta_i) \mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^-, \mathbf{y}_i) \\
\text{w.r.t.} \quad & \mathbf{w} \in \mathcal{R}^D, \boldsymbol{\delta} \in \{-1, +1\}^L, \mathbf{r}^+, \mathbf{r}^- \in \Pi(1, L)
\end{aligned}
$$

i indexes the training points present at the node being partitioned,
$\delta_i \in \{-1, +1\}$ indicates whether point i was assigned to the negative or positive partition,
and r^+ and r^- represent the predicted label rankings for the positive and negative partition respectively.

Learning to Partition a Node

- $DCG@L$ is performed on each node, even though the ultimate leaf node rankings will be evluated at $k << L$
- The separator function allows a label to be assigned to both partitions if 2 separate points containing the same label are split into the diff. feature space. This makes FastXML robust.

Learning to Partition a Node

Algorithm 1 FastXML($\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, T$)

```
parallel-for i = 1,..,T do
    n^root ← new node
    n^root.Id ← {1,..,N}                    # Root contains all instances
    GROW-NODE-RECURSIVE(n^root)
    T_i ← new tree
    T_i.root ← n^root
end parallel-for
return T_1,..,T_T

procedure GROW-NODE-RECURSIVE(n)
    if |n.Id| ≤ MaxLeaf then                 # Make n a leaf
        n.P ← PROCESS-LEAF({x_i, y_i}_{i=1}^N, n)
    else          # Split node and grow child nodes recursively
        {n.w, n.left_child, n.right_child}
                          ← SPLIT-NODE({x_i, y_i}_{i=1}^N, n)
        GROW-NODE-RECURSIVE(n.left_child)
        GROW-NODE-RECURSIVE(n.right_child)
    end if
end procedure

procedure PROCESS-LEAF({x_i, y_i}_{i=1}^N, n)
    P ← top-k( Σ_{i∈n.Id} y_i / |n.Id| )
return P                         # Return scores for top k labels
end procedure
```

$$\mathbf{P} \leftarrow \text{top-k}\left(\frac{\sum_{i \in n.Id} \mathbf{y}_i}{|n.Id|}\right)$$

Training FastXML

- Start by setting $w = 0$ and δ_i to be 1 or $+1$ uniformly at random.
- Each iteration, then, consists of taking three steps.
  1. $r+$ and r are optimized while keeping w and δ fixed. This determines the ranked list of labels that will be predicted by the positive and negative partitions respectively
  2. δ is optimized while keeping w and $r\pm$ fixed. his step assigns training points in the node to the positive or negative partition.
  3. Optimizing w while keeping δ and $r\pm$ fixed is taken only if the first two steps did not lead to a decrease in the objective function.

Outline

1. Introduction
2. Training
3. Testing

Prediction

Algorithm 3 PREDICT($\{\mathcal{T}_1, ..\mathcal{T}_T\}, \mathbf{x}$)

for $i = 1, .., T$ **do**
  $n \leftarrow \mathcal{T}_i$.root
  while n is not a leaf **do**
    $\mathbf{w} \leftarrow n.\mathbf{w}$
    if $\mathbf{w}^\top \mathbf{x} > 0$ **then**
      $n \leftarrow n$.left_child
    else
      $n \leftarrow n$.right_child
    end if
  end while
  $\mathbf{P}_i^{\text{leaf}}(\mathbf{x}) \leftarrow n.\mathbf{P}$
end for
$\mathbf{r}(\mathbf{x}) = \text{rank}_k \left(\frac{1}{T} \sum_{i=1}^{T} \mathbf{P}_i^{\text{leaf}}(\mathbf{x}) \right)$
return $\mathbf{r}(\mathbf{x})$

Results

(d) RCV1-X $N = 781K, D = 47K, L = 2.5K$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{91.23 \pm 0.22}$	$\mathbf{73.51 \pm 0.25}$	$\mathbf{53.31 \pm 0.65}$
MLRF	87.66 ± 0.46	69.89 ± 0.43	50.36 ± 0.74
LPSR	90.04 ± 0.19	72.27 ± 0.20	52.34 ± 0.61
1-vs-All	90.18 ± 0.18	72.55 ± 0.16	52.68 ± 0.57