Extreme Regression for Dynamic Search Advertising

Yashoteja Prabhu[*†]
t-yaprab@microsoft.com

Aditya Kusupati[‡§]
kusupati@cs.washington.edu

Nilesh Gupta[*]
t-nilgup@microsoft.com

Manik Varma[*†]
manik@microsoft.com

ABSTRACT

This paper introduces a new learning paradigm called eXtreme Regression (XR) whose objective is to accurately predict the numerical degrees of relevance of an extremely large number of labels to a data point. XR can provide elegant solutions to many large-scale ranking and recommendation applications including Dynamic Search Advertising (DSA). XR can learn more accurate models than the recently popular extreme classifiers which incorrectly assume strictly binary-valued label relevances. Traditional regression metrics which sum the errors over all the labels are unsuitable for XR problems since they could give extremely loose bounds for the label ranking quality. Also, the existing regression algorithms won't efficiently scale to millions of labels. This paper addresses these limitations through: (1) *new evaluation metrics* for XR which sum only the k largest regression errors; (2) a *new algorithm* called XReg which decomposes XR task into a hierarchy of much smaller regression problems thus leading to highly efficient training and prediction. This paper also introduces a (3) *new labelwise prediction algorithm* in XReg useful for DSA and other recommendation tasks.

Experiments on benchmark datasets demonstrated that XReg can outperform the state-of-the-art extreme classifiers as well as large-scale regressors and rankers by up to 50% reduction in the new XR error metric, and up to 2% and 2.4% improvements in terms of the propensity-scored precision metric used in extreme classification and the click-through rate metric used in DSA respectively. Deployment of XReg on DSA in Bing resulted in a relative gain of 58% in revenue and 27% in query coverage. XReg's source code can be downloaded from [1]

CCS CONCEPTS

• **Computing methodologies** → **Machine Learning**; *Supervised learning by regression.*

KEYWORDS

Extreme classification, dynamic search advertising, regression

ACM Reference Format:
Yashoteja Prabhu, Aditya Kusupati, Nilesh Gupta, and Manik Varma. 2020. Extreme Regression for Dynamic Search Advertising. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3336191.3371768

1 INTRODUCTION

Objective: This paper introduces a new learning paradigm called eXtreme Regression (XR) which can provide elegant solutions to many large-scale ranking and recommendation applications including Dynamic Search Advertising (DSA). To effectively solve XR problems, this paper also develops new evaluation metrics and a new highly scalable and accurate algorithm called XReg.

eXtreme Regression: The objective of eXtreme Regression is to learn to accurately predict the numerical degrees of relevance of an extremely large number of labels with respect to a data point. Many large-scale ranking and recommendation applications can naturally be reformulated as XR problems. For example, the tasks of DSA, movie recommendation and document tagging can be posed as the problems of predicting the search queries' click probabilities for an ad, the users' ratings for a movie and the informativeness of tags while describing a web document, respectively. These qualify as XR problems since the total number of queries, users and tags can potentially be in millions in these applications. The predicted relevance estimates could then be used to recommend the most relevant labels to a data point which is the desired end goal of recommendation systems. Alternatively, the recommendations can also be further refined by filtering off less relevant ones or by re-ranking them to improve their relevance, and the relevance estimates provide principled ways of achieving these. To successfully solve an XR problem, new algorithms which could train and predict efficiently over millions of labels as well as millions of data points while also maintaining high prediction accuracy are required. Furthermore, the definition of accuracy, or equivalently regression error, needs to be redefined for XR settings where both the relevant labels and the desired label recommendations are extremely small in number compared to the complete label set whose most labels have no influence on final recommendations. This paper addresses these challenges by developing new evaluation metrics and algorithms.

DSA: DSA is a format of search advertising where the ads to be shown against a search query, along with the associated ad-copy, ad-title, bid-phrases *etc.*, are algorithmically obtained by leveraging the content from the ad landing pages. This saves considerable efforts

[*]Microsoft Research India
[†]Indian Institute of Technology Delhi
[‡]University of Washington
[§]Work done during Research Fellowship at Microsoft Research India

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371768

for advertisers, results in faster deployment of new ad campaigns and enables more accurate user targeting. The ads shown by DSA algorithms need to be highly relevant and generate user clicks for the given query in order to earn revenue for the search engine and satisfy the users and advertisers. In addition, these algorithms need to train and predict very efficiently in order to scale to billions of ads and millions of search queries across multiple markets and maintain milliseconds' prediction latencies. This paper solves DSA as an XR task of estimating the click probabilities for the query, ad pairs by using the new XReg algorithm. Note that different ads can have different click probabilities for same query owing to multiple query intents. For example the query "throne" on Bing refers to an online strategy war game, an online tv series and a furniture product with click probabilities of 0.2, 0.06 and 0.004 respectively. Based on the predicted click probabilities, the less clickable ads are filtered off, the remaining ads are re-ranked to promote those of high quality and high advertiser bids, and a small number of top ranked ads are finally shown for the given query.

Extreme Classification: Extreme classifiers annotate a data point with the most relevant *subset* of labels from an extremely large label set. Owing to their high scalability and accuracy in label subset selection scenarios, extreme classifiers are increasingly being used for DSA [44] and other large-scale recommendation problems. Unfortunately, they make a fundamentally incorrect assumption that a label is either fully relevant or fully irrelevant to a data point which hurts their model accuracy. When applied to DSA, they approximate all click-through rates to either 0 or 1 during training and thus end up predicting less clickable ads. In turn, this also undermines further filtering and re-ranking steps due to the lack of reliable click probability estimates. Also, the ranking at the top metrics used for evaluating extreme classifiers ignore the errors in estimating the relevances and are hence not suitable for XR.

Regression and ranking: Multivariate regressors predict multiple numerical outcome variables as functions of the features of a data point. Although such regressors could reliably estimate the label relevances in XR, most existing regressors are designed for small number of outcome variables and do not scale to millions of labels in XR. Moreover, the standard regression metrics such as Mean Absolute Deviation (MAD) which sum the regression errors over all the labels are unsuitable for XR problems because the quality of recommended labels, both before and after the filtering and re-ranking steps, depend only on the accurate estimation of a small number of label relevances. The pairwise ranking approaches, which ensure that a more relevant item is ranked ahead of a less relevant one for each pair of items, have been extensively used for moderate-sized ranking and recommendation tasks. However, their complexity scales quadratically in number of labels and therefore don't scale to million labels.

eXtreme Regression metrics: This paper proposes new regression metrics for XR which serve as good proxies for the ranking accuracy and for the qualities of the subsequent label filtering and re-ranking steps. These metrics average of the largest few regression errors which are usually caused by highly underestimating or highly overestimating the relevances of the most or the least relevant labels which in turn degrade the ranking quality. The new XMAD@k metric can give up to 69x tighter bounds over ranking regret than MAD. These new metrics can guide the crucial steps in XR such as training, performance evaluation, hyper-parameter tuning, model selection *etc.*

eXtreme Regressor algorithm: This paper also develops a new eXtreme Regressor (XReg) algorithm which can efficiently regress on to millions of label relevance weights in only logarithmic time. XReg hierarchically clusters the labels into a balanced tree and learns approximate regressors in each tree node which are common to all the labels in the node. Due to high label sparsity, each data point only participates in a logarithmic number of tree nodes which can lead to a significant speed up during both training and prediction by using appropriate algorithms. XReg essentially extends the state-of-the-art Parabel extreme classifier to the regression setting. XReg consistently outperforms extreme classifiers, large-scale regressors and rankers in terms of ranking accuracy. On a DSA dataset with 5M ads & 1M queries, XReg can train within just 20 hours using 1 core, predict in just 3 ms per query and give up to 58% & 27% lifts in revenue and query coverage when deployed online.

Labelwise inference: The standard prediction scenario involves recommending the most relevant labels for a test point, referred here as pointwise prediction, but applications such as DSA and movie recommendation can more naturally be posed in the reverse manner of predicting the most relevant ads or movies (*i.e.* test points) for each query or user (*i.e.* each label), referred here as labelwise prediction. On these tasks, pointwise prediction might recommend a small set of highly popular labels that are relevant to all test points resulting in low label coverage. This paper develops an efficient labelwise prediction algorithm in XReg, which significantly improves the query coverage in DSA. Note that the XReg training is agnostic to the choice of the prediction setting and the same learnt model works well for both types of predictions.

Contributions: This paper: (a) introduces a new learning paradigm called eXtreme Regression (XR) and reformulates tagging, movie recommendation and DSA applications as XR problems; (b) develops new evaluation metrics and a highly scalable and accurate algorithm called XReg to effectively tackle XR problems; and (c) demonstrates that XReg can significantly improve revenue and query coverage on Bing DSA when deployed in production. XReg's source code can be downloaded from [1].

2 RELATED WORK

Extreme Classification: Much progress has recently been made in developing extreme multi-label classifiers based on trees [4, 25, 27, 43, 45, 50], embeddings [9, 11, 14, 18, 21, 35, 39, 52, 56, 59] and 1-vs-all approaches [5, 6, 24, 32, 36, 40, 44, 57, 60, 61, 64]. Among these, 1-vs-all approaches like DiSMEC [5], ProXML [6], Parabel [44] and Slice [24] achieve state-of-the-art results on Precision@k, nDCG@k and their propensity-scored counterparts, but train only from binary labels and are hence not apt for DSA. In terms of efficiency, Parabel is many orders faster to train and predict than DiSMEC and ProXML, hence XReg algorithm builds on top of Parabel. Slice only works on low-dimensional embeddings and does not scale to high-dimensional bag-of-words features used in this paper. Some extreme classifiers like PfastreXML [25] and LEML [65] can be easily adapted to learn from any relevance weights, but they tend to be inaccurate and inefficient since they train a large ensemble of

weak trees and inaccurate low-dimensional projections with linear reconstruction time, respectively.

Performance of extreme classifiers has traditionally been measured in terms of Precision@k and nDCG@k [9, 45]. Recently, propensity-scored metrics were introduced in [25] which give higher importance to more useful and informative tail labels. However, all these metrics ignore the regression error in the predicted relevance estimates when applied to XR.

Regression & ranking: Most of the conventional regression approaches [7, 16, 51, 54, 65] learn a separate regressor for each outcome variable and hence do not scale to millions of labels. This problem is mitigated to some extent in the multi-objective decision tree based approaches [4, 25, 30] which scale sublinearly in the number on outcome variables. However, these approaches suffer from low accuracy issues despite learning a large ensemble of weak trees. As seen from experiments, XReg can be significantly more scalable and accurate than the naive 1-vs-all least squares regressor [54], the more efficient LEML regressor with low-rank assumption on the parameter space [65] and the decision tree based PfastreXML [25]. The performance accuracy in regression have traditionally been measured by error metrics such as Mean Absolute Deviation (MAD) and Root Mean Square Error (RMSE) [10], but these are not appropriate for XR.

Learning to rank methods [12, 15, 22, 33, 34, 41, 46, 48, 58, 63] have been widely used in the recommendation and ranking literatures, primarily to re-rank a small shortlist of items which has been generated by simple heuristics like tf-idf scoring or by more scalable approaches like extreme classifiers or XReg. These rankers usually have super-linear dependence on the number of labels and hence do not scale to XR. Although negative label sampling could potentially be used to make these approaches more scalable, their ranking performance suffers significantly as demonstrated in Section 5 for the popular RankSVM [20, 33] and the more recent eXtreme Learning to Rank (XLR) [12] approaches. A plethora of accuracy metrics have been proposed in the ranking literature [9, 26, 29, 34, 45, 53, 67], but none of these measure the regression performance.

Dynamic search advertising: Various approaches have been proposed for DSA in the organic search literature including information retrieval based methods [28], probabilistic methods and topic models [55] and deep learning [23, 49]; however these do not work well for pithy ad-landing pages. Techniques based on landing page summarization [13], translation and query language models [47, 62] and keyword suggestion based on Wikipedia concepts [66] have also been proposed for sponsored search; but these suffer from low coverage problem. Extreme classifiers such as Parabel have also been used in DSA to improve accuracy and ad coverage; but they still suffer from low query coverage due to pointwise predictions. As demonstrated in Section 5, XReg significantly improves revenue and query coverage when included in the Bing DSA ensemble comprising all the above alternatives.

3 EXTREME REGRESSION METRICS

Notation: Let an XR dataset comprise N data points $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ where $\mathbf{x}_i \in \mathbb{R}^D$ is a D dimensional feature vector and $\mathbf{y}_i \in [0, \infty)^L$ is a ground truth relevance weight vector for point i. The weight y_{il} measures the true degree of relevance of label l to point i, with higher values indicating higher relevance. Similarly, let $\hat{\mathbf{y}}_i \in [0, \infty)^L$ denote the predicted relevance weight vector for point i. The function $S(\mathbf{v}, k)$ indicates the *ordered* index set of the k highest scoring labels in a score vector $\mathbf{v} \in [0, \infty)^L$.

Regression & ranking metrics: The regression metrics such as MAD and RMSE; the ranking metrics such as relevance-weighted Precision and nDCG at k (WP@k, WN@k) and Kendall's Tau at k (Tau@k) [29]; and WP@k-regret which is the difference between the optimal and the attained WP@k are pertinant for this paper. Their formulae are provided in the supplementary. The WP@k metric reduces to PSP@k, CTR@k suitable for DSA, or Rating@k based on whether $\mathbf{y}_i$ are set to inverse propensity-scored relevances, ad click-through rates, or user ratings respectively. Rating@k is the undiscounted version of the familiar rating-based nDCG@k metric used in recommender systems [26].

Extreme regression metrics: Let, $\mathbf{e}_i$ be the vector of regression errors where $e_{il} = |\hat{y}_{il} - y_{il}|$. The new XR metrics, eXtreme Mean Absolute Deviation at k (XMAD@k) and eXtreme Root Mean Square Error at k (XRMSE@k) are defined as follows:

$$\text{XMAD@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k} \sum_{l \in S(\mathbf{e}_i, k)} e_{il} \tag{1}$$

$$\text{XRMSE@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \sqrt{\frac{1}{k} \sum_{l \in S(\mathbf{e}_i, k)} e_{il}^2} \tag{2}$$

For ease of discussion, this paper mainly focusses on the XMAD metric, although most of the observations and results also apply to XRMSE. XMAD@k averages the k maximum regression errors but is minimized when all the L label relevances are predicted exactly right. The following lemma shows that XMAD serves as a good proxy for the ranking error. This is based on an intuition that the ranking errors at the top occur mainly due to either highly underestimating or highly overestimating the relevances of the most or the least relevant labels respectively leading to high regression errors on such labels.

Lemma 3.1. *For any true and predicted relevance vectors* $\mathbf{y}, \hat{\mathbf{y}} \in [0, \infty)^L$, $0 \leq$ *WP-regret@*$k(\hat{\mathbf{y}}, \mathbf{y}) \leq 2 *$ *XMAD@*$2k(\hat{\mathbf{y}}, \mathbf{y})$ *holds true.*

In addition, $0.5 * \text{XMAD@}2k(\hat{\mathbf{y}}, \mathbf{y}) \leq \text{WP-regret@}k(\hat{\mathbf{y}}, \mathbf{y})$ also usually holds empirically (see Section 5) thus making XMAD error a close bound for the ranking error.

Although the top ranked labels with the highest predicted relevances could directly be recommended to a test point, it usually helps to further improve the recommendations by either filtering or re-ranking. The objective of filtering step is to maximize both precision and recall by removing as many irrelevant labels across as many test points as possible. This is crucial in DSA where there are system limitations against online hosting of too many relevant query, ad pairs. The following lemma shows that when the estimated label relevances are almost accurate in terms of the XMAD metric, almost ideal precision-recall trade-offs could be obtained by directly using a global threshold on the predicted relevances.

Lemma 3.2. *Given a test set where the true and predicted relevance vectors of* i*th point are* $\mathbf{y}_i, \hat{\mathbf{y}}_i \in [0, \infty)^L$, $AUPRC \geq AUPRC^* - O(k * XMAD@k)$ *holds true where* $AUPRC, AUPRC^*$ *are the attained and ideal areas under the micro-averaged precision-recall curves plotted using a global threshold.*

The lemma assumes that the number of retained labels per each test point is less than k for the evaluated region of the curve. It is reasonable to set $k = \log(L)$ since only a small number of labels need to be recommended to each point.

Re-ranking the relevance estimates could significantly improve the final ranking quality, especially when the XMAD errors are small. An example of re-ranking is to combine these estimates with the scores from tail classifiers (see [25]) to improve the recommendation accuracies over rare labels. It is worth noting that bad relevance estimates, despite inducing a good initial ranking, could hurt the subsequent filtering or re-ranking performance. Unlike XMAD, the traditional MAD metric is sensitive to the sparsity in the $\hat{\mathbf{y}}$ vector which does not directly affect the ranking performance in any way. For example, MAD error becomes huge for a dense estimator like 1-vs-All least squares regressor since small regression errors could accrue over million labels into a large value. Results from Section 5 corroborate these observations.

Labelwise metrics: To evaluate performance in the labelwise prediction scenario, all the above ranking and regression metrics, defined for pointwise predictions, need to be redefined appropriately. The formulae for labelwise metrics are provided in the supplementary. Most discussions and results in this paper, while presented primarily for pointwise prediction case, also hold for labelwise prediction setting after interchanging the roles of data points and labels. To promote clarity, all pointwise and labelwise metrics will be used with suffixes "-p" and "-l" respectively.

Note that proofs for the lemmas in this section are available in the supplementary.

4 XREG: EXTREME REGRESSOR

This section describes XReg's key components including the label tree construction, the probabilistic regression model and the pointwise and labelwise prediction algorithms using the same model.

4.1 Label Tree Construction

XReg learns a small ensemble of up to 3 label trees quite similarly to Parabel. Each tree is grown by recursively partitioning the labels into two balanced groups. Label partitioning is achieved by a balanced spherical $k = 2$-means algorithm [44] is which takes as input the feature vectors for all those labels in the current node and outputs 2 label clusters, efficiently, in $O(\hat{D}L \log L)$ time where $\hat{D}$ is the number of non-zero features per data point. The feature vector for a label is represented by the unit vector that points along the average of the training points which are relevant to the label:

$$\mathbf{v}_l = \mathbf{v}'_l / \|\mathbf{v}'_l\|_2 \quad \text{where} \quad \mathbf{v}'_l = \sum_{i=1}^{N} y_{il} \mathbf{x}_i \tag{3}$$

This is based on the intuition that two labels are similar if they are active in similar training points. In DSA, two queries (labels) are similar according to the proposed representation if they lead to clicks on similar ads (training points). As a result, the k-means algorithm ensures that the labels relevant for a data point end up in the same leaf. Note that, unlike Parabel, XReg uses non-binary relevance-weighted average leading to more informative label feature representations.

4.2 A Probabilistic Regression Model

XReg is a regression method which takes a probabilistic approach to estimating the label relevance weights. Firstly, all the relevance weights are normalized to lie between 0 and 1 by dividing by its maximum value, thus allowing them to be treated as probability values. Note that while click-through rates in DSA are already valid probabilities, the inverse propensities and the user rating could exceed 1. Also, note that the predicted estimates can be easily scaled back since no information is lost due to this normalization.

XReg treats the normalized relevance weights for each label as the marginal probability of its relevance to a data point, which is, in fact, the case in DSA. This allows XReg to minimize the KL-divergence between the true and the predicted marginal probability for each label with respect to each data point. KL-divergence [31] measures how close 2 distributions are and is minimized when the 2 are identical, thus justifying its use while regressing on to probability values.

A naive 1-vs-All approach, which learns a separate regressor minimizing KL-divergences for each label, would be extremely costly to train when labels are in millions. To reduce this complexity, XReg leverages the previously trained label tree. XReg expresses the marginal probability of a label as the probability that a data point traverses the tree path starting from the root to the label. Let the path from root to label l consist of nodes $n_{l1}, \cdots, n_{lH}$ where H is tree height, n_{l1} is the root and n_{lH} is the leaf node containing solely label l. Let z_{lh} denote the probability that a data point $\mathbf{x}$ visits the node n_{lh} after it has already visited the parent $n_{l(h-1)}$. Then the true marginal probability y_l that the label l is relevant to $\mathbf{x}$ is equivalent to $y_l = \prod_{h=1}^{H} z_{lh}$. Similar equality holds for predicted marginal probability: $\hat{y}_l = \prod_{h=1}^{H} \hat{z}_{lh}$. XReg then learns to minimize an upper bound on the KL-divergence between the two according to the following theorem.

Theorem 4.1. *Given that $y_l = \prod_{h=1}^{H} z_{lh}$ and $\hat{y}_l = \prod_{h=1}^{H} \hat{z}_{lh}$ and under the standard unvisited node assumption of Parabel*

$$D_{KL}(y_l || \hat{y}_l) \leq \sum_{h=1}^{H} s_{lh} D_{KL}(z_{lh} || \hat{z}_{lh}) \text{ where } s_{lh} = \prod_{\tilde{h}=1}^{h} z_{l(\tilde{h}-1)} \tag{4}$$

Proof. Proof is provided in the supplementary. □

The unvisited node assumption formalizes the observation that the children of an unvisited internal node will never be traversed and that the labels in an unvisited leaf node will never be visited by a data point [44]. Due to the above theorem, XReg can separately minimize the KL-divergence over the true and predicted probabilities that a data point takes a particular edge in the tree, and still end up minimizing the KL-divergences over each of the individual marginal label probabilities. The true probability value of edge traversal z_{lh} is essentially the probability that the data point visits any of the labels in the subtree rooted at the node indexed lh. We instantiate it to be equal to the largest marginal probability of any label in the subtree, by assuming the worst-case scenario that labels in each subtree are fully correlated, which promotes model robustness.

The KL-divergence minimization is mathematically equivalent to training a logistic regressor for estimating z_{lh} values for each tree edge where every data point is duplicated with weights z_{lh}

and $1 - z_{lh}$:

$$\min_{\mathbf{w}_n} \|\mathbf{w}_n\|^2 + \frac{C}{|\mathcal{I}_n|} \sum_{i \in \mathcal{I}_n} \{s_{in} z_{in} \log(1 + \exp(-\mathbf{w}_n^\top x_i)) + \quad (5)$$

$$s_{in}(1 - z_{in}) \log(1 + \exp(+\mathbf{w}_n^\top x_i))\} \quad (6)$$

where n is used to index the node instead of lh, $\mathcal{I}_n$ only include those points which reach the node n. The problem in (Eq. 5) is strongly convex and was optimized using the modified CDDual algorithm available from Liblinear package [17]. To summarize, each internal node in XReg contains 2 1-vs-All regressors which give the probability that a data point traverses to each of its children, each leaf node contains M 1-vs-All regressors which gives the conditional probability of each label being relevant given the data point reaches its leaf.

We make a mild assumption that each data point has at most $O(\log L)$ positive labels is made which is often valid on extreme learning datasets. As a result, each data point traverses at most $O(\log^2 L)$ tree edges, which directly leads to a huge reduction in training complexity thus resulting in $O(N\hat{D} \log^2 L)$ where $\hat{D}$ is the average number of non-zero features per data point. The following lemma describes how XReg's training objective is related to the XMAD@k metric proposed earlier:

Lemma 4.2. *XReg's overall training objective minimizes an upper bound over XMAD@k for all k, with the bound being tighter for smaller k values.*

Proof. The proof is provided in the supplementary. □

4.3 Pointwise Inference

The pointwise inference algorithm in XReg utilizes the same beam search prediction technique proposed in Parabel where only the top ranked relevant labels are recommended based on a greedy, breadth-first tree traversal strategy. The following theorem proves that such traversal mechanism is not only asymptotically optimal for both WP@k and XMAD@k but also strongly generalizable with $O(polylog(L))$ sample complexity. This uses the assumption that each data point has at most $O(\log L)$ positive labels. Also the theorem assumes that each individual regressor in well-generalizable and achieves zero-regret with infinite data samples.

Theorem 4.3. *When each data point has at most $O(\log L)$ positive labels, the expected WP@k regret and XMAD@k error suffered by XReg's pointwise inference algorithm are bounded by:*

$$O(\log^2 L \sqrt{\frac{W}{\sqrt{Np}}} \sqrt{1 + \sqrt{5 \log \frac{3L}{\delta}}})$$

with probability at least $1 - \delta$, where N is the total training points, L is the number of labels, W is the maximum norm across all node classifier vectors and p is the minimum probability density of $\mathbf{x}$ distribution that any tree node receives.

Proof is available in the supplementary. Therefore the errors go to 0 as $N \to \infty$. The $\log^2 L$ dependence arises because each data point visits at most $\log^2 L$ nodes in a tree.

4.4 Labelwise Inference

The XReg model also allows efficient labelwise inference. The core idea here is to estimate from training data the fraction of points with non-zero relevance that visit each node of the tree and allot a factor F times the same fraction of top ranking test points to respective nodes. On large scale datasets with enough training and test points, the ratio of non-zero relevance points in each tree node remain almost the same over training and test points. The factor accounts for any small deviations. This strategy is adopted to ensure that all non-zero relevance points for a label end up reaching the label's leaf node. Finally, the topmost scoring test points that visit a label's leaf node are ranked at the top for that label, where the scores are marginal relevance probabilities, the average test time complexity is $O(F \log^2 L)$ per test point. Pseudocode for labelwise inference is provided in the supplementary.

5 EXPERIMENTS

Datasets: Experiments were carried out on several medium and large scale benchmark datasets with up to 4.9M training points, 1.8M features and 1.4M labels (see Table 1 for dataset statistics). These datasets cover diverse applications such as document tagging (BibTeX [45], EURLex-4K [38], Wiki10-31K [8] & WikiLSHTC-325K [9, 42]), content-based movie recommendation (YahooMovie-8K [3] & MovieLens-138K [2, 19]), item-to-item recommendation of Amazon products (Amazon-670K [9, 37]), sponsored search advertising (SSA-130K) and dynamic search advertising (DSA-130K, DSA-1M). For ease of discussion, the label size suffixes are dropped from dataset names hereafter except for DSA. The document tagging, item-to-item recommendation, and SSA datasets require pointwise inference whereas the movie recommendation and DSA datasets require labelwise inference. YahooMovie and MovieLens use normalized (between 0 and 1) user-provided movie ratings as relevance weights and movie meta-data like summary, genres, and tags as features. For all the datasets, bag-of-words feature representation derived from text descriptions are used. SSA and DSA are proprietary datasets that were created by mining the Bing logs. Rest of the datasets are available from [1].

Baselines: XReg was compared to leading extreme classifiers such as PfastreXML [25], Parabel [44], DiSMEC [5] and ProXML [6], traditional multivariate regressors such as one-vs-all least-squares regression (1-vs-all-LS) and LEML [65], and a popular pairwise ranker, RankSVM [20, 33]. XReg was also compared to the recent eXtreme Learning to Rank (XLR) [12] approach. ProXML is the current state-of-the-art over propensity scored precision@k (PSP-p@k) during pointwise inference. Results for DiSMEC and ProXML, which

Table 1: Dataset statistics

Dataset	Train N	Features D	Labels L	Test N'	Avg. labels per point	Avg. points per label
BibTeX	4,880	1,836	159	2,515	2.40	111.71
EURLex-4K	15,539	5,000	3,993	3,809	5.31	448.57
Wiki10-31K	14,146	101,938	30,938	6,616	18.64	8.52
SSA-130K	122,462	152,192	130,515	54,773	5.60	7.60
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
YahooMovie-8K	8,341	28,978	7,642	3,574	18.57	28.96
DSA-130K	122,462	152,192	130,515	54,773	5.60	7.60
MovieLens-138K	18,732	19,924	138,490	8,012	527.31	101.83
DSA-1M	4,914,640	1,840,877	1,453,150	2,106,273	0.23	7.80

Table 2: XReg achieves the best or close to the best ranking and regression performance in both pointwise ("-p") and labelwise ("-l") prediction settings. Re-ranking with tail classifiers (XReg-t) further improves the performance in many cases. More results are in the supplementary.

Method	PSP-p@5 (%)	Tau-p@5 (%)	XMAD-p@5	Training time (hrs)	Test time /point (ms)	Model size (GB)
BibTex						
PfastreXML	59.75	53.68	**0.3151**	0.0050	0.2348	0.0246
Parabel	57.36	51.48	0.3372	0.0015	0.1945	0.0035
LEML	56.42	51.58	0.3520	0.0229	0.1737	0.0032
1-vs-all-LS	**60.14**	**54.21**	0.3337	**0.0007**	0.1137	0.0023
RankSVM	59.12	52.58	0.7089	0.0015	**0.0719**	0.0023
DiSMEC	57.23	51.47	0.3371	0.0004	0.0951	**0.0012**
ProXML	58.30	-	-	-	-	-
XReg	58.61	52.35	0.3158	0.0035	0.1642	0.0030
XReg-t	58.77	52.46	0.3386	0.0025	0.1256	0.0043
EURLex-4K						
PfastreXML	45.17	48.85	0.1900	0.0887	1.3891	0.2265
Parabel	48.29	50.75	0.4227	**0.0245**	**1.1815**	0.0258
LEML	32.30	37.24	0.2115	0.3592	4.4483	0.0281
1-vs-all-LS	**52.27**	**53.96**	**0.1744**	0.1530	4.5378	0.1515
RankSVM	46.70	51.43	1.1967	0.1834	4.7635	0.1470
DiSMEC	50.62	52.33	0.4308	0.0999	1.9489	**0.0072**
ProXML	51.00	-	-	-	-	-
XReg	49.72	52.86	0.1849	0.0642	1.2899	0.0378
XReg-t	50.40	53.45	0.2132	0.0544	1.2074	0.0692
Wiki10-31K						
PfastreXML	15.91	20.29	0.5705	0.3491	11.6855	0.5466
Parabel	13.68	19.83	0.7085	**0.3204**	**3.7275**	0.1799
LEML	13.05	20.06	0.5716	0.9546	54.9470	0.5275
1-vs-all-LS	21.89	26.71	**0.5459**	2.4341	129.8342	16.9871
RankSVM	18.46	25.84	1.2236	4.9631	92.2684	10.8536
DiSMEC	15.61	22.43	0.7140	2.1945	13.8993	**0.0290**
XReg	16.94	24.97	0.5716	0.6184	3.7649	0.3218
XReg-t	**22.60**	**30.55**	0.5506	0.6431	5.4910	0.9026
WikiLSHTC-325K						
PfastreXML	28.04	36.38	0.1437	7.1974	6.9045	13.3096
Parabel	37.22	41.71	0.2459	**1.2195**	**2.2486**	**3.0885**
DiSMEC	39.50	-	-	-	-	-
ProXML	**41.00**	-	-	-	-	-
XReg	36.92	41.62	**0.1411**	4.5119	3.0312	3.5105
XReg-t	40.33	**43.39**	0.3140	3.8552	3.0896	4.1955
Amazon-670K						
PfastreXML	28.53	30.97	0.4019	3.3143	11.4931	9.8113
Parabel	32.88	31.32	0.4292	**0.5815**	2.3419	**1.9297**
DiSMEC	34.45	31.94	0.4275	373	1414	3.7500
ProXML	**35.10**	-	-	≈1200	≈1000	-
XReg	33.24	34.72	**0.3869**	1.4925	2.4633	3.4186
XReg-t	34.29	**35.83**	0.4473	1.1864	**2.2242**	4.5952

Method	CTR-p@5 (%)	Tau-p@5 (%)	XMAD-p@5	Training time (hrs)	Test time /point (ms)	Model size (GB)
SSA-130K						
PfastreXML	27.79	23.77	0.0655	1.3765	5.2419	1.6258
Parabel	**32.97**	**30.25**	0.1430	**0.2283**	1.9098	**0.3625**
LEML	6.54	8.10	**0.0654**	8.3253	161.6891	1.1308
RankSVM	13.06	14.03	2.7871	9.6026	130.0945	7.4834
DiSMEC	32.75	29.16	0.1562	31.4358	61.0967	0.0802
XReg	32.39	28.27	0.0684	0.4570	7.4715	0.7871
XReg-t	32.81	28.73	0.1131	0.5049	**1.7746**	1.4156

Method	Rating-l@5 (%)	Tau-l@5 (%)	XMAD-l@5	Training time (hrs)	Test time /point (ms)	Model size (GB)
YahooMovie-8K						
PfastreXML	10.18	19.72	0.6286	**0.0241**	8.5074	0.0753
Parabel	9.73	28.22	0.6284	0.0299	**0.9639**	0.1307
LEML	21.79	28.85	0.6408	0.0593	5.3650	0.0586
1-vs-all-LS	21.63	31.24	0.6269	0.0740	6.8841	1.6977
RankSVM	24.88	33.28	1.0579	0.1282	5.1620	0.7172
DiSMEC	24.53	32.75	0.6207	0.0337	3.4258	0.0376
XLR	4.66	10.72	0.6716	-	4.7724	**0.0293**
XReg	25.86	35.00	0.6248	0.0685	4.1965	0.2829
XReg-t	**26.05**	**35.33**	**0.6185**	0.0615	3.6353	0.4500
MovieLens-138K						
PfastreXML	7.25	22.84	0.9199	0.4514	19.8270	0.1837
Parabel	3.51	37.80	0.9200	1.7790	**1.6132**	3.4322
LEML	43.19	64.78	0.8722	**0.4186**	91.4262	0.2535
1-vs-all-LS	42.16	63.92	0.8832	2.5756	121.6169	16.1334
DiSMEC	45.35	61.55	0.8857	1.5437	74.9537	1.0514
XLR	9.67	21.42	0.9134	4.579	68.347	**0.0634**
XReg	48.94	66.99	0.8741	2.6287	7.7996	3.6223
XReg-t	**49.29**	**67.36**	**0.8285**	2.7437	9.8279	4.8958

Method	CTR-l@5 (%)	Tau-l@5 (%)	XMAD-l@5	Training time (hrs)	Test time /point (ms)	Model size (GB)
DSA-130K						
PfastreXML	28.18	**34.75**	0.0422	1.3765	5.2419	1.6258
Parabel	33.97	28.37	0.0891	**0.2283**	**1.9098**	0.3625
LEML	10.36	7.70	**0.0415**	8.3253	212.1707	1.1308
DiSMEC	34.06	27.96	0.1039	31.4358	55.4037	0.0802
XLR	0.09	0.10	0.4816	5.5430	64.1134	**0.0678**
XReg	35.66	28.51	0.0439	0.4570	7.4715	0.7871
XReg-t	**36.32**	28.45	0.0587	0.3669	8.4376	1.3512
DSA-1M						
Parabel	37.95	30.93	0.1004	**9.2800**	**2.5031**	**5.6774**
XReg	37.57	31.09	**0.0563**	20.7463	3.1792	11.0178
XReg-t	**38.81**	**31.41**	0.0714	15.4201	3.4036	18.7434

required 1000s of cores, could not be replicated on large datasets and hence the numbers from the corresponding papers have been reported directly. RankSVM was unable to scale to datasets larger than SSA-130K and hence required down-sampling of negatives up to 0.1% on these larger datasets. XLR, which specifically addresses the labelwise recommendation task, has only been applied to labelwise datasets. For the other baselines, results have been reported for only those datasets up to which the implementations scale. Since many of these large-scale datasets have a preponderance of tail labels, results for a variant of XReg where predicted labels have been reranked with tail classifier scores have also been reported with a "-t" suffix. The tail classifiers are generative classifiers which are tailored for accurate predictions on labels with < 5 training point samples [25]. For extreme classifiers which train on binary labels (Parabel, DiSMEC, and ProXML), all positive relevance weights were approximated to be fully relevant (value 1). Remaining baselines, including the PfastreXML and LEML, were trained on relevance weighted labels for a fair comparison.

Hyperparameters: XReg has 5 hyperparameters: (a) number of label trees in the ensemble (T); (b) number of tree paths explored by a test point during pointwise prediction (P); (c) maximum ratio of test to train points that traverse to each node during labelwise prediction (F); (d) maximum number of labels in a leaf node of XReg tree (M); and (e) regularization parameter common to logistic regressors in all the internal and leaf node classifiers (C). On medium-sized datasets, the XReg's hyperparameters were set by fine-grained tuning over a 10% validation dataset. On larger datasets where tuning was not feasible, the default hyperparameter setting of $T = 3$, $P = 10$, $F = 4$, $M = 100$ and $C = 10$ was used. Results in table 8 of the supplementary demonstrates that the above default values of T, P, M achieve the best trade-off between accuracy and

scalability across multiple datasets and increasing any of them further leads to minimal gains in accuracy while linearly increasing the training or prediction cost. The value of α, which adjusts the influence of tail classifiers in XReg-t, was also tuned on the validation data. The hyperparameters for baseline algorithms were also set by tuning on medium datasets and set to defaults suggested in the respective papers/codebases on larger datasets.

Metrics and hardware: Performances were evaluated using accuracy metrics such as WP@k variants, Tau@k and XMAD@k (see Section 3) as well as efficiency metrics such as training time, test time per data point and model size. Among WP@k variants, for tagging (BibTeX, EURLex, Wiki10, WikiLSHTC) and Amazon datasets PSP@k are reported; for SSA and DSA which are ads datasets CTR@k is reported; and for movie recommendation datasets (YahooMovie and MovieLens) Rating@k is reported. All accuracy metrics are suffixed with "-p" or "-l" depending on whether the prediction scenario is pointwise or labelwise. All experiments were run on an Intel Xeon 2.5 GHz processor with 256 GB RAM.

Results on benchmark datasets: Table 2 compares XReg's performance to diverse baselines on datasets belonging to tagging, recommendation and DSA applications. In terms of prediction accuracy, XReg consistently achieves close to best performance in terms of WP@5, Tau@5 as well as XMAD@5 metrics. In particular, XReg can be up to 2.4%, 3.89% and 2x better than all baselines in WP@5, Tau@5 and XMAD@5 respectively.

On most tagging datasets, XReg scores within 2% of the state-of-the-art ProXML in terms of the popular PSP@5 metric but can be up to 1000x faster during both training and prediction.

XReg consistently outperforms extreme classifiers like Parabel and DiSMEC which train only on binary labels. In particular, XReg can be up to 9% and 45% better than Parabel over pointwise and labelwise datasets in terms of WP@5. The larger gains on labelwise datasets are due to pointwise prediction in Parabel which can lead to low label coverage, especially on datasets like MovieLens with only 8K test points but around 140K labels. Owing to similar classifier architectures, XReg can be highly efficient just like Parabel. XReg is at most 3.75x and 4.8x slower during training and prediction and has at most 2.15x the model size as Parabel.

Owing to their high scalability, both Parabel and XReg scale to the largest DSA-1M dataset where none of the other approaches scale. On this dataset, XReg has 50% smaller XMAD@5 than Parabel.

XReg-t denotes the re-ranked XReg where the predicted relevance estimates are combined with tail classifier scores to improve ranking performance over more informative tail labels. XReg-t consistently improves performance over XReg since most XR datasets are dominated by tail labels. XReg-t can be up to 5.66% and 5.58% better than XReg in terms of PSP@5 and Tau@5. However, XReg-t often increases XMAD@5 over XReg since tail classifiers are not regressors but are good generative classifiers which and therefore increase regression errors. Since the tail classifiers are extremely efficient to train and the re-ranking step is only applied to a small number (100s) of labels with high relevance estimates from XReg, XReg-t can be very efficient with 1.1, 1.96 and 2.8 times the training time, prediction time and model size as XReg in worst case.

Additional results for WP@k, Tau@k where k=1,3, nDCG@5 and XRMSE@5 are available in the supplementary.

Filtering and re-ranking: The accurate relevance weight estimates that XReg produces can be used for many downstream tasks such as filtering and re-ranking as discussed in Sections 1 and 3. Table 3 reports (1) AUPRC which measures the quality of filtering and (2) WP-rerank@5 which measures the quality of reranking with tail classifiers by using the relevance estimates generated by (a) Parabel, (b) XReg and (c) XReg-zero which corrupts XReg's estimates by setting all relevances to almost 0 while maintaining the same rankings. As can be seen, XReg consistently outperforms Parabel and XReg-zero, both of which have higher regression errors as measured by XMAD@5, during both filtering and re-ranking. XReg-zero's results demonstrate that just accurate ranking, as measured by the WP@5 column, is not enough for good filtering and re-ranking performance and that low regression errors are also necessary. Furthermore, regression errors measured in terms of traditional MAD are not reliable since MAD is sensitive to the sparsity in relevances and can in fact be lower for corrupted relevances such as in XReg-zero. Figures showing the AUPRC plots can be found in supplementary.

Analysis of ranking errors and regression metrics: Table 4 presents the relationship between XMAD & MAD to the ranking error (WP-regret@k). Table 4 shows that, across all the baselines, 2*XMAD@2k is a much better upper bound for WP-regret@k compared to the traditional MAD. Particularly, on regression and classification techniques, 2*XMAD@2k is 1.35-5.84 times the WP-regret@k while MAD can be up to 69x larger than 2*XMAD@2k. In general, ranking baselines (RankSVM, XLR) do not produce good regression values making the ratio of 2*XMAD@2k and MAD to WP-regret@k much higher. Lastly, for the dense score prediction algorithms like 1-vs-all-LS, MAD is significantly high since it sums up the errors across all labels.

Ablation Studies: To test the effectiveness of the proposed XReg along with its novel labelwise prediction algorithm, experiments were done to show the boost due to each of the factors. First, the extension of Parabel-logloss to utilize label weights lead to pointwise XReg which improved the ranking metrics up to 1.5% over Parabel across the 3 labelwise datasets showing that XReg can learn better from relevance weights. Further, when XReg was coupled with the novel labelwise prediction algorithm, the gains were up to 16%, 1.1% and 45% on YahooMovie-8K, DSA-130K, and MovieLens-138K respectively due to higher label coverage. Lastly, the use of tail classifiers with XReg (XReg-t) further increased the ranking performance by up to 0.7% over labelwise XReg.

DSA Results: Table 2 shows the offline evaluation on DSA-130K and DSA-1M while Table 6 showcases the results of the live deployment of labelwise XReg in Bing DSA pipeline. Even though few of the extreme classification techniques could scale to DSA-130K, the live deployment requires the techniques to scale to tens of millions of labels (queries) and data points (ads). In the actual deployment only PfastreXML, Parabel, and XReg were able to scale.

Table 6 compares XReg's performance to the existing DSA ensemble, consisting of BM25 information retrieval based algorithm [28] and PfastreXML when deployed on Bing. Both pointwise and labelwise XReg were deployed and evaluated. Pointwise XReg increased RPM, CY, and IY by 5% while maintaining the CTR and BR. Finally, the labelwise XReg boosts the revenue by 58%, improves query

Table 3: XMAD@k is a better indicator of the filtering and re-ranking qualities than purely ranking metrics like WP@k or traditional regression metrics like MAD.

Method	AUPRC	WP-rerank-p @5 (%)	XMAD-p@5	MAD	WP-p @5 (%)
EURLex-4K					
Parabel	0.092	49.67	0.4227	3.96	48.29
XReg	**0.117**	**50.39**	**0.1849**	1.22	**49.72**
XReg-zero	0.085	50.12	0.2255	**1.21**	**49.72**
Wiki10-31K					
Parabel	0.036	21.14	0.7084	10.15	13.67
XReg	**0.046**	**22.60**	**0.5716**	6.01	**16.94**
XReg-zero	0.036	19.20	0.5781	**5.61**	**16.94**

Method	AUPRC	WP-rerank-l @5 (%)	XMAD-p@5	MAD	WP-l @5 (%)
YahooMovie-8K					
Parabel	0.135	10.09	0.6283	6.39	9.72
XReg	**0.175**	**26.03**	**0.6248**	6.78	**25.85**
XReg-zero	0.076	25.93	0.6306	**5.83**	**25.85**
DSA-130K					
Parabel	0.016	34.59	0.0890	0.88	33.97
XReg	**0.035**	**36.32**	0.0438	0.24	**35.65**
XReg-zero	0.010	35.90	**0.0402**	**0.20**	**35.65**

Table 4: Ranking regret at k is up to 69x more closely bounded by 2*XMAD@2k compared to the traditional MAD as proposed in Section 3. $k = 5$, "-p": pointwise, "-l": labelwise and "-t": use of tail classifiers. Please refer to the text for details.

Method	WP-regret-p @k	2*XMAD-p @2k	MAD	2*XMAD-p@2k / WP-regret-l@k	MAD / WP-regret-p@k
EURLex-4K					
PfastreXML	0.1237	0.2481	1.8667	2.01	15.09
Parabel	0.1166	0.5696	3.9622	4.89	33.98
LEML	0.1527	0.2739	2.0779	**1.79**	13.61
1-vs-all-LS	0.1076	0.2468	2.696	2.29	25.06
RankSVM	0.1202	2.1303	37.7822	17.72	314.33
DiSMEC	0.1107	0.5733	4.9883	5.18	45.06
XReg	0.1134	**0.2432**	**1.2284**	2.14	**10.83**
XReg-t	0.1119	0.3141	3.4887	2.81	31.18
Wiki10-31K					
PfastreXML	0.4861	0.8862	9.3561	1.82	19.25
Parabel	0.4990	1.1784	10.1523	2.36	20.35
LEML	0.5027	0.8886	25.5622	**1.77**	50.85
1-vs-all-LS	0.4515	**0.8492**	34.1409	1.88	75.62
RankSVM	0.4714	2.2960	75.4734	4.87	160.10
DiSMEC	0.4878	1.1519	80.3084	2.36	164.63
XReg	0.4802	0.8938	**6.0104**	1.86	**12.52**
XReg-t	**0.4475**	0.8741	32.2013	1.95	71.96

Method	WP-regret-l @k	2*XMAD-l @2k	MAD	2*XMAD-l@2k / WP-regret-l@k	MAD / WP-regret-l@k
YahooMovie-8K					
PfastreXML	0.5665	0.8875	8.4893	1.57	14.99
Parabel	0.5693	0.8850	**6.3913**	**1.55**	**11.23**
LEML	0.4933	0.9571	36.0738	1.94	73.13
1-vs-all-LS	0.4943	0.9288	47.1887	1.88	95.47
RankSVM	0.4738	2.0235	76.6099	4.27	161.69
DiSMEC	0.4760	0.9003	38.0254	1.89	79.89
XLR	0.6013	1.0423	17.0795	1.73	28.40
XReg	0.4676	**0.8847**	6.7809	1.89	14.50
XReg-t	**0.4664**	0.8964	12.1071	1.92	25.96
DSA-130K					
PfastreXML	0.0289	0.0500	0.3291	1.73	11.39
Parabel	0.0266	0.1230	0.8827	4.62	33.18
LEML	0.0361	**0.0486**	0.2828	**1.35**	7.83
DiSMEC	0.0265	0.1547	4.0878	5.84	154.26
XLR	0.0402	0.9157	34.429	22.78	856.44
XReg	0.0259	0.0519	**0.2482**	2.00	**9.58**
XReg-t	**0.0256**	0.0787	0.5862	3.07	22.90

Table 5: The ablation study of Parabel leading to per-label XReg-t which clearly outperforms its predecessors on ranking metrics.

Method	Rating-l@5 (%)	AUC-l@5 (%)	CTR-l@5 (%)	AUC-l@5 (%)	Rating-l@5 (%)	AUC-l@5 (%)
	YahooMovie-8K		DSA-130K		MovieLens-138K	
Parabel-logloss	8.99	32.00	33.58	28.46	2.36	48.20
Pointwise XReg	9.47	34.00	34.59	27.13	3.89	52.35
Labelwise XReg	25.86	35.00	35.66	**28.51**	48.94	66.99
Labelwise XReg-t	**26.05**	**35.33**	**36.32**	28.45	**49.29**	**67.36**

Table 6: XReg significantly improves query coverage and revenue over the existing ensemble for DSA on Bing. *Note*: RPM: Revenue Per Million impressions, Cov: Query Coverage, CY: Click Yield, IY: Impression Yield, BR: Bounce Rate

Method	Relative RPM (%)	Relative Cov (%)	Relative CY (%)	Relative IY (%)	Relative CTR (%)	Relative BR (%)
Pointwise XReg	106	-	105	105	100	100
Labelwise XReg	**158**	**127**	**148**	**150**	98	100

coverage by 27% along with a 48% and 50% increase in click yield and impression yields at a cost of only 2% reduction in CTR.

6 CONCLUSIONS

This paper proposed a new learning paradigm called eXtreme Regression (XR) which provides a scalable solution to many real-world recommendation and ranking problems such as tagging, recommendation, DSA *etc.* XR involves learning to accurately predict the numerical relevance weights of an extremely large number of labels with respect to a data point. These weights not only induce an accurate ranking but are also useful for subsequent filtering and re-ranking steps. To effectively solve XR problems, this paper also develops a new evaluation metric called XMAD@k and a new algorithm called XReg. XReg consistently outperforms the state-of-the-art extreme classifiers as well as large-scale regressors and rankers in terms of ranking accuracies and efficiently scales to datasets with millions of data points and labels. Deployment of XReg on DSA in Bing resulted in a relative gain of 58% in revenue and 27% in query coverage.

7 ACKNOWLEDGEMENTS

We are grateful to Kunal Dahiya, Prateek Jain, Nagarajan Natarajan, Deepak Saini and Harsha Vardhan Simhadri for helpful discussions, feedback and computing resources.

REFERENCES

[1] [n. d.]. Code and datasets for XReg. http://manikvarma.org/code/XReg/download.html
[2] [n. d.]. MovieLens 20M Dataset. https://grouplens.org/datasets/movielens/20m/.
[3] [n. d.]. Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0. https://webscope.sandbox.yahoo.com/catalog.php?datatype=r,.
[4] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*.
[5] R. Babbar and B. Schölkopf. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In *WSDM*.
[6] R. Babbar and B. Schölkopf. 2018. Adversarial Extreme Multi-label Classification. *arXiv preprint arXiv:1803.01570* (2018).
[7] R. Bekkerman, M. Bilenko, and J. Langford. 2011. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press.
[8] K. Bhatia, K. Dahiya, H. Jain, Y. Prabhu, and M. Varma. 2019. The Extreme Classification Repository: Multi-label Datasets & Code. http://manikvarma.org/downloads/XC/XMLRepository.html
[9] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. 2015. Sparse Local Embeddings for Extreme Multi-label Classification. In *NeurIPS*.
[10] T. Chai and R. R. Draxler. 2014. Root mean square error (RMSE) or mean absolute error (MAE).
[11] Y. Chen and H. Lin. 2012. Feature-aware label space dimension reduction for multi-label classification. In *NeurIPS*.
[12] Minhao Cheng, Ian Davidson, and Cho-Jui Hsieh. 2018. Extreme Learning to Rank via Low Rank Assumption. In *ICML*.
[13] Y. Choi, M. Fontoura, E. Gabrilovich, V. Josifovski, M. R. Mediano, and B. Pang. 2010. Using landing pages for sponsored search ad selection. In *WWW*.
[14] M. Cissé, N. Usunier, T. Artières, and P. Gallinari. 2013. Robust Bloom Filters for Large MultiLabel Classification Tasks. In *NeurIPS*.
[15] C. Dhanjal, R. Gaudel, and S. Clémençon. 2015. Collaborative filtering with localised ranking. In *AAAI*.
[16] J Engel. 1988. Polytomous logistic regression. *Statistica Neerlandica* (1988).
[17] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR* (2008).
[18] C. Guo, A. Mousavi, X. Wu, D. N. Holtmann-Rice, S. Kale, S. Reddi, and S. Kumar. 2019. Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces. In *NeurIPS*.
[19] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* (2015), 19:1–19:19.
[20] R. Herbrich, T. Graepel, and K. Obermayer. 1999. Support vector learning for ordinal regression. (1999).
[21] D. Hsu, S. Kakade, J. Langford, and T. Zhang. 2009. Multi-Label Prediction via Compressed Sensing. In *NeurIPS*.
[22] J. Hu and P. Li. 2018. Collaborative Multi-objective Ranking. In *PCIKM*.
[23] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.
[24] H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. 2019. Slice: Scalable Linear Extreme Classifiers trained on 100 Million Labels for Related Searches. In *WSDM*.
[25] H. Jain, Y. Prabhu, and M. Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. In *KDD*.
[26] K. Järvelin and J. Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002).
[27] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. 2016. Extreme F-measure Maximization Using Sparse Probability Estimates. In *ICML*.
[28] K. S. Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.* (2000).
[29] M. G. Kendall. 1938. A new measure of rank correlation. *Biometrika* 30 (1938).
[30] D. Kocev, C. Vens, J. Struyf, and S. Dzeroski. 2007. Ensembles of multi-objective decision trees. In *ECML*.
[31] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
[32] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma. 2018. FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network.. In *NeurIPS*.
[33] C. Lee and C. Lin. 2014. Large-scale linear ranksvm. *Neural computation* 26, 4 (2014), 781–817.
[34] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. 2014. Local collaborative ranking. In *WWW*.
[35] Z. Lin, G. Ding, M. Hu, and J. Wang. 2014. Multi-label Classification via Feature-aware Implicit Label Space Encoding. In *ICML*.
[36] J. Liu, W. Chang, Y. Wu, and Y. Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *SIGIR*.
[37] J. McAuley and J. Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.
[38] E. L. Mencia and J. Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML*.
[39] P. Mineiro and N. Karampatziakis. 2015. Fast Label Embeddings for Extremely Large Output Spaces. In *ECML*.
[40] A. Niculescu-Mizil and E. Abbasnejad. 2017. Label 's for Large Scale Multilabel Classification. In *AISTATS*.
[41] D. Park, J. Neeman, J. Zhang, S. Sanghavi, and I. S. Dhillon. 2015. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *ICML*.
[42] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artières, G. Paliouras, É. Gaussier, I. Androutsopoulos, M. R. Amini, and P. Gallinari. 2015. LSHTC: A Benchmark for Large-Scale Text Classification. (2015). http://arxiv.org/abs/1503.08581
[43] Y. Prabhu, A. Kag, S. Gopinath, K. Dahiya, S. Harsola, R. Agrawal, and M. Varma. 2018. Extreme multi-label learning with label features for warm-start tagging, ranking and recommendation. In *WSDM*.
[44] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*.
[45] Y. Prabhu and M. Varma. 2014. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*.
[46] B. Pradel, N. Usunier, and P. Gallinari. 2012. Ranking With Non-Random Missing Ratings: Influence Of Popularity And Positivity on Evaluation Metrics. In *RecSys*.
[47] S. Ravi, A. Z. Broder, E. Gabrilovich, V. Josifovski, S. Pandey, and B. Pang. 2010. Automatic generation of bid phrases for online advertising. In *WSDM*.
[48] D. Sculley. 2009. Large scale learning to rank. (2009).
[49] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*.
[50] S. Si, H. Zhang, S. S. Keerthi, D. Mahajan, I. S. Dhillon, and C. J. Hsieh. 2017. Gradient Boosted Decision Trees for High Dimensional Sparse Output. In *ICML*.
[51] A. J. Smola and B. Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing* (2004).
[52] Y Tagami. 2017. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. In *KDD*.
[53] Y. Wang, L. Wang, Y. Li, D. He, and T. Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *COLT*.
[54] G. S. Watson et al. 1967. Linear least squares regression. *The Annals of Mathematical Statistics* 38, 6 (1967), 1679–1699.
[55] X. Wei and W. B. Croft. 2006. LDA-based document models for ad-hoc retrieval. In *SIGIR*.
[56] J. Weston, S. Bengio, and N. Usunier. 2011. Wsabie: Scaling Up To Large Vocabulary Image Annotation. In *IJCAI*.
[57] J. Weston, A. Makadia, and H. Yee. 2013. Label Partitioning For Sublinear Ranking. In *ICML*.
[58] L. Wu, C. Hsieh, and J. Sharpnack. 2018. SQL-Rank: A Listwise Approach to Collaborative Ranking. In *ICML*.
[59] C. Xu, D. Tao, and C. Xu. 2016. Robust Extreme Multi-label Learning. In *KDD*.
[60] I. E. H. Yen, X. Huang, W. Dai, P. Ravikumar, I. Dhillon, and E. Xing. 2017. PPDsparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *KDD*.
[61] I. E. H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. S. Dhillon. 2016. PD-Sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*.
[62] W. T. Yih, J. Goodman, and V. R. Carvalho. 2006. Finding advertising keywords on web pages. In *WWW*.
[63] D. Yin, Y. Hu, J. Tang, T. Daly, M. Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, H. Deng, C. Nobata, et al. 2016. Ranking relevance in yahoo search. In *PKDD*.
[64] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu. 2019. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *NeurIPS*.
[65] H. F. Yu, P. Jain, P. Kar, and I. S. Dhillon. 2014. Large-scale Multi-label Learning with Missing Labels. In *ICML*.
[66] W. Zhang, D. Wang, G. Xue, and H. Zha. 2012. Advertising Keywords Recommendation for Short-Text Web Pages Using Wikipedia. *ACM TIST* (2012).
[67] M. Zhu. 2004. Recall, Precision and Average Precision.

Algorithm 1 XReg Labelwise Prediction

Input:
Test data points $\{\mathbf{x}_i\}_{i=1}^{M}$
Trained tree $\mathcal{T}$
Required no. of most relevant test points per label N
Fraction of test points relevant for each node $\{frac(n)\}_{n=1}^{|nodes(\mathcal{T})|}$
Multiplicative factor F # $F * frac(n)$ most relevant test points are passed down to node n
Output:
Predicted test points for each label $\{pred(l)\}_{l=1}^{L}$
Initialize:
$points(1) \leftarrow \{1, \cdots, M\}$ # $points(n)$ is set of test points passed to node n
$\hat{z}_{i1} = 1.0 \ \forall i \in \{1, \cdots, M\}$ # All test points visit the root node with probability 1
for $n \in \{1, \cdots, |nodes(\mathcal{T})|\}$ **do** # Breadth-first exploration of the tree
  if $n \in internalnodes(\mathcal{T})$ **then**
    for $n' \in children(n)$ **do** # Iterate over children nodes of n
      for $i \in points(n)$ **do**
        $\hat{z}_{in'} \leftarrow \hat{z}_{in} * Sigmoid(\mathbf{w}_{n'}^{\top}\mathbf{x}_i)$ # $Sigmoid(x) = \frac{1}{1+\exp(-x)}$
      end for
      $points(n') \leftarrow$ RETAIN TOP$(\{\hat{z}_{in'}\}_{i \in points(n)}, F * frac(n'))$
    end for
  else if $n \in leafnodes(\mathcal{T})$ **then**
    for $l \in labels(n)$ **do** # Iterate over labels in leaf node n
      for $i \in points(n)$ **do**
        $\hat{y}_{il} \leftarrow \hat{z}_{in} * Sigmoid(\mathbf{w}_l \top \mathbf{x}_i)$
      end for
      $pred(l) \leftarrow$ RETAIN TOP$(\{\hat{y}_{il}\}_{i \in points(n)}, N)$
    end for
  end if
end for
Note: In case there are multiple trees in ensemble, the probability predictions estimated by all trees are averaged for a each test point, label pair and top N test points are outputted for each label
return $\{pred(l)\}_{l=1}^{L}$

procedure RETAIN TOP$(\{s_i\}_{i=1}^{I}, N)$
  $R \leftarrow$ argsort$(\{s_i\}_{i=1}^{I}, \text{comparator} \leftarrow s_{i_1} > s_{i_2})$ # Sort the test points in decreasing order of node or label probabilities
  $\mathcal{B} \leftarrow \{R[1], .., R[N]\}$
return $\mathcal{B}$
end procedure

A THEOREMS AND PROOFS

Lemma A.1. *Given any true and predicted relevance weight vectors $\mathbf{y}, \hat{\mathbf{y}} \in [0, \infty)^L$, the following inequality hold true:*

$$0 \leq \frac{M}{2} \leq XMAD@2k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \leq XRMSE@2k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \quad (7)$$

$$with,\ M = max(Ranking\text{-}error@k, Regression\text{-}error@k) \quad (8)$$

Proof. The ranking and regression errors are defined as follows

$$\text{Ranking-error@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} y_{il} - \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} y_{il} \quad (9)$$

$$\text{Regression-error@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} |\hat{y}_{il} - y_{il}| \quad (10)$$

Since $S(\mathbf{y}_i, k)$ picks the k largest values of y_{il}, Ranking-error@$k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \geq 0$ always. Due to summation over only non-negative values, Regression-error@$k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \geq 0$ always too, which together establish the inequality $0 \leq \frac{M}{2}$.

Now, let's prove that $\frac{M}{2} \leq$ XMAD@$2k(\hat{\mathbf{y}}, \mathbf{y})$. First, we begin by showing that Ranking-error@$k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \leq$ 2XMAD@$2k(\hat{\mathbf{y}}, \mathbf{y})$. Without loss of generality, let's assume that the sets $S(\mathbf{y}_i, k)$ and $S(\hat{\mathbf{y}}_i, k)$ are non-overlapping. In the contrary case, the same arguments can be applied to another predicted set S' created by replacing the overlapping labels in $S(\hat{\mathbf{y}}_i, k)$ with different labels having smaller $\hat{y}_{il}$ values. Thus bounding ranking error on S' will also bound it on $S(\hat{\mathbf{y}}_i, k)$. Now,

$$\frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} y_{il} - \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} y_{il} \quad (11)$$

$$\leq \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} y_{il} - \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} y_{il} + \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} \hat{y}_{il} - \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} \hat{y}_{il} \quad (12)$$

$$\leq \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} e_{il} + \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} e_{il} \ \text{ where, } \ e_{il} = |y_{il} - \hat{y}_{il}| \quad (13)$$

$$\leq \frac{1}{k}\sum_{l \in S(\mathbf{e}_i, 2k)} e_{il} \quad (14)$$

$$= 2\text{MAD@}2k(\hat{\mathbf{y}}, \mathbf{y}) \quad (15)$$

Bounding the regression error is quite straightforward, hence we skip the proof here.

Finally, the XMAD@$2k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \leq$ XRMSE@$2k(\hat{\mathbf{y}}_i, \mathbf{y}_i)$ property follows by using Jensen's inequality on the square function which is concave. □

Theorem A.2. *Given that $y_l = \prod_{h=1}^{H} z_{lh}$ and $\hat{y}_l = \prod_{h=1}^{H} \hat{z}_{lh}$ and under the standard unvisited node assumption of Parabel*

$$D_{KL}(y_l || \hat{y}_l) \leq \sum_{h=1}^{H} s_{lh} D_{KL}(z_{lh} || \hat{z}_{lh}) \ \textit{where}\ s_{lh} = \prod_{\tilde{h}=1}^{h} z_{l(\tilde{h}-1)} \quad (16)$$

Proof. We assume that $0 \log \frac{0}{p} = 0$ where $0 \leq p \leq 1$. We also use the unvisited node assumption in Parabel, $\mathbb{P}(z_{lh} = 0 | z_{l(h-1)} = 0) = 1$, which means that a child of an unvisited node is never visited.

Let $I_{y_l} \in \{0, 1\}$ be the probabilistic variable which says whether label l is relevant to a data point in reference, *i.e.* $\mathbb{P}(I_{y_l} = 1) = y_l$ and $\mathbb{P}(I_{y_l} = 0) = 1 - y_l$. Similarly let $I_{z_{lh}} \in \{0, 1\}$ be the probabilistic variable which says whether the data point visits node n_{lh} or not, *i.e.* $\mathbb{P}(I_{z_{lh}} = 1) = z_{lh}$ and $\mathbb{P}(I_{z_{lh}} = 0) = 1 - z_{lh}$.

Now, since the relevance of label l to a data point is equivalent whether the label path is traversed in the tree by the data point: $y_l = z_{lH}$ and $\mathbb{P}(I_{y_l}) = \mathbb{P}(I_{z_{lH}}, \cdots, I_{z_{l1}})$ hold true.

Table 7: XReg has the best or close to the best ranking and regression performance across all the datasets compared to state-of-the-art extreme classifiers and large-scale regressors and rankers. Re-ranking with tail classifiers (XReg-t) further improves the accuracies. PSP@k, CTR@k and Rating@k are variants of WP@k as discussed in Section 3. "-p": pointwise, "-l": labelwise.

Method	PSP-p@1 (%)	PSP-p@3 (%)	Tau-p@1 (%)	Tau-p@3 (%)	nDCG-p@5 (%)	XRMSE-p@5
BibTex						
PfastreXML	52.43	53.41	41.31	48.36	56.41	0.3813
Parabel	50.88	52.42	36.57	46.28	54.58	0.4104
LEML	51.30	52.17	39.32	47.31	54.10	0.3935
1-vs-all-LS	**53.50**	**55.10**	**39.91**	**49.31**	**57.30**	0.3834
RankSVM	49.31	51.79	39.22	47.07	54.97	0.7228
DiSMEC	50.88	52.52	36.66	46.34	54.54	0.4104
ProXML	50.10	52.00	-	-	-	-
XReg	49.66	52.66	38.48	47.09	54.98	0.3958
XReg-t	49.86	53.04	38.68	47.33	55.14	**0.3805**
EURLex-4k						
PfastreXML	40.16	43.07	46.97	46.50	43.64	0.2188
Parabel	36.36	44.04	40.96	46.03	44.78	0.4673
LEML	27.20	30.15	30.47	33.62	30.73	0.2406
1-vs-all-LS	**47.02**	**50.85**	**54.12**	**52.45**	**50.82**	**0.2006**
RankSVM	39.52	43.82	51.23	49.79	44.49	1.2093
DiSMEC	37.58	45.92	42.32	47.56	46.73	0.4771
ProXML	45.20	48.50	-	-	-	
XReg	44.00	47.44	51.69	50.51	47.99	0.2127
XReg-t	45.23	48.51	53.06	51.59	48.89	0.2338
Wiki10-31K						
PfastreXML	12.94	14.80	11.93	17.53	15.13	0.5925
Parabel	11.66	12.73	13.36	17.18	13.13	0.7201
LEML	11.25	12.38	15.29	18.40	12.58	0.5938
1-vs-all-LS	**26.78**	23.06	36.59	29.50	23.01	**0.5691**
RankSVM	21.06	18.99	32.63	27.58	19.05	1.2279
DiSMEC	11.91	14.09	14.41	19.47	14.63	0.7140
XReg	17.33	16.73	26.76	25.01	16.98	0.5931
XReg-t	25.92	**23.56**	**38.72**	**33.09**	**23.40**	0.5722
WikiLSHTC-325K						
PfastreXML	25.67	26.57	31.11	34.15	27.09	0.1922
Parabel	26.71	33.16	28.05	36.99	33.48	0.3130
DiSMEC	29.10	35.60	-	-	-	-
ProXML	34.80	37.70	-	-	-	-
XReg	32.36	34.36	36.59	39.10	35.13	**0.1877**
XReg-t	**36.85**	**37.98**	**41.41**	**41.96**	**38.87**	0.3241
Amazon-670K						
PfastreXML	24.52	26.65	28.18	29.30	27.36	0.4356
Parabel	25.43	29.45	20.54	26.56	30.72	0.4640
DiSMEC	25.82	30.20	20.40	26.77	31.89	0.4582
ProXML	30.80	32.80	-	-	-	-
XReg	29.12	31.19	31.69	32.63	32.01	**0.4189**
XReg-t	**31.16**	**32.71**	**33.83**	**34.28**	**33.34**	0.4639

Method	CTR-p@1 (%)	CTR-p@3 (%)	Tau-p@1 (%)	Tau-p@3 (%)	nDCG-p@5 (%)	XRMSE-p@5
SSA-130K						
PfastreXML	21.34	25.24	22.33	23.1	25.56	**0.0817**
Parabel	21.95	28.87	26.98	**28.83**	29.22	0.1636
LEML	3.79	5.11	7.2	7.61	5.50	0.0835
RankSVM	8.92	10.96	13.14	13.74	11.51	2.7945
DiSMEC	21.36	28.41	25.68	27.64	28.85	0.1746
XReg	24.69	29.02	27.52	27.71	29.64	0.0826
XReg-t	**24.7**	**29.22**	**27.32**	27.83	**29.93**	0.1225

Method	Rating-l@1 (%)	Rating-l@3 (%)	Tau-l@1 (%)	Tau-l@3 (%)	nDCG-l@5 (%)	XRMSE-l@5
YahooMovie-8K						
PfastreXML	11.5	9.9	22.29	20.21	10.36	0.7047
Parabel	11.28	9.73	30.11	29.08	10.03	0.7054
LEML	21.33	21.06	28.81	29.5	21.55	0.6851
1-vs-all-LS	22.75	21.02	34.9	32.59	21.76	0.6791
RankSVM	24.89	23.16	36.99	34.7	24.53	1.0613
DiSMEC	23.76	23.19	34.62	33.45	24.10	0.6826
XLR	3.87	4.2	12.44	11.78	4.40	0.7182
XReg	26.49	24.77	39.02	35.8	25.76	0.6944
XReg-t	**26.53**	**24.86**	**39.28**	**36.42**	**25.90**	**0.6772**
MovieLens-138K						
PfastreXML	9.03	7.82	25.92	23.88	7.63	0.9253
Parabel	5.95	4.25	40.67	39.08	4.03	0.9254
LEML	46.51	44.89	69.58	66.96	43.97	0.8773
1-vs-all-LS	46.94	43.88	43.17	69.28	65.92	0.8882
DiSMEC	50.85	47.05	65.45	62.93	46.49	0.8909
XLR	14.49	10.31	31.61	21.5	10.55	0.9184
XReg	54.65	50.83	71.59	68.83	50.16	0.8793
XReg-t	**55.04**	**51.21**	**72.07**	**69.3**	**50.52**	**0.8337**

Method	CTR-l@1 (%)	CTR-l@3 (%)	Tau-l@1 (%)	Tau-l@3 (%)	nDCG-l@5 (%)	XRMSE-l@5
DSA-130K						
PfastreXML	18.15	23.7	**26.77**	**30.99**	23.93	**0.0647**
Parabel	19.97	28.06	23.44	26.04	28.13	0.1091
LEML	3.94	6.9	5.35	6.46	7.54	0.0657
XLR	0.03	0.07	0.1	0.1	0.07	0.4837
DiSMEC	18.94	27.52	22.20	25.25	27.70	0.1201
XReg	22.07	29.73	23.73	26.08	29.95	0.0654
XReg-t	**22.41**	**30.1**	23.98	26.13	**30.43**	0.0744
DSA-1M						
Parabel	25.78	33.15	27.93	29.38	33.28	0.1218
XReg	26.75	33.06	28.67	29.51	33.36	**0.0806**
XReg-t	**27.83**	**34.27**	**29.68**	**30.18**	**34.55**	0.0892

Due the fact that $x \log \frac{x}{y}$ is a convex function, it is easy to show that the KL-divergence between 2 marginal distributions is upper bounded by the KL-divergence of the corresponding joint distributions.

$$D_{KL}(\mathbb{P}(I_{y_l})||\mathbb{P}(I_{\hat{y}_l})) = D_{KL}(\mathbb{P}(I_{z_{lH}})||\mathbb{P}(I_{\hat{z}_{lH}})) \tag{17}$$

$$\leq D_{KL}(\mathbb{P}(I_{z_{lH}}, \cdots, I_{z_{l1}})||\mathbb{P}(I_{\hat{z}_{lH}}, \cdots, I_{\hat{z}_{l1}})) \tag{18}$$

By using chain rule of KL-Divergence: (19)

$$= D_{KL}(\mathbb{P}(I_{z_{l(H-1)}}, \cdots, I_{z_{l1}})||\mathbb{P}(I_{\hat{z}_{l(H-1)}}, \cdots, I_{\hat{z}_{l1}})) \tag{20}$$

$$+ \mathbb{P}(I_{z_{l(H-1)}} = 1) D_{KL}(\mathbb{P}(I_{z_{lH}}|I_{z_{l(H-1)}} = 1)||\mathbb{P}(I_{\hat{z}_{lH}}|I_{\hat{z}_{l(H-1)}} = 1)) \tag{21}$$

$$+ \mathbb{P}(I_{z_{l(H-1)}} = 0) D_{KL}(\mathbb{P}(I_{z_{lH}}|I_{z_{l(H-1)}} = 0)||\mathbb{P}(I_{\hat{z}_{lH}}|I_{\hat{z}_{l(H-1)}} = 0)) \tag{22}$$

By unvisited node assumption, (23)

$$I_{z_{l(H-1)}} = 0 \implies I_{z_{lH}} = 0 \text{ and } I_{\hat{z}_{l(H-1)}} = 0 \implies I_{\hat{z}_{lH}} = 0 \tag{24}$$

hence: (25)

$$= D_{KL}(\mathbb{P}(I_{z_{l(H-1)}}, \cdots, I_{z_{l1}})||\mathbb{P}(I_{\hat{z}_{l(H-1)}}, \cdots, I_{\hat{z}_{l1}})) \tag{26}$$

$$+ \mathbb{P}(I_{z_{l(H-1)}} = 1) D_{KL}(\mathbb{P}(I_{z_{lH}}|I_{z_{l(H-1)}} = 1)||\mathbb{P}(I_{\hat{z}_{lH}}|I_{\hat{z}_{l(H-1)}} = 1)) \tag{27}$$

$$= D_{KL}(\mathbb{P}(I_{z_{l(H-1)}}, \cdots, I_{z_{l1}})||\mathbb{P}(I_{\hat{z}_{l(H-1)}}, \cdots, I_{\hat{z}_{l1}})) \tag{28}$$

$$+ (\prod_{\tilde{h}=1}^{H-1} z_{l\tilde{h}}) \Big(z_{lh} \log \frac{z_{lh}}{\hat{z}_{lh}} + (1 - z_{lh}) \log \frac{1 - z_{lh}}{1 - \hat{z}_{lh}} \Big) \tag{29}$$

Table 8: Hyperparameter tuning for # trees (T), Max leaf labels (M), Beam width (P) and points reaching leaf node per label in labelwise prediction of XReg. Note: The hyperparameters in bold face are finally chosen for the default setting.

# trees	WP@5 (%)	Tau@5 (%)	XMAD@5	Training time (hrs)	Test time /point (ms)	Model Size (GB)
EURLex-4K (pointwise)						
1	48.05	51.14	0.1899	0.0307	0.3911	0.0125
3	49.72	52.86	0.1849	0.0642	1.2899	0.0378
5	50.25	53.24	0.1836	0.0995	2.638	0.0629
7	50.44	53.42	0.1831	0.1543	3.4703	0.0881
Amazon-670K (pointwise)						
1	30.50	32.31	0.3956	0.6788	0.6551	1.1478
3	33.24	34.72	0.3869	1.4925	2.4633	3.4186
5	34.00	35.45	0.3847	2.1499	6.9283	5.6978
7	34.37	35.86	0.3837	3.4298	8.564	7.9768
DSA-130K (labelwise)						
1	33.64	27.52	0.0448	0.2165	1.8552	0.2624
3	35.66	28.51	0.0439	0.4570	7.4715	0.7871
5	36.24	28.94	0.0436	0.6836	16.3785	1.3117
7	36.55	29.2	0.0435	1.0897	21.7585	1.8358

Beam width	WP@5 (%)	Tau@5 (%)	XMAD@5	Training time (hrs)	Test time /point (ms)	Model Size (GB)
EURLex-4K (pointwise)						
5	49.6	52.76	0.1858	0.0627	0.6869	0.0378
10	49.72	52.86	0.1849	0.0642	1.2899	0.0378
20	49.7	52.86	0.1847	0.0638	2.4982	0.0378
30	49.71	52.86	0.1847	0.0682	3.8542	0.0378
Amazon-670K (pointwise)						
5	32.77	34.37	38.77	1.3654	1.4467	3.4186
10	33.24	34.72	0.3869	1.4925	2.4633	3.4186
20	33.38	34.82	0.3866	1.3721	4.8842	3.4186
30	33.4	34.83	0.3866	1.5508	9.1728	3.4186

Max leaf labels	WP@5 (%)	Tau@5 (%)	XMAD@5	Training time (hrs)	Test time /point (ms)	Model Size (GB)
EURLex-4K (pointwise)						
20	49.34	52.75	0.1845	0.0323	0.4694	0.0494
50	49.6	52.71	0.1859	0.0458	0.8198	0.0428
100	49.72	52.86	0.1849	0.0642	1.2899	0.0378
200	50.11	53.05	0.1846	0.1031	2.6368	0.0337
Amazon-670K (pointwise)						
20	32.26	33.96	0.3869	0.7514	0.7051	6.0288
50	32.89	34.46	0.3868	1.1171	1.7382	4.124
100	33.24	34.72	0.3869	1.4925	2.4633	3.4186
200	33.56	34.99	0.3866	2.1426	4.3993	2.9268
DSA-130K (labelwise)						
20	34.76	28.2	0.0448	0.2678	2.279	0.9867
50	35.21	28.42	0.0443	0.3516	4.8664	0.8764
100	35.66	28.51	0.0439	0.4570	7.4715	0.7871
200	35.96	28.63	0.04351	0.6925	8.9019	0.7128

# per-label points	WP@5 (%)	Tau@5 (%)	XMAD@5	Training time (hrs)	Test time /point (ms)	Model Size (GB)
DSA-130K (labelwise)						
5	35.56	28.41	0.0438	0.4989	4.3407	0.7871
10	35.66	28.51	0.0439	0.4570	7.4715	0.7871
20	35.68	28.54	0.0438	0.5164	11.1622	0.7871
30	35.68	28.54	0.0438	0.5135	11.5373	0.7871

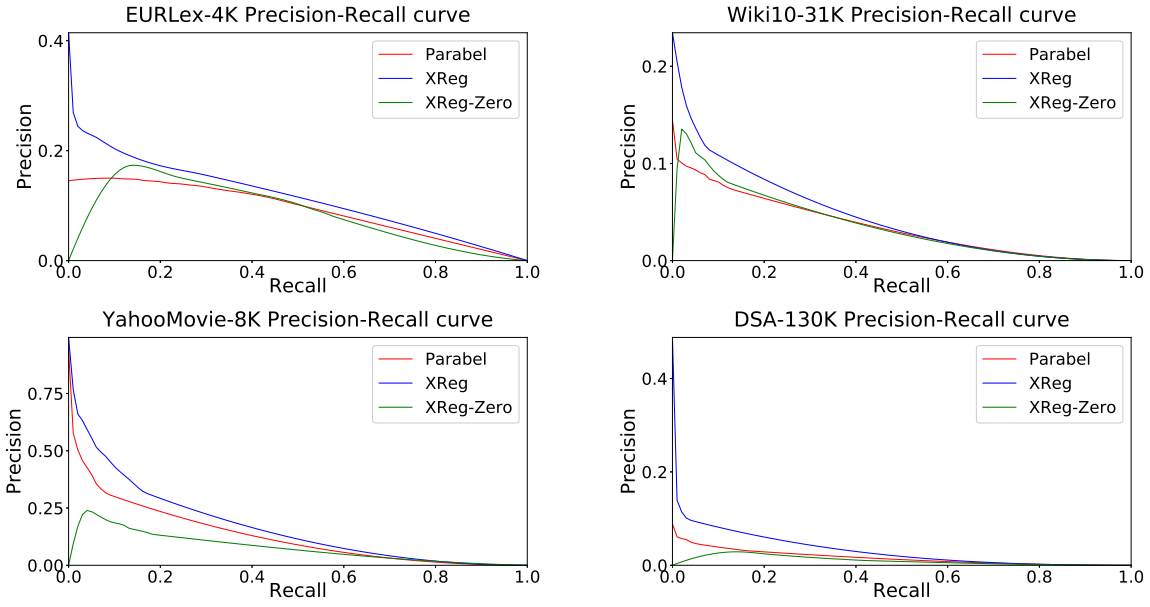

Figure 1: Precision-Recall curves showing that XReg is consistently better than XReg-Zero and Parabel approaches for precision recall tradeoff.

By recursively applying above simplification (30)

at higher level tree nodes: (31)

$$= \sum_{h=1}^{H} s_{lh}\Big(z_{lh}\log\frac{z_{lh}}{\hat{z}_{lh}} + (1-z_{lh})\log\frac{1-z_{lh}}{1-\hat{z}_{lh}}\Big) \quad (32)$$

The above upper bound is exactly the quantity that XReg minimizes during training by assuming logistic model for probability estimates. □

Lemma A.3. *XReg's overall training objective minimizes an upper bound over XMAD@k for all k, with the bound being tighter for smaller k values.*

Proof. As presented in the next theorem, XReg minimizes an upper bound on XMAD@1 = $max_{l=1}^{L}|y_l - \hat{y}_l|$. Note that XMAD@$k \leq$ XMAD@1$\forall k$. Furthermore, as k increases, XMAD@k averages smaller and smaller errors compared the largest errors, therefore the bound is tighter for smaller values of k which are close to $k = 1$. □

Theorem A.4. *When each data point has at most $O(\log L)$ positive labels, the expected WP@k regret and XMAD@k error suffered by XReg's pointwise inference algorithm are bounded by:*

$$O(\log^2 L\sqrt{\frac{W}{\sqrt{Np}}}\sqrt{1+\sqrt{5\log\frac{3L}{\delta}}})$$

with probability at least $1-\delta$, where N is the total training points, L is the number of labels, W is the maximum norm across all node classifier vectors and p is the minimum probability density of $\mathbf{x}$ distribution that any tree node receives.

Proof. The outline of the proof is as follows. First, we see that the WP@k regret and XMAD@k error for a given data point are bounded, in a straight forward manner, by XReg's node and label classifier objectives over that data point. For good overall generalization performance, each classifier needs to receive enough training samples as well as learn to generalize well from those samples. We derive probability bounds for those events simultaneously. While these steps together give the regret bounds for the classifier during exact prediction (*i.e.*, calculate the scores for all labels for a given test point), the regret suffered by the greedy, beam-search algorithm might actually be more than that. Therefore, in a follow-up step, we also give a bound for this approximate algorithm which is only worse by $O(\log L)$. This gives us the final sample complexities.

By Lemma (7), both $\frac{1}{2}$WP@k and XMAD@$2k$ are bounded by XRMSE@$2k$ which is in turn bounded by $\max_{l=1}^{L}|y_l - \hat{y}_l|$.

Now, using Pinsker's inequality [**?**],

$$\max_{l=1}^{L}|y_l - \hat{y}_l| \quad (33)$$

$$\leq \max_{l=1}^{L}\sqrt{\frac{1}{2}D_{KL}(y_l, \hat{y}_l)} \quad (34)$$

$$= \sqrt{\max_{l=1}^{L}\frac{1}{2}D_{KL}(y_l, \hat{y}_l)} \quad (35)$$

From (16): (36)

$$\leq \sqrt{\max_{l=1}^{L}\frac{1}{2}\sum_{h=1}^{H}s_{lh}D_{KL}(z_{lh}||\hat{z}_{lh})} \quad (37)$$

$$\leq \sqrt{\frac{1}{2}\sum_{n:z_{n-1}>0}s_n D_{KL}(z_n||\hat{z}_n)} \quad (38)$$

where z_{n-1} is value in parent of node n (39)

(40)

□

For good generalization performance, we need a small expected regret with respect to distribution over data point $\mathbf{x}$:

$$\mathbb{E}_x \max_{l=1}^{L}|y_l - \hat{y}_l| \quad (41)$$

By concavity of square root function: (42)

$$\leq \sqrt{\frac{1}{2}\mathbb{E}_x\sum_{n:z_{n-1}>0}s_n D_{KL}(z_n||\hat{z}_n)} \quad (43)$$

Now we try to bound the above quantity by relating it to training error.

Let p_n be the expected fraction of the probability density over $\mathbf{x}$ that a tree node n receives. This is precisely the density of data points which have at least one label with non-zero relevance in the subtree rooted at node n. Now, let's compute the probability that the node n receives at least $Np_n(1-k)$ training points where N is the number of total training points and Np_n is the expected number of training points that node n would receive. By using chernoff bound, this probability is at least $1-\exp(-\frac{p_n N k^2}{2})$. Now, the probability that all tree nodes n would simultaneously receive at least $Np_n(1-k)$ training points is at least $1-L\exp(-\frac{pNk^2}{2})$ since there are at most L tree nodes and each has $\mathbf{x}$ density of at least p.

Now, we use the result in [**?**]. Since the logistic loss used for modeling probabilities in XReg is lipschitz continuous with constant 1 and logistic regression parameters are bounded by norm W, and $\mathbf{x}$ is bounded by norm 1, for any regressor in XReg,

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq \hat{\mathbb{E}}_x s_n D_{KL}(z_n, \hat{z}_n) + 2W\sqrt{\frac{1}{Np(1-k)}} + 2W\Delta \quad (44)$$

with probability at least $1-\exp(-2Np(1-k)\Delta^2)$ where $\hat{\mathbb{E}}_x D_{KL}(z_n, \hat{z}_n)$ is the average training error in node n which is 0 as per our assumption.

Combining the above reasonings, along with the fact that there are at most $2L$ regressors in XReg, we can conclude that with probability of at least $1 - L\exp(-\frac{pNk^2}{2}) - 2L\exp(-2Np(1-k)\Delta^2)$, each node has expected error bounded simultaneously as below:

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np(1-k)}} + 2W\Delta \tag{45}$$

Now, note that k can be given any value in $[0, 1]$ and the above bounds vary accordingly. We choose to give $k = 2\Delta(\sqrt{\Delta^2+1} - \Delta)$. Then, with probability at least $1 - 3L\exp(-2(\sqrt{(2)}-1)^2 Np\Delta^2)$, for all regressors

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np(1-2\Delta(\sqrt{(\Delta^2+1)}-\Delta))}} + 2W\Delta \tag{46}$$

In other words, with probability at least $1 - \delta$ over the training samples, for all regressors,

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np(1-2\Delta(\sqrt{(\Delta^2+1)}-\Delta))}} \tag{47}$$

$$+ 2W\sqrt{\frac{1}{2(\sqrt{2}-1)^2 Np}\log\left(\frac{3L}{\delta}\right)} \tag{48}$$

where $\Delta = \sqrt{\frac{1}{2(\sqrt{2}-1)^2 Np}\log\left(\frac{3L}{\delta}\right)}$. Now since $\Delta \to 0$ as $N \to \infty$, for large enough N, the above bound can be approximated to

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np}} + 2W\sqrt{\frac{1}{2(\sqrt{2}-1)^2 Np}\log\left(\frac{3L}{\delta}\right)} \tag{49}$$

From (50),

$$\mathbb{E}_x \max_{l=1}^{L} |y_l - \hat{y}_l| \tag{50}$$

$$\leq \sqrt{\frac{1}{2}\mathbb{E}_x \sum_{n:z_{n-1}>0} s_n D_{KL}(z_n||\hat{z}_n)} \tag{51}$$

Since any $\mathbf{x}$ has on average $\log L$ non-zero labels and (52)

since height of the tree is $\log L$ (53)

the number of nodes with $z_{n-1} > 0$ for any $\mathbf{x}$ is on average $\log^2 L$, hence: (54)

$$\leq \sqrt{\frac{\log^2 L}{2}\left(2W\sqrt{\frac{1}{Np}} + 2W\sqrt{\frac{1}{2(\sqrt{2}-1)^2 Np}\log\left(\frac{3L}{\delta}\right)}\right)} \tag{55}$$

$$\leq \log L\sqrt{\frac{W}{\sqrt{Np}}}\sqrt{1 + \sqrt{5\log\left(\frac{3L}{\delta}\right)}} \tag{56}$$

with probability at least $1 - \delta$ over training samples.

The above bound holds for exact prediction where all label probabilities are computed for a given test point. Now we analyse the extra regret due to the greedy, approximate, beam search based, pointwise inference algorithm used by XReg.

During beam-search, a point traverses the tree level-by-level. At each tree level, a small shortlist of around $k = 10$ most probable nodes, *i.e.* nodes with most relevant labels their subtrees, are maintained and extended on to next level. If accurate label relevances were available, then beam search would always return the best set of labels, since each node's z_n variable value matches the most relevant label in its subtree. Unfortunately, due to generalization error, the estimated $\hat{z}_n$ values might not exactly match the Z_n values. As a result, the regret accumulates at each tree level whenever a node with lower z_n is maintained in shortlist instead of the highest one. The regret suffered is at most $\max_{n\in S} 2|z_n - \hat{z}_n|$, where S is the set of shortlisted nodes at a tree level. A little more algebra reveals that this quantity is in fact bounded by (50).

$$\max_{n\in S} 2|z_n - \hat{z}_n| \leq \sqrt{\frac{1}{2}\mathbb{E}_x \sum_{n:z_{n-1}>0} s_n D_{KL}(z_n||\hat{z}_n)} \tag{57}$$

which is the bound on the regret suffered by exact prediction algorithm. That is, beam-search can suffer at most the same amount of regret at each tree level that exact prediction suffers as a whole. Now since there are $\log L$ tree levels, the regret of beam search algorithm is bounded by

$$\leq \log^2 L\sqrt{\frac{W}{\sqrt{Np}}}\sqrt{1 + \sqrt{5\log\left(\frac{3L}{\delta}\right)}} \tag{58}$$