Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications

Himanshu Jain, Yashoteja Prabhu
Indian Institute of Technology Delhi
himanshu.j689@gmail.com,
yashoteja.prabhu@gmail.com

Manik Varma
Microsoft Research
manik@microsoft.com

ABSTRACT

The choice of the loss function is critical in extreme multi-label learning where the objective is to annotate each data point with the most relevant *subset* of labels from an extremely large label set. Unfortunately, existing loss functions, such as the Hamming loss, are unsuitable for learning, model selection, hyperparameter tuning and performance evaluation. This paper addresses the issue by developing propensity scored losses which: (a) prioritize predicting the few relevant labels over the large number of irrelevant ones; (b) do not erroneously treat missing labels as irrelevant but instead provide unbiased estimates of the true loss function even when ground truth labels go missing under arbitrary probabilistic label noise models; and (c) promote the accurate prediction of infrequently occurring, hard to predict, but rewarding tail labels. Another contribution is the development of the PfastreXML algorithm (code available from [1]) which efficiently scales to large datasets with up to 9 million labels, 70 million points and 2 million dimensions and which gives significant improvements over the state-of-the-art.

This paper's results also apply to tagging, recommendation and ranking which are the motivating applications for extreme multi-label learning. They generalize previous attempts at deriving unbiased losses under the restrictive assumption that labels go missing uniformly at random from the ground truth. Furthermore, they provide a sound theoretical justification for popular label weighting heuristics used to recommend rare items. Finally, they demonstrate that the proposed contributions align with real world applications by achieving superior clickthrough rates on sponsored search advertising in Bing.

1. INTRODUCTION

Extreme multi-label learning addresses the problem of learning a classifier that can annotate a data point with the most relevant *subset* of labels from an extremely large label set. Note that multi-label learning is distinct from multi-class classification which aims to predict a single mutually exclusive label.

Extreme multi-label learning is an important research problem as it has many applications in tagging, recommendation and ranking. For instance, there are more than a million labels (tags) on Wikipedia and one might wish to build an extreme multi-label classifier that tags a new article or web page with the subset of most relevant Wikipedia labels. Similarly, given a user's buying or viewing history, one might wish to build an extreme multi-label classifier that recommends the subset of millions of items that the user might wish to buy or view next. In general, one can reformulate ranking and recommendation problems as extreme classification tasks by treating each item to be ranked/recommended as a separate label, learning an extreme multi-label classifier that maps a user's feature vector to a set of labels, and then using the classifier to predict the subset of items that should be ranked/recommended to each user.

Extreme multi-label learning differs from traditional multi-label learning in a number of ways including the need for logarithmic time prediction, training at an extreme scale with millions of data points, features and labels, *etc.* Two aspects are germane to this paper. First, every data point has missing labels in its ground truth labelling since it is impossible for annotators to go through millions of labels and mark out the exact relevant subset. This has a fundamental impact on training, validation and performance evaluation. Second, the notion of what constitutes a good prediction changes when one moves from traditional to extreme multi-label learning. In particular, due to relevant label sparsity, it is more important to accurately predict relevant labels than irrelevant ones. Furthermore, due to the power law distribution over labels, infrequently occurring tail labels have little training data and are harder to predict than frequently occurring ones but might also be more informative and rewarding. As such, design choices made for traditional multi-label learning might not apply at the extreme scale.

One of the most critical design choices is that of the loss function. It determines whether the training algorithm learns a good solution, whether hyper-parameters are tuned appropriately, influences model selection and, perhaps most importantly, ensures that performance evaluation on the test set is aligned with real world application requirements.

This paper argues that traditional multi-label loss functions are unsuitable for extreme multi-label learning even though they have been used extensively thus far. For instance, the popular Hamming loss [5, 7, 10, 11, 14, 19, 22,

KDD '16, August 13 - 17, 2016, San Francisco, CA, USA

ISBN 978-1-4503-4232-2/16/08…$15.00

DOI: http://dx.doi.org/10.1145/2939672.2939756

40, 43, 48] does not prioritize predicting the few relevant labels over the millions of irrelevant ones, erroneously treats missing labels as irrelevant, treats all relevant labels as being equally important and is biased due to missing ground truth. As a result, extreme multi-label models optimized and selected using traditional loss functions might perform poorly when deployed in real world applications.

The primary contribution of this paper is to develop loss functions suitable for extreme multi-label learning. It is argued that losses which focus on ranking relevant labels as highly as possible are more suitable than the Hamming loss. Propensity scored variants of such losses, including precision@k and nDCG@k, are developed and proved to give unbiased estimates of the true loss function even when ground truth labels go missing under arbitrary probabilistic label noise models. Furthermore, it is shown that the propensity models developed in this paper based on real world applications naturally promote the accurate prediction of infrequently occurring, difficult to predict, but rewarding tail labels. This addresses both the limitations of traditional multi-label loss functions as discussed in this paper. Another contribution is the development of the PfastreXML algorithm that can scale to extreme multi-label datasets with up to 9 million labels, 70 million training points and 2 million dimensional features and achieves significant improvements over the state-of-the-art. The code for PfastreXML is available from [1].

This paper's results generalize beyond extreme multi-label learning and are also relevant to tagging, recommendation and ranking. Previous attempts at developing unbiased loss functions in these areas have been limited to square loss [17], recall [38] and average discounted gain [26] under the restrictive assumption that ground truth labels go missing uniformly at random. Furthermore, the propensity models developed in this paper present a sound theoretical justification for the popular label weighting heuristics [9, 12, 35, 39, 44, 47, 49, 51] used in the recommendation literature to promote the prediction of rare and novel items. Finally, the fact that higher clickthrough rates are achieved while ranking queries for sponsored search advertising in Bing demonstrates that the loss functions and algorithms proposed in this paper are better aligned with real world applications.

2. RELATED WORK

Extreme multi-label learning algorithms typically follow a tree [4, 34, 46] or an embedding based approach [5, 6, 7, 10, 11, 14, 18, 20, 22, 27, 32, 36, 43, 45, 48, 50]. While some algorithms have been proposed for training with missing labels [23, 40, 48] under restrictive settings, aspects such as hyper-parameter tuning, model selection and performance evaluation have not been addressed before. As such, the Hamming loss [5, 7, 10, 11, 14, 19, 22, 40, 43, 48] continues to be one of the most popular losses for extreme multi-label learning along with precision [4, 6, 18, 21, 22, 34, 45, 46] and the F-measure [5, 11, 15, 19, 20, 22, 40, 50]. On the other hand, unbiased estimators for recall [38], average discounted gain [26] and square loss [17] have been developed under the restrictive assumption that labels go missing uniformly at random from the ground truth. By contrast, this paper develops propensity scored variants of precision, nDCG and other loss functions and proves that they are unbiased even under general probabilistic label noise models.

Propensity scoring has been used to develop unbiased estimators for observational data [37]. In machine learning, propensities have been used for bias correction in situations where the training and test data have been drawn from different distributions [3]. Propensities have also been used for off policy evaluation, whereby feedback data from the interaction logs of an existing system is used to evaluate a new system [8, 24, 25, 41, 42].

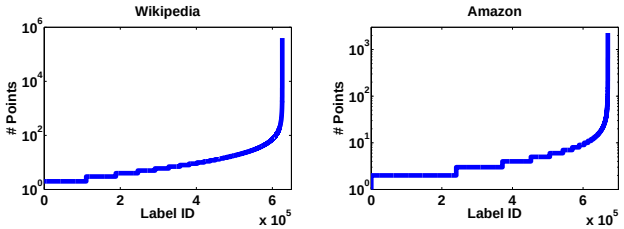


Figure 1: Plot showing the number of times each label occurs in a dataset: 246201 and 452262 labels occur less than 5 times each in Wikipedia and Amazon respectively. Such labels are harder to predict than popular ones but might also be more informative and rewarding in certain applications.

Label (item) weighting loss functions have been proposed to promote the accurate prediction of infrequently occurring labels (rare items) which might delight and surprise the user. For instance, denoting a label's normalized frequency of occurrence by p_l, [9, 44, 47, 49, 51] used the heuristic of weighting each label by $-\log p_l$ whereas [12, 35, 39] recommended a weight of $p_l^{-\gamma/(\gamma+1)}$ where $\gamma \geq 0$ was a user tunable parameter. This paper provides theoretical justification for such heuristics by showing how similar weights can arise from the proposed propensity models.

3. A MOTIVATING EXAMPLE

Consider evaluating the performance of an extreme multi-label algorithm for tagging Wikipedia articles by computing a chosen loss function on the ground truth labels provided by Wikipedia's editors. To take a concrete example, Wikipedia's editors annotated the article for the Divine Comedy with 15 labels such as "14th-century Christian texts", "Epic poems in Italian", "1300 in Italy", *etc.* Note that many relevant labels such as "Dante Alighieri", "Medieval philosophical literature" and "Allegory" are missing since it is impossible for any annotator or expert to go through the entire list of Wikipedia labels and select all the relevant ones. Evaluating performance using the Hamming loss leads to the following issues which can be overcome by the proposed propensity scored nDCG@k and precision@k.

Relevant labels: Table 2 shows that the number of labels relevant to any given data point is far smaller than the number of irrelevant ones. Accurately predicting a relevant label is therefore more important than predicting an irrelevant one. For instance, predicting that "Epic poems in Italian" is relevant to the Divine Comedy is more difficult, and informative, than predicting that "Baseball" is irrelevant. Similarly, it is more important to accurately predict relevant labels to fill the few slots available in typical recommendation applications than it is to predict irrelevant ones. Unfortunately, the Hamming loss charges the same penalty for misclassifying relevant and irrelevant labels. Precision@k and nDCG@k avoid this by promoting the prediction of relevant labels with high ranks.

Missing labels: The Hamming loss would penalize an algorithm for predicting that the label "Dante Alighieri" was relevant to the article for the Divine Comedy since the label was missing from the ground truth. Section 4 addresses this issue by developing propensity scored variants of precision@k and nDCG@k which provide unbiased estimates of the true loss as if computed on the complete ground truth without any missing labels.

Tail labels: Labels follow a power law distribution in extreme multi-label learning applications (see Figure 1). Infrequently occurring labels have little training data and are harder to predict than frequently occurring ones but might also be more informative and rewarding. This is particularly important on a dataset such as Wikipedia where 246201 labels occur in less than 5 articles each. For instance, little information is gained by predicting popular generic labels such as "Poems" for the Divine Comedy article as compared to predicting relatively infrequent labels such as "Epic poems in Italian" (which implies "Poems" and more) or "14th-century Christian texts". Similarly, in certain applications, there is little to be gained by recommending popular items since users might know about them already. Predicting rare items might be more desirable in these cases. While existing losses treat all labels as equal, Section 5 develops a propensity model that naturally promotes the accurate prediction of infrequent labels with high ranks.

4. PROPENSITY SCORED LOSSES

This Section develops propensity scored variants of precision@k, nDCG@k and other popular loss functions (see Table 1 for examples). It is proved that the proposed propensity scored losses computed on the observed labels provide unbiased estimates of the true loss function computed on the complete (but unobtainable) ground truth without any missing labels.

Label representation: Extreme multi-label learning deals with applications having an extremely large number of labels L where it is not possible for any annotator to select the exact relevant label subset for even a single data point. Let $\mathbf{y}^*, \mathbf{y} \in \{0,1\}^L$ denote the complete (but unobtainable) and observed (but with missing labels) ground truth label vectors for a given data point such that $y_l^* = y_l = 1$ for observed relevant labels, $y_l^* = 1, y_l = 0$ for unobserved relevant labels and $y_l^* = y_l = 0$ for irrelevant labels. Continuing with the example of the Divine Comedy from Section 3, $y_l^* = y_l = 1$ for the observed relevant label "Epic poems in Italian", while $y_l^* = 1, y_l = 0$ for the relevant, but missing, label "Dante Alighieri" and $y_l^* = y_l = 0$ for the irrelevant label "Baseball". Furthermore, it is assumed that the noise in the labelling process is one sided and that irrelevant labels are never marked as relevant. For instance, Wikipedia's editors are not malicious and never allow an article to be tagged with an irrelevant label. Note that, even if this mild assumption was violated, it might be possible to hire annotators to weed out the irrelevant tags. Also note that, since $\mathbf{y}^*$ is unavailable, this label representation does not assume that the position of missing labels is known unlike previous work [48]. Finally, let $\hat{\mathbf{y}} \in \{0,1\}^L$ denote an algorithm's predicted label vector for a given data point.

Table 1: (a) presents unbiased propensity scored loss functions $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ corresponding to precision@k and nDCG@k for an unrestricted probabilistic label noise model which is the focus of this paper. The unbiased losses in (b), including the Mean Reciprocal Rank (MRR) and the Average Discounted Gain (ADG), require either knowledge of $\mathbf{1}^\top \mathbf{y}^*$ or that labels go missing with probability $1 - g_l/\mathbf{1}^\top \mathbf{y}^*$ with known g_l (except for the F-score). Note that $\hat{\mathbf{y}}$ has only k non-zero entries for precision@k, nDCG@k and recall@k and that r_l represents the rank of label l in $\hat{\mathbf{y}}$.

(a)		(b)	
Gain	$-\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$	Gain	$-\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$
Precision@k	$\frac{1}{k}\sum_l \frac{1}{p_l} y_l \hat{y}_l$	Recall@k	$\frac{1}{\mathbf{1}^\top \mathbf{y}^*}\sum_l \frac{1}{p_l} y_l \hat{y}_l$
nDCG@k	$\frac{\sum_l \frac{y_l \hat{y}_l}{p_l \log(r_l+1)}}{\left(\sum_{l=1}^k \frac{1}{\log(1+l)}\right)}$	MRR	$\frac{1}{\mathbf{1}^\top \mathbf{y}^*}\sum_l \frac{y_l}{p_l r_l}$
		ADG	$\frac{1}{\mathbf{1}^\top \mathbf{y}^*}\sum_l \frac{y_l}{p_l \log(r_l+1)}$
-Hamming Loss	$\sum_l \left(\frac{1}{p_l}(2\hat{y}_l - 1)\right) y_l - \hat{y}_l^2$	F_β score	$\frac{(1+\beta^2)}{\beta^2(\mathbf{1}^\top \mathbf{y}^*) + \mathbf{1}^\top \hat{\mathbf{y}}} \sum_l \frac{1}{p_l} y_l \hat{y}_l$

Propensities: The propensity $p_{il} \equiv P(y_{il} = 1 | y_{il}^* = 1)$ denotes the marginal probability of a relevant label l being observed for a data point i. No constraints have been placed on the propensities apart from the fact that label noise is one sided – i. e. $P(y_{il} = 1 | y_{il}^* = 0) = 0$. In particular, it is not assumed that labels go missing independently or uniformly at random. Note that, for notational convenience, the subscript i will be dropped from p_{il} even though the propensity depends on both the label l and the data point i.

Propensity scored loss functions: Let $\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}}) = \sum_{l=1}^L \mathcal{L}_l^*(y_l^*, \hat{y}_l) = \sum_{l:y_l^*=1}^L \mathcal{L}_l^*(1, \hat{y}_l)$ denote the family of loss functions which decompose over individual labels l and are computed over the relevant labels alone ($\{l | y_l^* = 1\}$). $\mathcal{L}^*$ represents the true loss function measuring the loss incurred for predicting $\hat{\mathbf{y}}$ when the complete ground truth vector was $\mathbf{y}^*$. Training and performance evaluation using $\mathcal{L}^*$ is desirable but infeasible as $\mathbf{y}^*$ is unavailable. The propensity scored variant of $\mathcal{L}^*$ computed on the observed ground truth $\mathbf{y}$ is defined to be $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{l:y_l=1}^L \mathcal{L}_l(1, \hat{y}_l) = \sum_{l:y_l=1}^L \mathcal{L}_l^*(1, \hat{y}_l)/p_l$. Then the following theorem implies that $\mathcal{L}$ can be a viable proxy for $\mathcal{L}^*$ for training, model selection, hyperparameter tuning and performance evaluation.

THEOREM 4.1. *The loss function $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ evaluated on the observed ground truth $\mathbf{y}$ is an unbiased estimator of the true loss function $\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}})$ evaluated on the complete ground truth $\mathbf{y}^*$. Thus, $\mathbb{E}_{\mathbf{y}}[\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})] = \mathbb{E}_{\mathbf{y}^*}[\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}})]$, for any $P(\mathbf{y}^*)$ and $P(\mathbf{y})$ related through propensities p_l and any fixed $\hat{\mathbf{y}}$.*

PROOF. Please click here for the supplementary material containing the proof. □

Theorem 4.1 covers loss functions which decompose over individual labels such as precision@k and nDCG@k which are the primary focus of this paper. Unbiased estimators of recall, average discounted gain, mean reciprocal rank and other relevant non-decomposable loss functions can also be derived if it is assumed that $P(\mathbf{y}^*)$ is a delta function implying that each ground truth label is either definitely relevant or definitely irrelevant (with no uncertainty) to the data point being annotated.

THEOREM 4.2. *If $P(\mathbf{y}^*)$ is a delta function then $\mathbb{E}_{\mathbf{y}}[\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})] = \mathbb{E}_{\mathbf{y}^*}[\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}})]$ for non-decomposable loss functions of the form $\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}}) = \sum_{l:y_l^*=1}^L \frac{\mathcal{L}_l^*(1, \hat{y}_l)}{g^*(\mathbf{y}^*, \hat{\mathbf{y}})}$ and $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{l:y_l=1}^L \frac{\mathcal{L}_l^*(1, \hat{y}_l)}{g^*(\mathbf{y}^*, \hat{\mathbf{y}}) p_l}$ with arbitrary propensities p_l.*

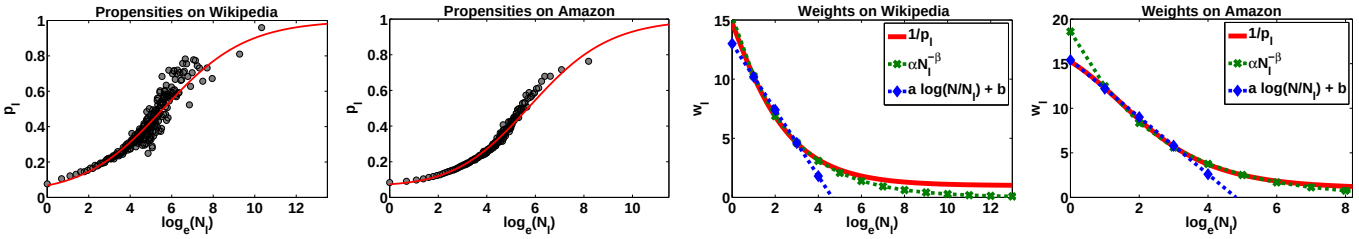


Figure 2: Propensities p_l and their corresponding weights $w_l = 1/p_l$ on Wikipedia and Amazon. The estimated propensities follow a sigmoidal curve on the semi-log plot and provide a principled setting of the weights for recommending rare items as compared to popular heuristics such as $N_l^{-\beta}$ and $\log(N/N_l)$.

PROOF. Please click here for the supplementary material containing the proof. □

Note that Theorem 4.2 is useful only if $g^*(\mathbf{y}^*, \hat{\mathbf{y}})$ can be evaluated even though $\mathbf{y}^*$ is unknown. This is certainly possible in some applications. For instance, in the case of recall, $g^*(\mathbf{y}^*, \hat{\mathbf{y}}) = \mathbf{1}^\top \mathbf{y}^*$ counts how many labels were relevant in the complete ground truth. This can be readily estimated by counting the number of face detections ($\mathbf{1}^\top \mathbf{y}^*$) in a given image in name tagging applications on Facebook even though the name of each individual might not be known ($\mathbf{y}^*$ is unknown). Alternatively, if $g^*(\mathbf{y}^*, \hat{\mathbf{y}})$ is unknown, the following corollary derives unbiased estimators when $g^*(\mathbf{y}^*, \hat{\mathbf{y}}) = g^*(\mathbf{y}^*)$ and labels go missing with probability $1 - g_l/g^*(\mathbf{y}^*)$ with known g_l. Note that this generalizes the results of [26, 38] which assume that labels go missing uniformly at random with constant $g_l = \mathbf{1}\top\mathbf{y}$.

COROLLARY 4.2.1. *If $P(\mathbf{y}^*)$ is a delta function and labels are retained with propensities $p_l = g_l/g^*(\mathbf{y}^*)$, then $\mathbb{E}_{\mathbf{y}}[\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})] = \mathbb{E}_{\mathbf{y}^*}[\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}})]$ for non-decomposable loss functions of the form $\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}}) = \sum_{l:y_l^*=1}^{L} \frac{\mathcal{L}_l^*(1,\hat{y}_l)}{g^*(\mathbf{y}^*)}$ and $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{l:y_l=1}^{L} \frac{\mathcal{L}_l^*(1,\hat{y}_l)}{g_l}$.*

PROOF. Please click here for the supplementary material containing the proof. □

The theorems so far prove that the propensity scored losses are unbiased in expectation. In practice, one can only compute the propensity scored loss on the observed labels rather than in expectation over $\mathbf{y}$. The following theorem shows that the bias induced by this point estimate depends on the average number of relevant labels rather than the total number of labels and that it reduces as the number of points over which the loss is computed is increased.

THEOREM 4.3. ***(Concentration bound)*** *Let $\mathbf{Y} = \{\mathbf{y}_i \in \{0,1\}^L\}_{i=1}^N$ be a set of N independent observed ground truth random variables. Then with probability at least $1 - \delta$ $\left|\mathbb{E}_{\mathbf{Y}}\left[\frac{1}{N}\sum_{i=1}^N \mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i)\right] - \frac{1}{N}\sum_{i=1}^N \mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i)\right| \leq \rho\bar{L}\sqrt{\frac{1}{2N}\log\left(\frac{2}{\delta}\right)}$ where $\rho = max_{il}\left|\frac{1}{p_{il}}\frac{\mathcal{L}_l^*(y_{il},\hat{y}_{il})}{g(\mathbf{y}_i^*,\hat{\mathbf{y}}_i)}\right|$, $\bar{L} = \sqrt{\frac{1}{N}\sum_{i=1}^N L_i^{*2}}$ and L_i^* is the maximum number of labels that can be relevant to a data point i in the complete ground truth.*

PROOF. Please click here for the supplementary material containing the proof. □

Finally, an unbiased estimator can also be derived for the Hamming loss even though other loss functions, such as precision@k and nDCG@k, might be preferable

THEOREM 4.4. *For any $P(\mathbf{y}^*)$ and $P(\mathbf{y})$ related through propensities p_l and any fixed $\hat{\mathbf{y}}$, $\mathbb{E}_{\mathbf{y}}[\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})] = \mathbb{E}_{\mathbf{y}^*}[\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}})]$ where $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_l \left(\frac{1}{p_l}(1 - 2\hat{y}_l)\right) y_l + \hat{y}_l^2$ is an unbiased estimator of the Hamming loss $\mathcal{L}^*(\mathbf{y}^*, \hat{\mathbf{y}}) = \sum_l \|y_l^* - y_l\|^2$ with concentration bound $\rho\bar{L}\sqrt{\frac{1}{2N}\log(2/\delta)}$ where $\rho = max_{il}(1/p_{il})$.*

PROOF. Please click here for the supplementary material containing the proof. □

5. PROPENSITY MODEL

The unbiased variants of precision@k, nDCG@k and other loss functions developed in Section 4 require that the marginal propensities of labels being retained is known. Unfortunately, propensities are generally unknown as $\mathbf{y}^*$ is unavailable due to the large label space. Based on empirical observation, this Section proposes that the propensities might be modelled as a sigmoidal function of $\log N_l$

$$p_l \equiv P(y_l = 1 | y_l^* = 1) = \frac{1}{1 + Ce^{-A\log(N_l + B)}} \qquad (1)$$

where N_l is the number of data points annotated with label l in the observed ground truth dataset of size N and A, B are application specific parameters and $C = (\log N - 1)(B+1)^A$. In particular, propensities are estimated on Wikipedia and Amazon where meta-data is available for the task and shown to give a close fit to (1) (see Figure 2).

Tagging on Wikipedia ($A = 0.5, B = 0.4$): The marginal propensity of a label can be estimated as $p_l = N_l/N_l^*$ where N_l and N_l^* are the number of times the label occurred in the observed and complete ground truth respectively. Estimates of N_l^* can be obtained for Wikipedia by leveraging its hierarchy. It is assumed that if a label is relevant to a Wikipedia article then so are all its ancestor labels. For instance, the Divine Comedy article has been annotated with "1300 in Italy" and should therefore have also been annotated with its ancestor "14th century in Italy". Examining all Wikipedia articles revealed that 45 articles were annotated with a descendant of the label "14th century in Italy" but only 10 were annotated with the label itself resulting in a propensity estimate of $p_l = 10/(10 + 45) = 0.182$. This procedure was carried out for all labels with more than 4 descendants for robust estimation. Labels with similar frequencies were binned together in an equiheight histogram with 20 labels per bin. Figure 2 plots the average propensity per bin as a function of label frequency on a log scale. As can be seen, the proposed sigmoid propensity model with parameters $A = 0.5$ and $B = 0.4$ is a close fit to the estimated propensities.

Product recommendation on Amazon ($A = 0.6, B = 2.6$): The item-to-item recommendation task on Amazon is to predict the subset of items (labels) that a user might buy along with a given item. In this case, N_l represents the number of items that item l was bought along with in the observed dataset (across all transactions) while N_l^* represents the total number of items that item l could have been bought along with. While N_l is known, N_l^* can be estimated by figuring out which items could have been substituted in place of the ones that item l was bought along with. It has been argued that substitutable products can be inferred from the "also viewed" items while complimentary items can be inferred from the "also bought" items [28, 30]. Following this principle, N_l^* can be estimated as the total number of unique items viewed along with items that item l was "also bought" along with. For robustness, propensities $p_l = N_l/N_l^*$ were estimated only for those items for which "also viewed" information was available for each of the items that item l was "also bought" with . Items with similar N_l were binned together in an equiheight histogram as in the case of Wikipedia. Figure 2 plots the average propensity per bin as a function of item frequency on a log scale. As can be seen, the proposed sigmoid propensity model with parameters $A = 0.6$ and $B = 2.6$ is a close fit to the estimated propensities.

Recommending rare items: It is important to remove the popularity bias and recommend rare/novel items in many applications [9, 12, 35, 39, 44, 47, 49, 51]. A common heuristic is to weight each item inversely to its popularity and to assign weighted rewards for accurate recommendations. Weights have often been set in an ad hoc fashion as $w_l \propto N_l^{-\beta}$ [12, 35, 39] or $w_l \propto \log(N/N_l)$ [9, 44, 47, 49, 51].

Such weights arise naturally as inverse propensities in the unbiased losses developed in this paper. As can be seen from (1) and Table 1, each label in the proposed unbiased losses has a weight given by

$$w_l = 1/p_l = 1 + C(N_l + B)^{-A} \tag{2}$$

which somewhat matches $(N_l + B)^{-A}$ and $\log(N/N_l)$ in different ranges of N_l (see Figure 2). This not only provides a sound theoretical justification of label weighting heuristics for recommending rare items but also leads to a more principled setting of the weights.

Other datasets: Propensity estimation might not be possible on datasets where meta information is not available. In such cases, it is recommended that $A = (0.5 + 0.6)/2 = 0.55$ and $B = (0.4 + 2.6)/2 = 1.5$ are set to their values averaged over Wikipedia and Amazon. This was verified to be reasonably close to the parameter settings on the Wiki10 dataset ($A = 0.55, B = 0.1$).

6. ALGORITHMS

This Section develops the PfastreXML algorithm for extreme multi-label learning. PfastreXML optimizes propensity scored nDCG by leveraging FastXML [34] for nDCG optimization. PfastreXML then further extends FastXML to improve tail label prediction which is the most challenging aspect of extreme multi-label learning. PfastreXML achieves this at scale by making key approximations which increase FastXML's training time by just seconds while retaining the prediction accuracy gains of the extension.

6.1 Propensity scored FastXML

Classifier architecture: Propensity scored FastXML (PfastXML) shares the same architecture as FastXML [34] which learns an ensemble of trees during training. Trees are grown by recursively partitioning nodes starting at the root until each tree is fully grown. Nodes are split by learning a separating hyperplane which partitions training points between a left and a right child. The FastXML hyperplane is learnt by optimizing nDCG such that each training point's relevant labels are ranked as highly as possible in its partition. Node partitioning terminates when a node contains less than a user specified number of points. Leaf nodes contain a probability distribution over labels generated by normalizing the frequency counts of all the training labels reaching the node.

Predictions are made in logarithmic time by passing a test point down each of the balanced trees in the ensemble. The test point is sent to an internal node's left (right) child if it lies on the negative (positive) side of the separating hyperplane at that node. The label distributions of all the leaves containing the test point are aggregated in order to make a prediction as follows:

$$\mathbf{P}_{\text{pf}}(\mathbf{y}^*|\mathbf{x}) = \frac{\sum_{t=1}^{T} \mathbf{P}_t^{\text{leaf}}(\mathbf{x})}{T} \tag{3}$$

Propensity scored objective function: PfastXML improves upon FastXML by replacing the nDCG loss with its propensity scored variant which is unbiased and assigns higher rewards for accurate tail label predictions. Given a set of N training points at a node $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$ with features $\mathbf{x}_i \in \mathcal{R}^D$ and observed ground truth label vectors $\mathbf{y}_i \in \{0,1\}^L$, PfastXML's separating hyperplane $\mathbf{w}^*$ at the node is given by the optimal solution of

$$\begin{aligned} \min \quad & \|\mathbf{w}\|_1 + C_\delta \sum_i \log(1 + e^{-\delta_i \mathbf{w}^\top \mathbf{x}_i}) \\ & + C_r \sum_i \tfrac{1}{2}(1 + \delta_i) \mathcal{L}_{\text{PSnDCG@}L}(\mathbf{r}^+, \mathbf{y}_i) \\ & + C_r \sum_i \tfrac{1}{2}(1 - \delta_i) \mathcal{L}_{\text{PSnDCG@}L}(\mathbf{r}^-, \mathbf{y}_i) \\ \text{w. r. t.} \quad & \mathbf{w} \in \mathcal{R}^D, \boldsymbol{\delta} \in \{-1,+1\}^L, \mathbf{r}^+, \mathbf{r}^- \in \Pi(1, L) \end{aligned} \tag{4}$$

where $\mathcal{L}_{\text{PSnDCG@}L}(\mathbf{r}, \mathbf{y}) = -\frac{\sum_l \frac{y_l}{p_l \log(r_l+1)}}{\sum_{l=1}^{L} \frac{1}{\log(1+l)}}$, i indexes all 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 $\mathbf{r}^+$ and $\mathbf{r}^-$ represent the predicted label rankings for the positive and negative partition respectively.

Optimization: Note that FastXML's objective function can be recovered from PfastXML's by substituting $y_{il}^p = y_{il}/p_{il}$ – i. e. by replacing each label y_{il} with label y_{il}^p. PfastXML's objective function can therefore be optimized by FastXML's iterative alternating optimization applied to y_{il}^p. In each iteration, the algorithm fixes $\mathbf{w}$ and alternates between optimizing $\boldsymbol{\delta}$ and $\mathbf{r}^\pm$ using efficient closed form solutions until a stationary point is reached according to Theorem 1 of [34]. Then $\boldsymbol{\delta}$ and $\mathbf{r}^\pm$ are fixed and $\mathbf{w}$ is optimized by solving a standard l_1 regularized logistic regression binary classification problem using Liblinear [13].

Scale: PfastXML enjoys all the scaling properties of FastXML and is therefore one of the most efficient extreme multi-label

Table 2: Dataset statistics

Dataset	Train N	Features D	Labels L	Test M	Avg. labels per point	Avg. points per label
EUR-Lex	15,539	5,000	3,993	3,809	5.31	25.73
AmazonCat-13K	1,186,239	203,882	13,330	306,782	5.05	566.01
Wiki10-31K	14,149	101,938	30,935	6,613	17.25	11.58
WikiLSHTC-325K	1,778,351	1,617,899	325,056	587,084	3.26	23.74
Amazon-670K	490,449	135,909	670,091	153,025	5.38	5.17
Ads-9M	70,455,530	2,082,698	8,838,461	22,629,136	1.79	14.32

learning algorithm for large scale problems. It could train on WikiLSHTC-325K (325 K labels, 1.7 M training points and 1.6 M dimensions) and Ads-9M (9 M labels, 70 M training points and 2 M dimensions) in less than 30 minutes and 17 hours respectively using a 16 core Intel Xeon 2.6 GHz server. In contrast, MLRF [4] required 4 and 42 hours on a thousand core cluster for training on these datasets and resulted in significantly lower prediction accuracies [34]. No other algorithm has been shown to scale to datasets of the size of Ads-9M to the best of our knowledge. Equally importantly in terms of scaling, PfastXML's predictions required less than 1.5 milliseconds per test point even at the largest scale which is critical for deployment in real world applications.

6.2 PfastreXML

Propensity scoring improves FastXML but tree classifiers are still prone to predicting tail labels with low probabilities as partitioning errors in the internal nodes disproportionately reduce the support of tail labels in the leaf node distributions. PfastreXML addresses this limitation by re-ranking PfastXML's predictions using classifiers designed specifically for tail labels. PfastreXML's training and prediction routines are shown in Algorithms 2 and 3 of the supplementary material while code is available from [1].

Tail label classifiers: It is assumed that each label can be predicted independently based on a hyperspherical decision boundary generated by

$$P(y_{il}^*|\mathbf{x}_i) = 1/(1+v_{il}^{2y_{il}^*-1}) \quad \text{where } v_{il} = e^{\frac{\gamma}{2}\|\mathbf{x}_i-\boldsymbol{\mu}_l\|_2^2} \quad (5)$$

Discriminative MLE estimation of the parameters $\{\boldsymbol{\mu}_l\}$ on the *observed* training data $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$ with features $\mathbf{x}_i \in \mathcal{R}^D$ and observed ground truth label vectors $\mathbf{y}_i \in \{0,1\}^L$ can be carried out as $\arg\max_{\{\boldsymbol{\mu}_l\}} \prod_{i=1}^N \prod_{l=1}^L \sum_{y_{il}^*=0}^1 P(y_{il}|y_{il}^*)P(y_{il}^*|\mathbf{x}_i)$ where it has been reasonably assumed that $P(y_{il}|y_{il}^*, \mathbf{x}_i) = P(y_{il}|y_{il}^*)$.

Optimization: Taking the log leads to L independent optimization problems

$$\boldsymbol{\mu}_l^* = \arg\max_{\boldsymbol{\mu}_l} \sum_{i=1}^N \log\left((1-y_{il}) + \frac{p_{il}(2y_{il}-1)}{1+v_{il}}\right) \quad (6)$$

Taking the gradient and equating it to zero yields $\boldsymbol{\mu}_l^* = \frac{\sum_{i=1}^N u_{il}\mathbf{x}_i}{\sum_{i=1}^N u_{il}}$ where $u_{il} = \frac{v_{il}}{1+v_{il}} - \frac{(1-y_{il})v_{il}}{1+v_{il}-p_{il}}$. Note that this is not a closed form solution since v_{il} is a function of $\boldsymbol{\mu}_l^*$ and hence one needs to apply an optimization technique, such as stochastic gradient descent, to obtain $\boldsymbol{\mu}_l^*$.

Approximation: Unfortunately, stochastic gradient descent is too expensive at the scale of Ads-9M. A simple, yet effective, approximation is therefore proposed which allowed training on Ads-9M in 18 minutes on a single core (all other datasets were trained in less than 20 seconds), led to sparse solutions taking up only 2.64 Gigabytes of RAM for Ads-9M and reduced prediction accuracy over the stochastic gradient descent solution by 0.2% as measured on the smaller datasets. In particular, assuming that $\frac{\gamma}{2}\|\mathbf{x}_i-\boldsymbol{\mu}_l\|^2 \gg 0 \forall\ i \in \{1,..,N\}$, led to the simplification $u_{il} \approx y_{il}$ yielding

$$\boldsymbol{\mu}_l^* = \frac{\sum_{i=1}^N y_{il}\mathbf{x}_i}{\sum_{i=1}^N y_{il}} \quad (7)$$

Thus, each $\boldsymbol{\mu}_l^*$ turns out to be the mean of the training points for which the label was observed to be relevant. This implies that solutions preserving data sparsity can be efficiently computed for millions of tail labels as each of them has only a handful of training points.

Re-ranking: The final ranked list of labels is predicted by sorting a linear combination of (3) and (5)

$$s_l = \alpha \log P_{\text{pf}}(y_l^* = 1|\mathbf{x}) + (1-\alpha)\log P(y_l^* = 1|\mathbf{x}) \quad (8)$$

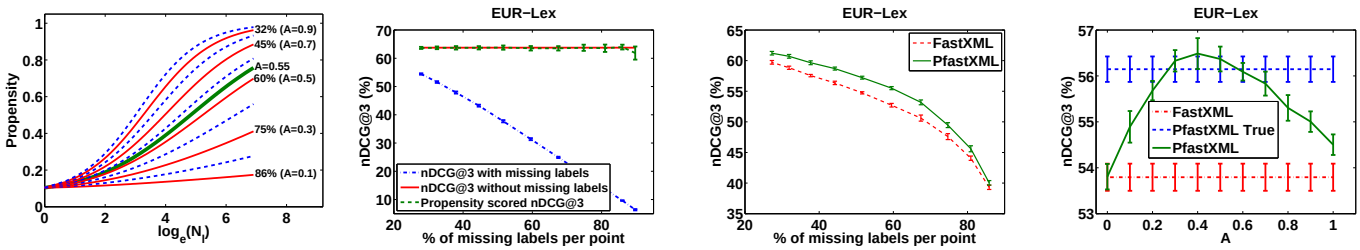
restricted to those l for which $P_{\text{pf}}(y_l^* = 1|x) \neq 0$. Note that the re-ranking takes place in 0.13 milliseconds per point on Ads-9M so that the overall prediction time continues to be less than 1.5 milliseconds per test point.

7. EXPERIMENTS

Experiments were carried out on a synthetic dataset to show that the proposed propensity scored loss functions are unbiased and preferable for both training and performance evaluation. Experiments were also carried out on the largest benchmark datasets demonstrating that PfastreXML could achieve significantly higher prediction accuracies as compared to the state-of-the-art. Improvements in the click-through rates on Bing Ads indicated that the proposed loss functions and algorithms were better suited for real world applications.

Datasets: Experiments were carried out on extreme multi-label datasets including Ads-9M, Amazon-670K [6, 29], Wiki10-31K [6, 52],WikiLSHTC-325K [33, 34], AmazonCat-13K [29] and EUR-Lex [31]. The Ads-9M dataset is proprietary. All the other datasets are publically available and can be downloaded from The Extreme Classification Repository [2]. The tasks include annotating Wikipedia articles with the subset of relevant 325 K Wikipedia tags, item-to-item recommendation on Amazon with 670 K items and ranking 9 M queries for sponsored search advertising on Bing. Table 2 lists the statistics of these datasets.

Baseline algorithms: PfastreXML was compared to a number of baseline extreme multi-label algorithms including FastXML [34] and SLEEC [6] which are the leading tree and embedding based approaches respectively. Other



(a) (b) (c) (d)

Figure 3: (a) propensity curves used for simulating missing labels on the EUR-Lex dataset with each curve labelled with the corresponding percentage of missing labels; (b) propensity scored nDCG@k is unbiased; (c) propensity scoring improves training; and (d) training using incorrect propensities ($A \neq 0.55$) might be better than training without propensities. See text for details. Figure best viewed under magnification.

baseline algorithms include 1-vs-All [16], LEML [48], WSABIE [45], CPLST [10], CS [18], ML-CSSP [7] and LPSR [46]. A Popularity baseline was also included which predicted a constant ranking of the most frequently occurring labels in each dataset. Unfortunately, some of the algorithms did not scale beyond the EUR-Lex dataset and results are presented in Table 3.

Implementations of SLEEC, FastXML, LEML and 1-vs-All were provided by the authors. The remaining algorithms were implemented by us taking care to ensure that the published results could be reproduced and were verified by the authors wherever possible.

Hyper-parameters: PfastreXML has 2 hyper-parameters α, γ in addition to FastXML's hyper-parameters. These were set to constants $\alpha = 0.8, \gamma = 30$ across all datasets. All of FastXML's hyper-parameters were also set to constant default values across all datasets as was done in [34]. This helps significantly reduce training time by eliminating hyper-parameter tuning. The hyper-parameters for all the other algorithms were set using fine grained validation on each data set so as to achieve the highest possible prediction accuracy.

Evaluation metrics: Given a set of M test points, performance was evaluated using the unbiased propensity scored loss functions of Table 1 as $\mathcal{G}(\{\hat{\mathbf{y}}_i\}) = \frac{1}{M}\sum_{i=1}^{M}\mathcal{L}(\mathbf{y}_i, \hat{\mathbf{y}}_i)$. Note that the gain $\mathcal{G}$ could be greater than 1 due to the propensities. Therefore, for greater interpretability, Table 3 reports $100*\mathcal{G}(\{\hat{\mathbf{y}}_i\})/\mathcal{G}(\{\mathbf{y}_i\})$ for Precision@k and nDCG@k, referred to as Pk and Nk respectively, for $k = 1, 3$ and 5. Coverage@k measuring the percentage of normalized unique labels present in the top $k = 1, 3$ and 5 predictions made by an algorithm across all test points is also reported as Ck.

Simulations: It was assumed that the complete ground truth was available for the EUR-Lex dataset and missing labels were simulated according to the propensity curves shown in Figure 3a.

In the second experiment, FastXML was trained on missing labels by optimizing nDCG while PfastXML was trained on the same set by optimizing propensity scored nDCG. Both methods were evaluated using nDCG@3 computed on the complete ground truth with no missing labels. As Figure 3c shows, PfastXML consistently outperformed FastXML thereby indicating that propensity scoring could improve training. A related concern might be that PfastXML (or PfastreXML) might outperform FastXML just because it optimizes the loss function being used for evaluation while FastXML does not. This experiment demonstrates otherwise as PfastXML was trained using propensity scored nDCG but evaluated using standard nDCG. Results of similar experiments on the benchmark datasets are provided in the supplementary material.

Finally, Figure 3d demonstrates that training with incorrect propensities might be better than training with no propensities. In this experiment, labels were removed from the training set using a reference propensity curve depicted in bold in Figure 3a. FastXML and PfastXML were trained on this set and their performance evaluated using nDCG@3 computed on the complete test set without any missing labels. PfastXML was then trained on the very same training set but using incorrect propensities (also shown in Figure 3a with $A \neq 0.55$). As can be seen, PfastXML trained on incorrect propensities could outperform FastXML trained without propensities. This indicates that using even incorrect propensity estimates might be beneficial in certain situations as compared to using no propensities.

Benchmark Results: Table 3 compares PfastreXML's performance to that of state-of-the-art SLEEC, FastXML and other baseline algorithms using unbiased precision and nDCG. As can be seen, the proposed PfastXML and PfastreXML lead to significantly better prediction accuracies as compared to the state-of-the-art. PfastreXML's improvements ranged from 3% on Ads-9M to more than 20% on AmazonCat-13K. Figure 4 shows that most of these improvements were made for infrequently occurring tail labels. In addition, Table 4 shows that PfastreXML predicted a larger number of unique labels than SLEEC or FastXML further indicating that PfastreXML had better coverage in the tail.

PfastreXML also improves upon PfastXML's prediction accuracy with negligible training and prediction overheads. For instance, PfastXML could train on WikiLSHTC-325K and Ads-9M in less than 30 minutes and 17 hours respectively using a 16 core Intel Xeon 2.6 GHz server. PfastreXML took an extra 12 seconds and 18 minutes for training on these datasets using a single core. PfastreXML's predictions took an extra 0.13 milliseconds per test point over PfastXML's and continued to be under 1.5 milliseconds.

Sponsored search on Bing: PfastreXML's query rankings were also used to serve ads on the Bing search engine. PfastreXML was observed to give an improvement of sig-

Table 3: The proposed PfastreXML and PfastXML algorithms make significantly more accurate predictions as compared to state-of-the-art SLEEC, FastXML and other baseline algorithms. PfastreXML's predictions are more accurate than PfastXML's with negligible training and prediction overheads. Performance is evaluated according to the unbiased propensity scored Precision@k (Pk) and nDCG@k (Nk) for $k = 1, 3$ and 5.

(a) EUR-Lex $N = 15K, D = 5K, L = 4K$

Algorithm	N1(%)	N3(%)	N5(%)	P1(%)	P3(%)	P5(%)
Popularity	1.80	2.10	2.36	1.80	2.20	2.62
1-vs-All	37.97	42.44	43.97	37.97	44.01	46.17
SLEEC	35.45	39.79	41.97	35.45	41.35	44.62
LEML	24.33	26.45	27.70	24.33	27.22	29.13
WSABIE	31.65	34.12	35.43	31.65	35.04	36.99
CPLST	28.93	31.60	32.92	28.93	32.57	34.55
CS	25.31	26.98	25.71	25.31	27.57	25.13
ML-CSSP	25.25	26.70	27.79	25.25	27.27	28.97
FastXML	27.61	33.22	36.28	27.61	35.35	39.95
LPSR	33.65	38.20	39.82	33.65	39.88	42.17
PfastXML	41.31	44.01	45.13	41.31	45.02	46.67
PfastreXML	**45.38**	**46.42**	**47.25**	**45.38**	**46.79**	**48.08**

(b) AmazonCat-13K $N = 1.18M, D = 203K, L = 13K$

Algorithm	N1(%)	N3(%)	N5(%)	P1(%)	P3(%)	P5(%)
Popularity	14.41	13.08	13.22	14.41	12.59	12.95
SLEEC	46.75	55.19	60.08	46.75	58.46	65.96
FastXML	46.58	55.48	61.59	46.58	59.00	68.50
PfastXML	67.44	70.70	72.27	67.44	71.94	74.32
PfastreXML	**69.05**	**71.79**	**73.33**	**69.05**	**72.83**	**75.21**

(c) Wiki10-31K $N = 14K, D = 101K, L = 31K$

Algorithm	N1(%)	N3(%)	N5(%)	P1(%)	P3(%)	P5(%)
Popularity	2.30	2.26	2.28	2.30	2.25	2.29
SLEEC	13.43	13.66	13.81	13.43	13.75	13.96
FastXML	10.31	10.18	10.35	10.31	10.14	10.42
PfastXML	15.62	16.05	16.42	15.62	16.20	16.71
PfastreXML	**21.32**	**19.97**	**19.48**	**21.32**	**19.53**	**18.93**

(d) WikiLSHTC-325K $N = 1.78M, D = 1.62M, L = 325K$

Algorithm	N1(%)	N3(%)	N5(%)	P1(%)	P3(%)	P5(%)
Popularity	2.56	1.91	1.83	2.56	1.65	1.53
SLEEC	20.51	22.45	23.52	20.51	23.32	25.23
FastXML	16.52	19.70	21.17	16.52	21.12	23.69
PfastXML	25.58	26.55	27.42	25.58	27.01	28.59
PfastreXML	**31.16**	**31.56**	**32.40**	**31.16**	**31.80**	**33.35**

(e) Amazon-670K $N = 490K, D = 136K, L = 670K$

Algorithm	N1(%)	N3(%)	N5(%)	P1(%)	P3(%)	P5(%)
Popularity	0.03	0.04	0.04	0.03	0.04	0.04
SLEEC	20.62	22.63	24.43	20.62	23.32	25.98
FastXML	20.20	22.94	25.26	20.20	23.88	27.28
PfastXML	25.61	26.95	28.09	25.61	27.42	29.09
PfastreXML	**29.93**	**30.91**	**31.94**	**29.93**	**31.26**	**32.80**

(f) Ads-9M $N = 70.45M, D = 2.08M, L = 8.84M$

Algorithm	N1(%)	N3(%)	N5(%)	P1(%)	P3(%)	P5(%)
Popularity	0.05	0.08	0.09	0.05	0.09	0.12
FastXML	12.89	14.86	15.61	12.89	15.88	17.26
PfastXML	13.27	15.39	16.32	13.27	16.49	18.22
PfastreXML	**13.52**	**16.43**	**17.79**	**13.52**	**17.95**	**20.50**

nificantly more than 5% in the clickthrough rate over both FastXML and the highly specialized system in production

Table 4: PfastreXML has more unique labels Ck in the top $k = 1, 3$ and 5 predictions across all test points in a dataset as compared to SLEEC or FastXML indicating that it has better coverage of tail labels.

(a) EUR-Lex $N = 15K, D = 5K, L = 4K$

Algorithm	C1 (%)	C3 (%)	C5 (%)
1-vs-All	33.50	44.29	53.82
SLEEC	27.21	38.10	49.26
FastXML	16.39	28.39	40.49
PfastreXML	**48.81**	**56.29**	**62.31**

(b) AmazonCat-13K $N = 1.18M, D = 203K, L = 13K$

Algorithm	C1 (%)	C3 (%)	C5 (%)
SLEEC	8.62	23.50	49.16
FastXML	2.13	15.76	53.69
PfastreXML	**83.03**	**85.12**	**86.49**

(c) Wiki10-31K $N = 14K, D = 101K, L = 31K$

Algorithm	C1 (%)	C3 (%)	C5 (%)
SLEEC	5.94	5.73	6.59
FastXML	1.57	2.12	2.88
PfastreXML	**25.67**	**18.48**	**17.67**

(d) WikiLSHTC-325K $N = 1.78M, D = 1.62M, L = 325K$

Algorithm	C1 (%)	C3 (%)	C5 (%)
SLEEC	14.06	24.53	30.11
FastXML	9.52	18.43	23.29
PfastreXML	**29.63**	**36.45**	**40.67**

(e) Amazon-670K $N = 490K, D = 136K, L = 670K$

Algorithm	C1 (%)	C3 (%)	C5 (%)
SLEEC	24.19	26.18	30.97
FastXML	22.06	25.25	30.94
PfastreXML	**34.58**	**36.41**	**40.13**

(f) Ads-9M $N = 70.45M, D = 2.08M, L = 8.84M$

Algorithm	C1 (%)	C3 (%)	C5 (%)
FastXML	3.26	3.94	4.33
PfastreXML	**6.04**	**7.45**	**8.09**

(which was a large ensemble of many different ranking techniques). Note that the production system was very good at predicting head queries with high ranks and that PfastreXML was rewarded for making only those predictions which could not be made by the production system. Accurately ranking tail queries highly was therefore critical in this case and PfastreXML was able to successfully serve ads which had never received clicks before. This helps verify that the propensity scored loss functions and proposed algorithm align with the requirements of real world applications.

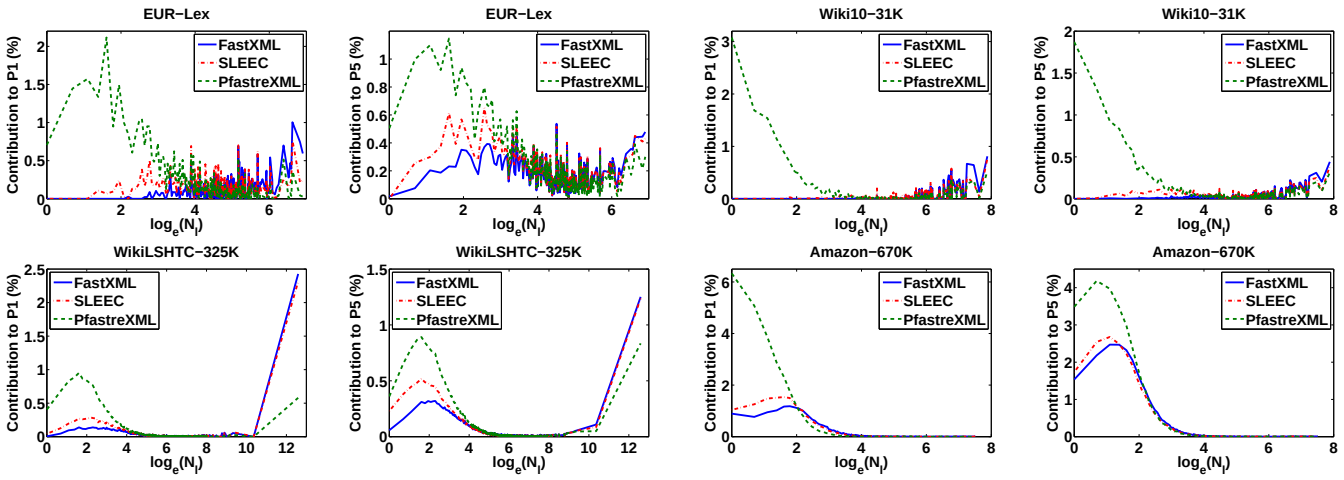


Figure 4: Plot showing the contribution of each label to the overall propensity scored Precision@1 and Precision@5. PfastreXML is significantly more accurate at predicting infrequently occurring (small N_l) tail labels. Figure best viewed under magnification

8. CONCLUSIONS

This paper developed loss functions suitable for extreme multi-label learning and long tail, missing label applications such as ranking, recommendation and tagging. Propensity scored variants of precision and nDCG were developed and proved to give unbiased estimates of the true loss function evaluated on the complete ground truth. No restrictions were placed on the propensities apart from the mild assumption that irrelevant labels were never marked as relevant. Furthermore, propensity models were developed based on real world applications and were shown to naturally promote the accurate prediction of infrequently occurring tail labels. This provides a sound theoretical justification of popular label weighting heuristics used to remove the popularity bias and recommend rare/novel items. The results also provide a more principled setting of the weights as compared to previous heuristics.

This paper also developed the PfastreXML algorithm for optimizing propensity scored nDCG. PfastreXML was shown to make significantly more accurate predictions on all datasets as compared to the state-of-the-art. PfastreXML was demonstrated to be specially well suited to predicting tail labels which is the most challenging aspect of extreme multi-label learning. This helped PfastreXML achieve significantly higher clickthrough rates for sponsored search advertising on Bing as compared to the large ensemble of highly specialized rankers currently in production. In terms of scaling, PfastreXML could train on WikiLSHTC-325K and Ads-9M in less than 30 minutes and 17 hours respectively using a 16 core Intel Xeon 2.6 GHz server. Finally, PfastreXML's predictions were made in under 1.5 milliseconds per test point which is critical for deployment in real world applications. The code for PfastreXML is available from [1].

Acknowledgement

We are grateful to Rahul Agrawal, Samy Bengio, Abhishek Kadian, Shrutendra Harsola, Purushottam Kar, Prateek Jain and Ambuj Tewari for discussions, feedback and help with experiments. Himanshu Jain is supported by a Google PhD Fellowship. Yashoteja Prabhu is supported by a TCS PhD Fellowship and MSR India travel grant. Manik Varma would like to thank the School of Information Technology at IIT Delhi where he holds an adjunct position.

References

[1] Code for PfastreXML. http://manikvarma.org/code/PfastreXML/download.html.

[2] The Extreme Classification Repository. http://manikvarma.org/downloads/XC/XMLRepository.html.

[3] D. Agarwal, L. Li, and A. J. Smola. Linear-time estimators for propensity scores. In *AISTATS*, 2011.

[4] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, 2013.

[5] K. Balasubramanian and G. Lebanon. The landmark selection method for multiple output prediction. In *ICML*, 2012.

[6] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, 2015.

[7] W. Bi and J. T. Kwok. Efficient multi-label classification with many labels. In *ICML*, 2013.

[8] L. Bottou, J. Peters, J. Quinonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR*, 2013.

[9] P. Castells, S. Vargas, and J. Wang. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. In *DDR*, 2011.

[10] Y. N. Chen and H. T. Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, 2012.

[11] M. Cissé, N. Usunier, T. Artières, and P. Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, 2013.

[12] C. Dhanjal, R. Gaudel, and S. Clémençon. Collaborative filtering with localised ranking. In *AAAI*, 2015.

[13] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 2008.

[14] C. S. Ferng and H. T. Lin. Multi-label classification with error-correcting codes. In *ACML*, 2011.

[15] S. Gopal and Y. Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *KDD*, 2013.

[16] B. Hariharan, S. V. N. Vishwanathan, and M. Varma. Efficient max-margin multi-label classification with applications to zero-shot learning. *ML*, 2012.

[17] C. J. Hsieh, N. Natarajan, and I. Dhillon. PU learning for matrix completion. In *ICML*, 2015.

[18] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, 2009.

[19] K. Jasinska and K. Dembczyǹski. Consistent label tree classifiers for extreme multi-label classification. In *Extreme Classification Workshop, ICML*, 2015.

[20] S. Ji, L. Tang, S. Yu, and J. Ye. Extracting shared subspace for multi-label classification. In *KDD*, 2008.

[21] A. Kapoor, R. Viswanathan, and P. Jain. Multilabel classification using bayesian compressed sensing. In *NIPS*, 2012.

[22] N. Karampatziakis and P. Mineiro. Scalable multilabel prediction via randomized methods. *CoRR*, 2015.

[23] X. Kong, Z. Wu, L. J. Li, R. Zhang, P. S. Yu, H. Wu, and W. Fan. Large-scale multi-label learning with incomplete label assignments. In *SDM*, 2014.

[24] L. Li, S. Chen, J. Kleban, and A. Gupta. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *WWW*, 2015.

[25] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

[26] D. Lim, J. McAuley, and G. Lanckriet. Top-N recommendation with missing implicit feedback. In *RecSys*, 2015.

[27] Z. Lin, G. Ding, M. Hu, and J. Wang. Multi-label classification via feature-aware implicit label space encoding. In *ICML*, 2014.

[28] J. McAuley, T. Christopher, S. Qinfeng, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.

[29] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.

[30] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *KDD*, 2015.

[31] E. L. Mencia and J. Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *SIGIR*, 2008.

[32] P. Mineiro and N. Karampatziakis. Fast label embeddings for extremely large output spaces. In *ECML*, 2015.

[33] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artières, G. Paliouras, É. Gaussier, I. Androutsopoulos, M. R. Amini, and P. Gallinari. LSHTC: A benchmark for large-scale text classification. *CoRR*, 2015.

[34] Y. Prabhu and M. Varma. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, 2014.

[35] B. Pradel, N. Usunier, and P. Gallinari. Ranking with non-random missing ratings: Influence of popularity and positivity on evaluation metrics. In *RecSys*, 2012.

[36] P. Rai, C. Hu, R. Henao, and L. Carin. Large-scale bayesian multi-label learning via topic-based label embeddings. In *NIPS*, 2015.

[37] P. R. Rosenbaum and D. B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 1983.

[38] H. Steck. Training and testing of recommender systems on data missing not at random. In *KDD*, 2010.

[39] H. Steck. Item popularity and recommendation accuracy. In *RecSys*, 2011.

[40] Y. Y. Sun, Y. Zhang, and Z. H. Zhou. Multi-label learning with weak label. In *AAAI*, 2010.

[41] A. Swaminathan and T. Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *ICML*, 2015.

[42] A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *NIPS*, 2015.

[43] F. Tai and H. T. Lin. Multi-label classification with principal label space transformation. In *MLD*, 2010.

[44] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *RecSys*, 2011.

[45] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

[46] J. Weston, A. Makadia, and H. Yee. Label partitioning for sublinear ranking. In *ICML*, 2013.

[47] H. Wu, X. Cui, J. He, B. Li, and Y. Pei. On improving aggregate recommendation diversity and novelty in folksonomy-based social systems. *Personal and Ubiquitous Computing*, 2014.

[48] H. F. Yu, P. Jain, P. Kar, and I. S. Dhillon. Large-scale multi-label learning with missing labels. In *ICML*, 2014.

[49] L. Zhang. The definition of novelty in recommendation system. *JESTR*, 2013.

[50] Y. Zhang and J. G. Schneider. Multi-label output codes using canonical correlation analysis. In *AISTATS*, 2011.

[51] T. Zhou, Z. Kuscsik, J. G. Liu, M. Medo, J. R. Wakeling, and Y. C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proc. Nat. Acad. Sci. USA*, 2010.

[52] A. Zubiaga. Enhancing navigation on wikipedia with social tags. *CoRR*, 2012.