Online probabilistic label trees (OPLTs)

Kalina Jasinska-Kobus*,[1,3] Marek Wydmuch*,[1] Devanathan Thiruvenkatachari[2] Krzysztof Dembczyński[1,2]

*Equal contribution [1]*Poznań University of Technology, Poland* [2]*Yahoo! Research, New York, USA* [3]*ML Research at Allegro.pl, Poland*

Motivation

- In modern machine learning applications, the label space can be enormous, containing even millions of different labels (**eXtreme Classification (XC)**):
  - content annotation for multimedia search,
  - different types of recommendation: webpages-to-ads, ads-to-bid-words, users-to-items, queries-to-items, or items-to-queries.
- In these applications learning algorithms run in rapidly changing environments. The space of labels and features might grow over time, as new data points arrive.
- Retraining a XC model every time a new label is observed is expensive.
- There is need for XC algorithms that can efficiently adapt to the growing label and feature space.

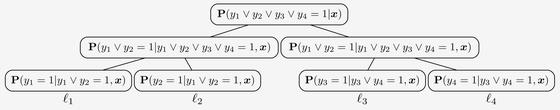
Probabilistic Label Trees (PLTs)

- **Multi-label classification**:

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \xrightarrow{\boldsymbol{h}(\boldsymbol{x})} \boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \{0,1\}^m$$

- An instance $\boldsymbol{x} \in \mathcal{X}$ is associated with a subset of labels $\mathcal{L}_{\boldsymbol{x}} \subseteq \mathcal{L}$ (positive labels). Set $\mathcal{L}_{\boldsymbol{x}}$ is identified with the binary vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m) \in \mathcal{Y}$, in which $y_j = 1 \Leftrightarrow \ell_j \in \mathcal{L}_{\boldsymbol{x}}$.
- **Probabilistic Label Trees (PLTs)** [1] use tree T (with set of nodes V_T), in which each leaf corresponds to one label, to factorize conditional probabilities of labels:

$$\eta_{l_j}(\boldsymbol{x}) = \mathbf{P}(y_j = 1 | \boldsymbol{x}) = \prod_{v \in \mathrm{Path}(l_j)} \eta(\boldsymbol{x}, v)$$



- PLTs has been recently implemented in several state-of-the-art algorithms: Parabel [2], extremeText [3], Bonsai [4], AttentionXML [5].

References

[1] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hüllermeier. Extreme F-measure maximization using sparse probability estimates. In *ICML*, 2016

[2] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *World Wide Web*, 2018

[3] Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *NeurIPS*, 2018

[4] Sujay Khandagale, Han Xiao, and Rohit Babbar. Bonsai - diverse and shallow trees for extreme multi-label classification. 2019

[5] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *NeurIPS*. 2019

[6] Alina Beygelzimer, John Langford, Yury Lifshits, Gregory B. Sorkin, and Alexander L. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI*, 2009

[7] Wen Sun, Alina Beygelzimer, Hal Daumé Iii, John Langford, and Paul Mineiro. Contextual memory trees. In *ICML*, 2019

Incremental PLTs

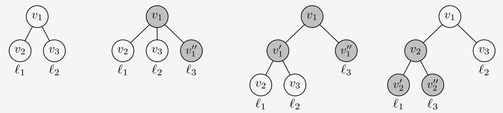
- PLT can be trained incrementally (**Incremental PLT (IPLT)**), on observations from $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, using an incremental learning algorithm A_{online} (e.g. Adagrad) for updating the **tree node classifiers** $\hat{\eta}$.
- IPLT requires the tree structure T to be given in advance.
- In each iteration, AssignToNodes procedure returns the set of **positive** and **negative nodes** for given tree T and set of positive labes $\mathcal{L}_{\boldsymbol{x}_i}$.

Algorithm 1 IPLT.Train$(T, A_{\text{online}}, \mathcal{D})$

for $(\boldsymbol{x}_i, \mathcal{L}_{\boldsymbol{x}_i}) \in \mathcal{D}$ **do** ▷ For each observation in $\mathcal{D}$
  $(P, N) =$ AssignToNodes$(T, \mathcal{L}_{\boldsymbol{x}_i})$ ▷ Compute its positive and negative nodes
  for $v \in P$ **do** A_{online}.Update$(\hat{\eta}_v, (\boldsymbol{x}_i, 1))$ ▷ Update classifiers of all positive nodes
  for $v \in N$ **do** A_{online}.Update$(\hat{\eta}_v, (\boldsymbol{x}_i, 0))$ ▷ Update classifiers of all negative nodes

Online Probabilistic Label Trees (OPLTs)

- **Online Probabilistic Label Tree (OPLT)** is an extension of IPLT that trains a label tree classifier **fully online – the tree is constructed simultaneously with the incremental training of node classifiers**, without any prior knowledge of the set of labels or training data.
- OPLT uses **auxiliary node classifiers** $\hat{\theta}$ that accumulate positive updates and are used to initialize classifiers in new nodes added to a tree.
- UpdateTree builds the tree structure. It iterates over all new labels from $\mathcal{L}_{\boldsymbol{x}}$ and adds them to the tree according to policy A_{policy}.
- The tree needs to be extended by one or two nodes for each new label, there are in general three variants of performing this step:



Tree T_{t-1} after $t-1$ iterations with label l_1 and l_2.

Variant 1: A leaf node v_1'' for new label l_3 added as a child of an internal node v_1.

Variant 2: A leaf node v_1'' for new label l_3 and an internal node v_1' (with all children of v_1 reassigned to it) added as children of v_1.

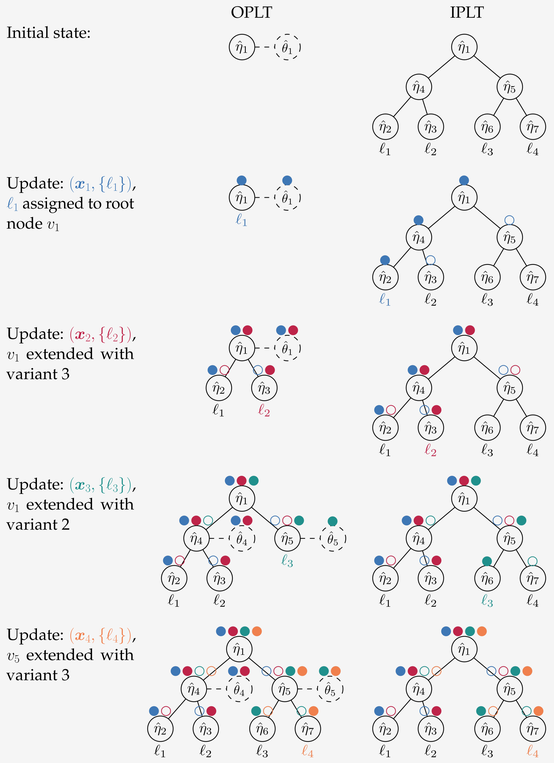
Variant 3: A leaf node v_2'' for new label j and a leaf node v_2' (with a reassigned label of v_2) added as children of v_2.

Algorithm 2 OPLT.Train$(T, A_{\text{online}}, A_{\text{policy}}, \mathcal{D})$

for $(\boldsymbol{x}_i, \mathcal{L}_{\boldsymbol{x}_i}) \in \mathcal{D}$ **do** ▷ For each observation in $\mathcal{D}$
  if $\mathcal{L}_{\boldsymbol{x}_i} \setminus \mathcal{L}_{t-1} \neq \emptyset$ **then** ▷ If the observation contains new labels,
    $T =$ UpdateTree$(\boldsymbol{x}_i, \mathcal{L}_{\boldsymbol{x}_i}, T, A_{\text{policy}})$ ▷ add them to the tree.
  $(P, N) =$ AssignToNodes$(T, \mathcal{L}_{\boldsymbol{x}_i})$ ▷ Compute its positive and negative nodes
  for $v \in P$ **do** ▷ For all positive nodes
    if $\hat{\theta}(v) \in \Theta$ **then** ▷ If auxiliary classifier exists,
      A_{online}.Update$(\hat{\theta}_v, (\boldsymbol{x}, 1))$ ▷ update it with a positive update with $\boldsymbol{x}_i$.
    A_{online}.Update$(\hat{\eta}_v, (\boldsymbol{x}, 1))$ ▷ Update node classifier with a positive update
  for $v \in N$ **do** A_{online}.Update$(\hat{\eta}_v, (\boldsymbol{x}, 0))$ ▷ Update classifiers of all negative nodes

Two properties of OPLT

- **Two properties of OPLT in relation to IPLT**, which we formalize and prove:
  - **properness**: the model trained by OPLT is equivalent to an IPLT model trained with the tree structure obtained by OPLT.
  - **efficiency**: space and time complexity of OPLT is in a constant factor of the complexity of IPLT.



Implementation

- Two policies A_{policy} for OPLT, both traverse a tree from top to bottom, until they reach a pre-leaf node (a parent of a leaf node):
  - **Random policy (OPLT$_R$)** – randomly selects child node.
  - **Best-greedy policy (OPLT$_B$)** – selects a child node using a trade-off between the balancedness of the tree and fit of $\boldsymbol{x}$. Similar to the tree building algorithm of Conditional Probability Tree (CPT) [6].
- Similar to Parabel [2] and extremeText [3], pre-leaf nodes can be of much higher arity than other internal nodes.

Results

- **Contextual Memory Tree (CMT)** [7] is currently the only algorithm that addresses the problem of fully online learning in the extreme setting.
- OPLT can also be used with an initial tree structure. We call such approach **OPLT with warm-start (OPLT-W)**. Here, we use a sample of 10% of training examples to construct the initial tree.

Table 1: Standard and propensity scored precision at $\{1, 5\}$ and train CPU time of Parabel, PLT, CMT, OPLT for extreme multi-label classification tasks.

	AmazonCat-13K					Wiki10-31K				
Algorithm	P@1	P@5	PP@1	PP@5	T_{train}	P@1	P@5	PP@1	PP@5	T_{train}
Parabel	92.64	63.81	50.89	71.27	10.8m	84.17	63.30	11.67	13.77	4.2m
IPLT	93.11	63.98	49.97	70.77	34.2m	84.87	65.31	12.26	14.80	18.3m
CMT	89.43	54.23	47.48	56.03	168.2m	80.59	55.25	9.68	9.67	35.1m
OPLT$_R$	92.66	62.52	48.98	67.51	99.5m	84.87	65.31	11.64	14.12	30.3m
OPLT$_B$	92.74	62.91	49.30	68.64	84.1m	84.87	65.31	11.59	14.26	27.7m
OPLT-W	93.14	63.92	49.86	70.44	43.7m	84.87	65.31	11.69	14.43	28.2m
	WikiLSHTC-320K					Amazon-670K				
Algorithm	P@1	P@5	PP@1	PP@5	T_{train}	P@1	P@5	PP@1	PP@5	T_{train}
Parabel	62.78	30.27	25.91	34.77	14.4m	43.13	34.00	25.39	31.21	7.2m
IPLT	60.80	29.24	24.16	32.29	175.1m	43.55	35.20	25.25	32.12	79.4m
OPLT$_R$	47.76	23.37	16.12	22.68	330.1m	38.42	31.32	20.28	27.35	134.2m
OPLT$_B$	54.69	26.31	20.69	28.14	300.0m	41.09	33.42	23.41	30.49	111.9m
OPLT-W	59.23	28.38	22.56	30.23	205.7m	42.21	34.25	23.65	30.95	98.3m

Figure 1: Online progressive performance of CMT and OPLT with respect to the number of samples on few-shot multi-class classification tasks.

Source code: `https://github.com/mwydmuch/napkinXC`

The 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021.