Extreme Regression for Dynamic Search Advertising

Anonymous Author(s)

ABSTRACT

Extreme multi-label classification techniques, which efficiently choose a subset of labels that are relevant to a data point from millions of labels, are being increasingly applied to large-scale recommendation and ranking problems. These techniques, however, implicitly assume that each label is either fully relevant or fully irrelevant to a data point which does not hold in practice. Consequently, they often recommend less relevant labels in the place of more relevant ones since they are all implicitly treated as being fully relevant. When applied to Dynamic Search Advertising (DSA), they show ads that have low probabilities of receiving clicks for a search query, thus reducing the ad quality and revenue. This paper addresses these limitations through: (1) a *new learning paradigm* called eXtreme Regression (XR) which chooses the most relevant labels by directly predicting the label relevance weights, such as ad click probabilities or movie ratings, of an extremely large number of labels with respect to a data point; (2) *new evaluation metrics* for XR which measure how relevant are the top ranked labels to a data point, while also measuring how accurate are their estimated relevance weights; and (3) a *new algorithm* called eXtreme Regressor (XReg) that minimizes an upper bound of such metrics thus achieving good ranking performance as well as producing accurate regression estimates for downstream processing. Additionally, this paper also introduces the *new labelwise inference paradigm* in XReg which is necessary for DSA and other recommendation tasks.

Experiments on benchmark datasets demonstrated that XReg consistently outperforms the state-of-the-art extreme classifiers as well as large-scale regressors and rankers by 100% on the new regression metrics, by 2% on the propensity-scored precision metric popular in extreme classification and by 2.4% on the click-through rate metric used in DSA. Furthermore, XReg is highly efficient with $O(\text{polylog}(L))$ computational and sample complexities in the number of labels L and thus easily scales large datasets with millions of labels. Deployment of XReg on DSA in Bing resulted in a relative gain of 58% in revenue and 27% in query coverage. Efficient code for XReg will be made publicly available.

ACM Reference Format:
Anonymous Author(s). 2020. Extreme Regression for Dynamic Search Advertising. In *Proceedings of 13th ACM International Conference on Web Search and Data Mining (WSDM 2020).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Objective: This paper introduces a new learning paradigm called eXtreme Regression (XR) which involves predicting the labels that are most relevant to a data point from an extremely large label set by directly estimating the non-binary relevance weights of all the labels with respect to a data point. XR provides a natural solution to large-scale applications such as Dynamic Search Advertising (DSA) by treating each ad as a data point and each search query as a label, and predicting the probability that an ad receives a click when shown against a query. This paper also introduces suitable metrics and algorithms for XR.

Dynamic Search Advertising: DSA is a form of search advertising where the search engine automatically chooses the ads to be shown against a user's query beside automatically generating the ad-copy, ad-title, bid-phrases *etc.* that are necessary for showing these ads. DSA is being increasingly adopted by advertisers since it eliminates the need for manually chosen bid-phrases, results in faster deployment of new ad campaigns and allows more accurate user targeting. It is important in DSA that the shown ads satisfy the user's requirements and receive clicks, since these clicks earn revenue for the search engine and perhaps cause conversions desired by the advertiser. Consequently, it is essential to ensure that the ads shown are highly relevant to the entered query. Furthermore, search engines like Bing have a constantly evolving corpus of billions of ads, cater to millions of different search queries with millisecond latencies and serve multiple markets. As a result, the algorithms used for DSA need to train and predict very efficiently and comfortably scale to extremely large number of ads and queries. Traditional approaches to DSA based on information retrieval techniques [25], topic models [51], language models [43, 58] as well as deep learning [20, 46] fail on pithy ad-landing pages or suffer from low query and ad coverages. Extreme classifiers such as Parabel [40] have also been applied to DSA with improved ad coverage and click-through rates, but their query coverage is low.

Extreme Multi-label Classification: Extreme multi-label classifiers (henceforth referred to as extreme classifiers) are being increasingly used to solve large-scale recommendation and ranking problems such as DSA, document tagging and product recommendation [21, 41]. Extreme classifiers annotate a data point with the most relevant *subset* of labels from an extremely large label set. This uses an implicit, binary relevance assumption that each label in the subset is fully relevant and each of the remaining labels is fully irrelevant to the data point. State-of-the-art extreme classifiers such as Parabel and Slice [21] are able to efficiently train on millions of data points and labels within just a few hours and make predictions in just a few milliseconds for a test point, thus making them very appealing for real-world applications.

Unfortunately, the binary relevance assumption rarely holds true in practice. For example, in DSA, a query covering multiple intents may have different click probabilities for different ads: 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. The click probabilities might also vary based on the ad's quality, the query's purchase intent, the user's geographical location *etc.* By approximating the click probabilities to either 0 or 1, these extreme classifiers lose useful information and might end up recommending less relevant ads. Although some extreme classifiers such as PfastreXML [22] and LEML [60] allow learning from non-binary relevance weights, they

WSDM 2020, February 2020, Houston, Texas, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

tend to be significantly inaccurate and inefficient. The proposed XReg algorithm extends Parabel to directly predict these non-binary click probabilities thus leading to improved prediction accuracy for the same computational efficiency as Parabel.

Ranking and regression approaches: Ranking and regression approaches are the natural alternatives for addressing the non-binary label relevance scenario. Given a data point, most ranking algorithms learn to rank a more relevant label higher than a less relevant label for each label pair. Since their training complexities scale quadratically in the number of labels L, they don't scale to millions of labels in DSA. On the other hand, the regression algorithms (sometimes called as pointwise rankers) predict a numerical score for each label that captures its relevance to a data point. These have $O(L)$ training and prediction time complexities, but still take weeks to train and several seconds to predict per test point, and hence don't meet the requirements of DSA. In comparison, XReg has $O(\log L)$ training and prediction time complexities and makes more relevant recommendations than the existing rankers and regressors.

eXtreme Regression: Several real-world applications can be reformulated as XR problems. Extreme classification itself can be posed as an XR problem of predicting the label relevance weights, generated as binary label relevances multiplied by the inverse propensity scores of respective labels, for a given data point. The inverse propensity scores give larger weights to rare labels which are often more useful [22]. Similarly, XR can also be used to predict the ratings that a user might give to new movies and thus recommend the most relevant movies. DSA can also be posed as an XR problem. The primary goal in these applications is to recommend a small number of most relevant labels for a data point. XR achieves this by estimating the relevance weights for each label and then recommending the top ranked labels according to these estimates. These estimates are also useful in several downstream processing tasks such as: (1) label filtering based on relevance thresholds to obtain a favourable precision-recall trade-off; (2) combining these relevance weight estimates in a principled way with those from other approaches to improve accuracy further. Hence, the predictions need to be accurate in terms of both ranking and regression performance. For faithful evaluation in such settings, the newly proposed XR metrics are more suitable than the existing regression or ranking metrics.

eXtreme Regression metrics: XR metrics are based on an intuition that the ranking errors at the top occur mainly due to either highly underestimating or highly overestimating the relevances of most or least relevant labels respectively, leading to high regression errors on these labels. Therefore, measuring the largest regression errors gives a good proxy for the ranking quality as well. The traditional regression metrics which sum up the regression errors across all the labels provide a very loose bound for ranking error. The XR metrics modify them to focus only on the most erroneous labels thus giving a tighter bound on both ranking and regression errors at the top. These metrics could then for training XR models, evaluating predictions, tuning hyper-parameters, selecting accurate models, and so on.

eXtreme Regressor algorithm: The XReg algorithm efficiently regresses over millions of label relevance weights in logarithmic time by using the observation that most labels in XR applications are fully irrelevant to a data point. XReg achieves this by leveraging a balanced label tree which partitions the relevant labels together for most data points. Due to this, large groups of fully irrelevant labels frequently occur for which common regressors can be learnt, thus leading to highly efficient training. During prediction, a data point greedily traverses the label tree by exploring only logarithmic number of most promising paths which lead to highly relevant labels. The XReg method is similar to Parabel in many ways. However, unlike Parabel, the tree construction uses non-binary relevance weights leading to better partitions, the nodes contain efficient regressors instead of classifiers, and the predictions give reliable relevance weight estimates along with label rankings. As a result, XReg can be up to 3% better in relevance-weighted precision and 100% better in XR metrics than Parabel. XReg also consistently outperforms large-scale regressors and rankers in terms of both regression and ranking errors.

Labelwise inference: Whereas the usual prediction scenario in extreme classifiers involves recommending the most relevant labels for a test point, applications such as DSA and movie recommendation are more naturally 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 to here as labelwise inference. On these tasks, Parabel and PfastreXML end up recommending a small set of highly popular labels that are relevant to all test points, resulting in low label coverage. On the other hand, naive regressors such as LEML and 1-vs-all least squares can easily predict the highest scoring test points for one label at a time, but they are highly inefficient. This paper develops an efficient labelwise inference algorithm for the XReg model, which results in 27% better query coverage than Parabel on DSA.

Contributions: This paper makes the following technical contributions: (1) a *new learning paradigm* called eXtreme Regression (XR) applicable to many real-world applications like extreme classification, DSA and movie recommendation; (2) *new evaluation metrics* for XR which evaluate both ranking and regression performance; (3) a *new algorithm* called eXtreme Regressor (XReg) which achieves good ranking performance and produces accurate regression estimates for downstream processing; and (4) a *new labelwise inference paradigm* for XReg which is necessary for DSA and other recommendation tasks. It further demonstrates that XReg could significantly improve the quality of dynamic search advertising on the Bing search engine.

2 RELATED WORK

Extreme classification metrics: Performance of extreme classifiers has, traditionally, been measured in terms of Precision@k and nDCG@k [7, 41]. Recently, propensity-scored metrics were introduced in [22] which give higher importance to more useful and informative tail labels. However, all these metrics ignore the regression error in the predicted relevance estimates unlike the new XR metrics introduced in this paper. When applied to DSA, these existing metrics also ignore the ad click-through rates leading to less meaningful measurements [40].

Extreme classification algorithms: Much progress has recently been made in developing extreme multi-label classifiers based on trees [3, 22, 24, 39, 41, 47], embeddings [7, 9, 12, 18, 31, 35, 48, 52, 55] and 1-vs-all approaches [4, 5, 21, 32, 36, 40, 53, 56, 57]. Among these, 1-vs-all approaches like DiSMEC [4], ProXML [5], Parabel [40] and Slice [21] achieve state-of-the-art results on Precision@k, nDCG@k and their propensity-scored counterparts, but only train from binary labels and 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 [22] and LEML [60] 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.

Ranking & regression metrics: A plethora of accuracy metrics have been proposed in the ranking literature [7, 23, 27, 30, 41, 49, 62] including the ranking at the top metrics such as Precision@k, nDCG@k and AUC@k which are relevant to this paper. However, none of these measure the regression performance. On the other hand, the regression literature utilizes metrics such as Root Mean Squared Error [8] and Mean Absolute Deviation [8], which fully ignore the ranking performance. To avoid these pitfalls, some works in advertising [45] evaluate performance using 2 different metrics for regression and ranking, but then simultaneously optimizing for multiple metrics is non-trivial. In comparison, the new XR metrics simultaneously minimize both ranking and regression error and are amenable to efficient optimization.

Ranking & regression algorithms: Learning to rank methods [10, 14, 19, 29, 30, 37, 42, 44, 54, 59] have been highly popular in recommendation and ranking literature. But most of these approaches have super-linear dependence on the number of labels and hence don't scale to our datasets. During recommendation, they are primarily used to accurately re-rank a small shortlist of labels which are generated beforehand by more scalable approaches like extreme classifiers and XReg, and hence complementary to this work. Nevertheless, this work uses 2 learning to rank techniques called RankSVM [17, 29], and the more recent eXtreme Learning to Rank (XLR) [10] as baselines by using all non-zero ratings along with a subsample of zero ratings for scalable training. Regression literature has mostly focussed on smaller output spaces and cannot be applied out-of-the-box to XR datasets. This paper compares against the simplest, 1-vs-all least squares regression [50]. Results indicate that XReg is much more scalable and accurate than all these regression and ranking baselines.

Dynamic search advertising: Various approaches have been proposed for DSA in the organic search literature including information retrieval based methods [25], probabilistic methods and topic models [51] and deep learning [20, 46]; however these do not work well for pithy ad-landing pages. Techniques based on landing page summarization [11], translation and query language models [43, 58] and keyword suggestion based on Wikipedia concepts [61] have also been proposed for sponsored search; but these suffer from low coverage problem. Extreme classifiers such as Parabel have also been used for DSA to improve accuracy and ad coverage; but these still suffer from low query coverage and low generated revenue. As demonstrated in Section 5, XReg complements such approaches and significantly improves various online metrics when included in the Bing DSA ensemble in production.

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 actual 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 $\mathbf{v} \in [0,\infty)^L$.

Ranking metrics: This paper generalizes the precision@k and nDCG@k metrics traditionally used in extreme classification to *relevance-weighted* precision@k and nDCG@k which are defined, for a data point i, as follows:

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

$$\text{WN@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{iDCG(\mathbf{y}_i)} \sum_{l \in S(\hat{\mathbf{y}}_i, k)} \frac{y_{il}}{\log_2(l+1)} \tag{2}$$

where $iDCG(\mathbf{y}_i)$ is a standard normalizer which ensures that WN@k always lies between 0 and 1 [23]. They measure the quality of ranking only in the top k positions and also take into account the degree of label relevances. The WP@k metric reduces to (a) PSP@k when $\mathbf{y}_i$ are set to inverse propensity-scores multiplied by binary relevances; (b) CTR@k suitable for DSA when $\mathbf{y}_i$ is set to the ad click-through rates; (c) Rating@k when $\mathbf{y}_i$ is set to the user ratings, which is the undiscounted version of the familiar rating-based nDCG@k metric used in recommender systems [23].

Beside these, we also report using the AUC@k metric which is widely used in recommendation and ranking literature [30]:

$$\text{AUC@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = 1 - \frac{1}{|M(\mathbf{y}_i)|} \sum_{\substack{(l,l') \in M(\mathbf{y}_i) \\ l \in S(\hat{\mathbf{y}}_i, k)}} \mathbb{I}[\hat{y}_{il} > \hat{y}_{il'}] \tag{3}$$

This metric measures the number of wrongly ordered label pairs in terms of their relevance weights with $M(\mathbf{y}_i)$ being the maximum number of such pairs.

Extreme regression metrics: Although the above metrics are useful for measuring the ranking accuracy, they fully ignore the regression performance necessary for XR applications. To address this limitation, we introduce novel XR error metrics which could be used to simultaneously minimize the ranking and regression errors in the top k ranks:

$$\text{Ranking-error@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k} \sum_{l \in S(\mathbf{y}_i, k)} y_{il} - \frac{1}{k} \sum_{l \in S(\hat{\mathbf{y}}_i, k)} y_{il} \tag{4}$$

$$\text{Regression-error@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k} \sum_{l \in S(\hat{\mathbf{y}}_i, k)} |\hat{y}_{il} - y_{il}| \tag{5}$$

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:

$$\text{XMAD@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k} \sum_{l \in S(\mathbf{e}_i, k)} e_{il} \tag{6}$$

$$\text{XRMSE@}k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \sqrt{\frac{1}{k} \sum_{l \in S(\mathbf{e}_i, k)} e_{il}^2} \tag{7}$$

Note that the above metrics only measure the regression errors on the k most erroneous labels. The following lemma shows that these metrics upper bound both the regression and the ranking errors at k.

Lemma 3.1. *Given any true and predicted relevance weight vectors* $\mathbf{y}, \hat{\mathbf{y}} \in [0, \infty)^L$, *the following inequality hold true:*

$$0 \leq \frac{M}{2} \leq XMAD@2k(\hat{\mathbf{y}}, \mathbf{y}) \leq XRMSE@2k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \quad (8)$$
$$with,\ M = max(Ranking\text{-}error@k, Regression\text{-}error@k) \quad (9)$$

Proof. The proof is available in the supplementary material. □

Although the above lemma does not provide a lower bound on ranking and regression errors in terms of XR metrics, empirically it has been observed that XMAD@$2k(\hat{\mathbf{y}}, \mathbf{y}) \leq 2M$ usually holds (see Section 5), thus making XMAD@$2k$ error a good proxy for regression and ranking errors at k. Note that for $k = L$, XMAD@k, XRMSE@k reduce to the conventional MAD, RMSE metrics used in regression literature.

Labelwise metrics: The pointwise and labelwise prediction scenarios respectively involve recommending relevant labels to test points and test points to labels. To evaluate performance in the labelwise prediction scenario, all the above ranking and regression metrics, defined for pointwise predictions, need to be redefined appropriately. As an example, the relevance-weighted precision for a label l is be defined as:

$$\text{WP-l@}k(\hat{\mathbf{y}}_{*l}, \mathbf{y}_{*l}) = \frac{1}{k} \sum_{i \in S(\hat{\mathbf{y}}_{*l}, k)} y_{il} \quad (10)$$

where $\mathbf{y}_{*l}, \hat{\mathbf{y}}_{*l}$ now represent the vectors of true and predicted relevance weights of all the test points for label l. The other metrics can be modifications similarly. To promote clarity, all pointwise and labelwise metrics will be used with suffixes "-p" and "-l" respectively.

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 [40] 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 the points along the average of the training points which are relevant to the label:

$$\mathbf{v}_l = \mathbf{v}'_l / \|\mathbf{v}'_l\|_2 \quad \text{where} \quad \mathbf{v}'_l = \sum_{i=1}^{N} y_{il} \mathbf{x}_i \quad (11)$$

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 [28] 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*

$$D_{KL}(y_l || \hat{y}_l) \leq \sum_{h=1}^{H} s_{lh} D_{KL}(z_{lh} || \hat{z}_{lh}) \ where\ s_{lh} = \prod_{\tilde{h}=1}^{h} z_{l(\tilde{h}-1)} \quad (12)$$

Proof. Proof is provided in the supplementary material. □

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 [40]. 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 $1 - z_{lh}$:

$$\min_{\mathbf{w}_n} \|\mathbf{w}_n\|^2 + \frac{C}{|\mathcal{I}_n|} \sum_{i \in \mathcal{I}_n} \{s_{in} z_{in} \log(1 + \exp(-\mathbf{w}_n^\top x_i)) + \tag{13}$$

$$s_{in}(1 - z_{in}) \log(1 + \exp(+\mathbf{w}_n^\top x_i))\} \tag{14}$$

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. 13) is strongly convex and was optimized using the modified CDDual algorithm available from Liblinear package [15]. 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 material. □

4.3 XReg 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 material. 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 XReg 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 material.

5 EXPERIMENTS

Datasets: Experiments were carried out on several small/medium (< 100K labels) and large scale benchmark datasets (> 100K labels) with up to 4.9M training points, 1.8M dimensional features and 1.4M labels (see Table 1 for dataset statistics). These datasets cover diverse applications such as document tagging (BibTeX [41], EURLex-4K [34], Wiki10-31K [6] & WikiLSHTC-325K [7, 38]), content-based movie recommendation (YahooMovie-8K [2] & MovieLens-138K [1, 16]), item-to-item recommendation of Amazon products (Amazon-670K [7, 33]), 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 user-provided movie ratings as relevance weights and movie meta-data like text 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 either already available from [6] or will be publicly released.

Baselines: XReg was compared to leading extreme classifiers such as PfastreXML [22], Parabel [40], LEML [60], DiSMEC [4] and ProXML [5], traditional multivariate regressors such as one-vs-all least-squares regression (1-vs-all-LS) and LEML, and a popular pairwise ranker named RankSVM [17, 29]. XReg was also compared to the recent eXtreme Learning to Rank (XLR) [10] 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 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

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 has the best or close to the best ranking and regression metrics across all the datasets while being significantly faster during training and prediction when compared to 1-vs-all techniques. PSP@k, CTR@k and Rating@k are variants of WP@k as discussed in section 3. "-p": pointwise, "-l": labelwise and "-t": use of tail classifiers. Please refer to the Table 6 in the supplementary material for more results.

Method	PSP-p@5 (%)	AUC-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	58.3	-	-	-	-	-
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**	-	-	-	-	-
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 (%)	AUC-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 (%)	AUC-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 (%)	AUC-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

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 respective 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 < 10 training point samples [22]. 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 original relevance weights 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 points to training 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 found to generally work well with acceptable scalability and was used. The hyperparameter tuning studies over XReg's T, P, M can be found in Table 7 which show that the chosen default setting has optimal trade-off of accuracy and scalability and that increasing any of them further lead to minimal gains in accuracies 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 hyperparameter tuning on medium datasets and set to defaults suggested in the respective papers/codebases on larger datasets where tuning was not feasible.

Metrics and hardware: Performances were evaluated using accuracy metrics such as WP@k variants, AUC@k and XMAD@k (see Section 3) as well as efficiency metrics such as training time in hours, test time per data point and model size in GB. Among WP@k variants, for standard extreme classification repository datasets (BibTeX, EURLex, Wiki10, WikiLSHTC, and Amazon), 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 pointwise datasets: Table 2 compares XReg's performance to the baselines on the pointwise prediction datasets (BibTeX, EURLex, Wiki10, WikiLSHTC, Amazon & SSA) using the pointwise metrics discussed above. Starting with ranking performance, XReg-t is 0.71% and 3.84% better in terms of PSP-p@5 and AUC-p@5 respectively on Wiki10. On large-scale datasets (WikiLSHTC & Amazon), XReg-t is the state-of-the-art when evaluated using PSP-p@k where $k = 1, 3, 5$ (Table 2 and Table 6 in the supplementary material) metrics. XReg-t is up to 2% better than the current state of the art ProXML in PSP-p@1 while being scalable up to 300x during training and 700x during prediction. For fair comparison, all the baselines were run on small-scale datasets (BibTex & EURLex) where 1-vs-all techniques scale effectively, XReg-t is only up to 1.8% lower in PSP-p@5, up to 1.75% lower in AUC-p@5 while being up to 3x and 3.5x faster in training and prediction respectively. 1-vs-all techniques like 1-vs-all-LS, LEML, DiSMEC, and ProXML don't scale to large-scale datasets due to their linear dependence on the number of labels during training and prediction.

Regression performance is measured through XMAD@5 metric, XReg and XReg-t always have XMAD@5 only up to 10% worse than the best baseline techniques (usually 1-vs-all). XMAD@5 values of XReg are up to 50% better when compared to the other baselines. The XMAD@5 scores for XReg-t are generally higher than XReg due to the re-ranking using tail classifiers which are not regressors but are good generative classifiers which increase tail labels regression values leading to higher XMAD@5. XReg and XReg-t strike a balance between good ranking as well as close to optimal regression scores. For example, in SSA, XReg and XReg-t's ranking performances are as good as Parabel, but their XMAD@5 scores are up to 150% better. This shows that for a similar ranking, XReg and XReg-t result in better regression performance.

Results on labelwise datasets: Table 2 also compares XReg's performance to the baselines on labelwise datasets using the labelwise metrics discussed above. Naive labelwise prediction works for 1-vs-all classifiers (LEML, 1-vs-all-LS, DiSMEC, and XLR) where the complete score matrix is generated through expensive 1-vs-all prediction. This paper proposed a novel labelwise prediction algorithm for XReg making it up to 25x faster in prediction when compared to the 1-vs-all techniques. On both of the movie recommendation datasets (YahooMovie and MovieLens), XReg-t is up to 4% better than the next best baseline in Rating-l@5 and AUC-l@5 ranking metrics. XReg and XReg-t also have the lowest XMAD-l@5 scores for these datasets showing that their regression quality is better than all the baselines while having a better ranking performance. On DSA-130K, XReg-t is up to 2.26% better than DiSMEC in CTR-l@5 metric while being about 120% better in XMAD-l@5 metric. XReg-t has better CTR-l@k and AUC-l@k (see Table 2 and Table 6 in the supplementary material) performances, particularly up to 0.86% better CTR-l@5, when compared to Parabel on DSA-1M while having a 100% improvement in XMAD-l@5. This shows that XReg and XReg-t successfully ensure both good ranking and regression performance in labelwise datasets.

Label filtering: The accurate relevance weight estimates that XReg produces are useful for many downstream tasks as discussed in Section 1. For example, the predicted labels can be filtered to retain only the most relevant labels based on a relevance threshold. This is particularly useful in DSA where the generated dictionaries need to be small in size while also containing the relevant ads for all the queries. Figure 1 shows the Precision-Recall curves for different ways of filtering XReg's label predictions. Here, Precision is the average true relevance of the retained labels and Recall is the ratio of true relevance summed over the retained labels and all labels in the ground truth respectively. "XReg-Rel" refers to the proposed approach of filtering based on XReg's relevance weights. "XReg-TopK" is the commonly used approach of retaining top k highest-scoring labels per each data point. "XReg-Rand" uses randomly generated relevance weights using a uniform random distribution while maintaining the same ranking as in XReg, and applied relevance threshold over them. As observed from the plots, XReg-Rel attains higher precision levels for the same recall levels compared to XReg-TopK and XReg-Rel. Furthermore, XReg-Rel gives finer control over filtering than the other approaches in the high precision, low recall areas. For DSA on Bing, a relevance threshold of 0.05 was found to work well since most ads have quite low CTRs.

Analysis of regression and ranking errors: Table 3 presents the relationship between the XMAD@2k metric and the regression and ranking errors at k. As discussed above in the Section 3, the 2*XMAD@2k error value gives close bound for both the errors at once. As can be seen from Table 3 across all baselines, when 2*XMAD@2k is high, usually either ranking error or regression error is high, and is at worst 2.5x larger than their maximum. On the other hand, the numbers for the popular MAD metric provide much looser bounds for regression and ranking errors. In particular, MAD is significantly high for dense score predictions (like 1-vs-all-LS) 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 increased 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-103K and DSA-1M while Table 5 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 5 compares XReg's performance to the existing DSA ensemble, consisting of BM25 information retrieval based algorithm [25]

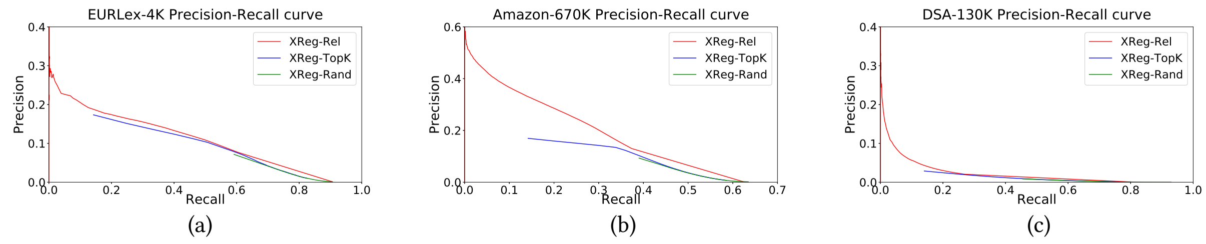

Figure 1: Precision-Recall curves showing that XReg-Rel is consistently better than XReg-TopK and XReg-Rand approaches for label filtering.

Table 3: Ranking and Regression errors@k are better bounded and approximated by 2*XMAD@2k compared to the traditional MAD. $k = 5$, "-p": pointwise, "-l": labelwise and "-t": use of tail classifiers. Please refer to the text for details.

Method	Ranking-error-p @k	Regression-error-p @k	2*XMAD-p @2k	MAD
		EURLex-4K		
PfastreXML	0.1356	0.1964	0.4091	2.5309
Parabel	0.1436	0.2121	0.4345	3.0574
LEML	0.1468	0.2420	0.5210	6.2443
1-vs-all-LS	**0.1342**	0.2277	0.4736	14.7169
RankSVM	0.1377	0.6253	1.2224	37.7822
DiSMEC	0.1607	0.1908	0.4163	46.9473
XReg	0.1394	**0.1894**	**0.3957**	**2.1915**
XReg-t	0.1389	0.2355	0.4906	5.9701
		Wiki10-31K		
PfastreXML	0.4861	**0.0797**	0.8862	9.3561
Parabel	0.4990	0.5016	1.1784	10.1523
LEML	0.5027	0.0859	0.8886	25.5622
1-vs-all-LS	0.4515	0.1628	**0.8492**	34.1409
RankSVM	0.4714	1.1799	2.2960	75.4734
DiSMEC	0.4878	0.4983	1.1519	80.3084
XReg	0.4802	0.1119	0.8938	**6.0104**
XReg-t	**0.4475**	0.2022	0.8741	32.2013

Method	Ranking-error-l @k	Regression-error-l @k	2*XMAD-l @2k	MAD
		YahooMovie-8K		
PfastreXML	0.5665	0.0719	0.8875	8.4893
Parabel	0.5693	**0.0669**	0.8850	**6.3913**
LEML	0.4933	0.2423	0.9571	36.0738
1-vs-all-LS	0.4943	0.2087	0.9288	47.1887
RankSVM	0.4738	0.9168	2.0235	76.6099
DiSMEC	0.4760	0.1909	0.9003	38.0254
XLR	0.6013	0.2379	1.0423	17.0795
XReg	0.4676	0.1708	**0.8847**	6.7809
XReg-t	**0.4664**	0.2039	0.8964	12.1071
		DSA-130K		
PfastreXML	0.0289	0.0149	0.0500	0.3291
Parabel	0.0266	0.0711	0.1230	0.8827
LEML	0.0361	**0.0067**	**0.0486**	0.2828
DiSMEC	0.0265	0.0898	0.1547	4.0878
XLR	0.0402	0.4807	0.9157	34.4290
XReg	0.0259	0.0198	0.0519	**0.2482**
XReg-t	**0.0256**	0.0395	0.0787	0.5862

Table 4: 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 5: 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

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. However, the labelwise XReg boosts the revenue by 58%, improves query 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 is applicable to many real-world problems such as extreme classification, dynamic search advertising, movie recommendation *etc.* XR enables efficient recommendation of most relevant labels for a data point when the relevances are non-binary and when labels and data points range in millions. This paper also developed novel evaluation metrics for eXtreme Regression which serve as a reliable proxy for both regression and ranking errors at the same time. Further, it developed the XReg algorithm along with efficient and optimal pointwise and labelwise prediction algorithms which obtains good performance in terms of both ranking labels and obtaining reliable label relevance estimates. Experiments on benchmark datasets demonstrated that XReg consistently outperforms the state-of-the-art extreme classifiers as well as large-scale regressors and rankers by 100% on the new regression metrics, by 2% on the propensity-scored precision metric popular in extreme classification and by 2.4% on the click-through rate metric used in DSA. Furthermore, XReg is highly efficient with $O(\text{polylog}(L))$ computational and sample complexities in the number of labels L making it suitable for really large-scale tasks like DSA. Deployment of XReg on Bing DSA resulted in a relative gain of 58% in revenue and 27% in query coverage.

REFERENCES

[1] [n. d.]. MovieLens 20M Dataset. https://grouplens.org/datasets/movielens/20m/,.

[2] [n. d.]. Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0. https://webscope.sandbox.yahoo.com/catalog.php?datatype=r,.

[3] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*.

[4] Rohit Babbar and Bernhard Schölkopf. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 721–729.

[5] Rohit Babbar and Bernhard Schölkopf. 2018. Adversarial Extreme Multi-label Classification. *arXiv preprint arXiv:1803.01570* (2018).

[6] Kush Bhatia, Kunal Dahiya, Himanshu Jain, Yashoteja Prabhu, and Manik Varma. [n. d.]. The Extreme Classification Repository: Multi-label Datasets & Code. http://manikvarma.org/downloads/XC/XMLRepository.html

[7] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. 2015. Sparse Local Embeddings for Extreme Multi-label Classification. In *NIPS*.

[8] Tianyou Chai and Roland R. Draxler. 2014. Root mean square error (RMSE) or mean absolute error (MAE).

[9] Yao-Nan Chen and Hsuan-Tien Lin. 2012. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*. 1529–1537.

[10] Minhao Cheng, Ian Davidson, and Cho-Jui Hsieh. 2018. Extreme Learning to Rank via Low Rank Assumption. In *International Conference on Machine Learning*.

[11] Y. Choi, M. Fontoura, E. Gabrilovich, V. Josifovski, M. R. Mediano, and B. Pang. [n. d.]. Using landing pages for sponsored search ad selection. In *WWW 2010*.

[12] M. Cissé, N. Usunier, T. Artières, and P. Gallinari. [n. d.]. Robust Bloom Filters for Large MultiLabel Classification Tasks.

[13] Krzysztof Dembczyński, Wojciech Kotłowski, Willem Waegeman, Róbert Busa-Fekete, and Eyke Hüllermeier. 2016. Consistency of Probabilistic Classifier Trees. In *Machine Learning and Knowledge Discovery in Databases*.

[14] C. Dhanjal, R. Gaudel, and S. Clémençon. 2015. Collaborative filtering with localised ranking. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[15] 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).

[16] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* (2015), 19:1–19:19.

[17] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression. (1999).

[18] D. Hsu, S. Kakade, J. Langford, and T. Zhang. 2009. Multi-Label Prediction via Compressed Sensing. In *Advances in Neural Information Processing Systems*.

[19] Jun Hu and Ping Li. 2018. Collaborative Multi-objective Ranking. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*.

[20] 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*.

[21] H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. 2019. Slice: Scalable Linear Extreme Classifiers trained on 100 Million Labels for Related Searches. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[22] H. Jain, Y. Prabhu, and M. Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[23] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002).

[24] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. 2016. Extreme F-measure Maximization Using Sparse Probability Estimates. In *Proceedings of the International Conference on Machine Learning*.

[25] K. S. Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.* (2000).

[26] Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. 2009. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in neural information processing systems*. 793–800.

[27] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30 (1938).

[28] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[29] Ching-Pei Lee and Chih-Jen Lin. 2014. Large-scale linear ranksvm. *Neural computation* 26, 4 (2014), 781–817.

[30] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. 2014. Local collaborative ranking. In *Proceedings of the International World Wide Web Conference*.

[31] Z. Lin, G. Ding, M. Hu, and J. Wang. 2014. Multi-label Classification via Feature-aware Implicit Label Space Encoding. In *Proceedings of the International Conference on Machine Learning*.

[32] J. Liu, W. Chang, Y. Wu, and Y. Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 115–124.

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

[34] E. L. Mencia and J. Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

[35] P. Mineiro and N. Karampatziakis. 2015. Fast Label Embeddings for Extremely Large Output Spaces. In *ECML*.

[36] A. Niculescu-Mizil and E. Abbasnejad. 2017. Label ‘s for Large Scale Multilabel Classification. In *International Conference on Artificial Intelligence and Statistics*.

[37] Dohyung Park, Joe Neeman, Jin Zhang, Sujay Sanghavi, and Inderjit Dhillon. 2015. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *International Conference on Machine Learning*. 1907–1916.

[38] 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

[39] 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 *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[40] 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 *Proceedings of the International World Wide Web Conference*.

[41] Y. Prabhu and M. Varma. 2014. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[42] B. Pradel, N. Usunier, and P. Gallinari. 2012. Ranking With Non-Random Missing Ratings: Influence Of Popularity And Positivity on Evaluation Metrics. In *Proceedings of the 12th ACM Conference on Recommender Systems*.

[43] S. Ravi, A. Z. Broder, E. Gabrilovich, V. Josifovski, S. Pandey, and B. Pang. [n. d.]. Automatic generation of bid phrases for online advertising. In *WSDM*.

[44] D Sculley. 2009. Large scale learning to rank. (2009).

[45] David Sculley. 2010. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*.

[46] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*.

[47] 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 *Proceedings of the International Conference on Machine Learning*. 3182–3190.

[48] Y. Tagami. 2017. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 455–464.

[49] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *Conference on Learning Theory*.

[50] Geoffrey S Watson et al. 1967. Linear least squares regression. *The Annals of Mathematical Statistics* 38, 6 (1967), 1679–1699.

[51] X. Wei and W. B. Croft. 2006. LDA-based document models for ad-hoc retrieval. In *SIGIR*.

[52] J. Weston, S. Bengio, and N. Usunier. 2011. Wsabie: Scaling Up To Large Vocabulary Image Annotation. In *IJCAI*.

[53] J. Weston, A. Makadia, and H. Yee. 2013. Label Partitioning For Sublinear Ranking. In *Proceedings of the International Conference on Machine Learning*.

[54] Liwei Wu, Cho-Jui Hsieh, and James Sharpnack. 2018. SQL-Rank: A Listwise Approach to Collaborative Ranking. In *ICML*.

[55] C. Xu, D. Tao, and C. Xu. 2016. Robust Extreme Multi-label Learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[56] 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 *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 545–553.

[57] 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 *Proceedings of the International Conference on Machine Learning*.

[58] W. T. Yih, J. Goodman, and V. R. Carvalho. [n. d.]. Finding advertising keywords on web pages. In *WWW 2006*.

[59] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. 2016. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[60] H. F. Yu, P. Jain, P. Kar, and I. S. Dhillon. 2014. Large-scale Multi-label Learning with Missing Labels. In *ICML*.

[61] W. Zhang, D. Wang, G. Xue, and H. Zha. 2012. Advertising Keywords Recommendation for Short-Text Web Pages Using Wikipedia. *ACM TIST* (2012).

[62] Mu 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$(\{s_i\}_{i=1}^{I}$, 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:*

$$0 \le \frac{M}{2} \le XMAD@2k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \le XRMSE@2k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \tag{15}$$

$$\text{with, } M = max(\text{Ranking-error}@k, \text{Regression-error}@k) \tag{16}$$

Proof. The ranking and regression errors are defined as follows

$$\text{Ranking-error}@k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} y_{il} - \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} y_{il} \tag{17}$$

$$\text{Regression-error}@k(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} |\hat{y}_{il} - y_{il}| \tag{18}$$

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

Now, let's prove that $\frac{M}{2} \le$ XMAD@$2k(\hat{\mathbf{y}}, \mathbf{y})$. First, we begin by showing that Ranking-error@$k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \le 2$XMAD@$2k(\hat{\mathbf{y}}, \mathbf{y})$. 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,

$$\frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} y_{il} - \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} y_{il} \tag{19}$$

$$\le \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} y_{il} - \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} y_{il} + \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} \hat{y}_{il} - \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} \hat{y}_{il} \tag{20}$$

$$\le \frac{1}{k}\sum_{l \in S(\mathbf{y}_i, k)} e_{il} + \frac{1}{k}\sum_{l \in S(\hat{\mathbf{y}}_i, k)} e_{il} \quad \text{where,} \ \ e_{il} = |y_{il} - \hat{y}_{il}| \tag{21}$$

$$\le \frac{1}{k}\sum_{l \in S(\mathbf{e}_i, 2k)} e_{il} \tag{22}$$

$$= 2\text{MAD}@2k(\hat{\mathbf{y}}, \mathbf{y}) \tag{23}$$

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

Finally, the XMAD@$2k(\hat{\mathbf{y}}_i, \mathbf{y}_i) \le$ 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*

$$D_{KL}(y_l || \hat{y}_l) \le \sum_{h=1}^{H} s_{lh} D_{KL}(z_{lh} || \hat{z}_{lh}) \ \textit{where} \ s_{lh} = \prod_{\tilde{h}=1}^{h} z_{l(\tilde{h}-1)} \tag{24}$$

Proof. We assume that $0 \log \frac{0}{p} = 0$ where $0 \le p \le 1$. 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.

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

Table 6: XReg outperforms baselines on most of the datasets when evaluated on ranking metrics like WP@k, AUC@k and nDCG@k. "-p": pointwise, "-l": labelwise and "-t": use of tail classifiers.

Method	PSP-p@1 (%)	PSP-p@3 (%)	AUC-p@1 (%)	AUC-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 (%)	AUC-p@1 (%)	AUC-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 (%)	AUC-l@1 (%)	AUC-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 (%)	AUC-l@1 (%)	AUC-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

bounded by the KL-divergence of the corresponding joint distributions.

$$D_{KL}(\mathbb{P}(I_{y_l})||\mathbb{P}(I_{\hat{y}_l})) = D_{KL}(\mathbb{P}(I_{z_{lH}})||\mathbb{P}(I_{\hat{z}_{lH}})) \tag{25}$$

$$\leq D_{KL}(\mathbb{P}(I_{z_{lH}},\cdots,I_{z_{l1}})||\mathbb{P}(I_{\hat{z}_{lH}},\cdots,I_{\hat{z}_{l1}})) \tag{26}$$

By using chain rule of KL-Divergence: (27)

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

$$+\mathbb{P}(I_{z_{l(H-1)}}=1)D_{KL}(\mathbb{P}(I_{z_{lH}}|I_{z_{l(H-1)}}=1)||\mathbb{P}(I_{\hat{z}_{lH}}|I_{\hat{z}_{l(H-1)}}=1)) \tag{29}$$

$$+\mathbb{P}(I_{z_{l(H-1)}}=0)D_{KL}(\mathbb{P}(I_{z_{lH}}|I_{z_{l(H-1)}}=0)||\mathbb{P}(I_{\hat{z}_{lH}}|I_{\hat{z}_{l(H-1)}}=0)) \tag{30}$$

By unvisited node assumption, (31)

$$I_{z_{l(H-1)}}=0 \implies I_{z_{lH}}=0 \text{ and } I_{\hat{z}_{l(H-1)}}=0 \implies I_{\hat{z}_{lH}}=0 \tag{32}$$

hence: (33)

$$= D_{KL}(\mathbb{P}(I_{z_{l(H-1)}},\cdots,I_{z_{l1}})||\mathbb{P}(I_{\hat{z}_{l(H-1)}},\cdots,I_{\hat{z}_{l1}})) \tag{34}$$

$$+\mathbb{P}(I_{z_{l(H-1)}}=1)D_{KL}(\mathbb{P}(I_{z_{lH}}|I_{z_{l(H-1)}}=1)||\mathbb{P}(I_{\hat{z}_{lH}}|I_{\hat{z}_{l(H-1)}}=1)) \tag{35}$$

$$= D_{KL}(\mathbb{P}(I_{z_{l(H-1)}},\cdots,I_{z_{l1}})||\mathbb{P}(I_{\hat{z}_{l(H-1)}},\cdots,I_{\hat{z}_{l1}})) \tag{36}$$

$$+\Big(\prod_{\tilde{h}=1}^{H-1} z_{lh}\Big)\Big(z_{lh}\log\frac{z_{lh}}{\hat{z}_{lh}}+(1-z_{lh})\log\frac{1-z_{lh}}{1-\hat{z}_{lh}}\Big) \tag{37}$$

By recursively applying above simplification (38)
at higher level tree nodes: (39)

$$=\sum_{h=1}^{H} s_{lh}\Big(z_{lh}\log\frac{z_{lh}}{\hat{z}_{lh}}+(1-z_{lh})\log\frac{1-z_{lh}}{1-\hat{z}_{lh}}\Big) \tag{40}$$

Table 7: 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 (%)	AUC@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 (%)	AUC@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 (%)	AUC@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 (%)	AUC@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

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 (15), 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 [13],

$$\max_{l=1}^{L} |y_l - \hat{y}_l| \tag{41}$$
$$\leq \max_{l=1}^{L} \sqrt{\frac{1}{2} D_{KL}(y_l, \hat{y}_l)} \tag{42}$$
$$= \sqrt{\max_{l=1}^{L} \frac{1}{2} D_{KL}(y_l, \hat{y}_l)} \tag{43}$$
$$\text{From (24):} \tag{44}$$
$$\leq \sqrt{\max_{l=1}^{L} \frac{1}{2} \sum_{h=1}^{H} s_{lh} D_{KL}(z_{lh} || \hat{z}_{lh})} \tag{45}$$
$$\leq \sqrt{\frac{1}{2} \sum_{n:z_{n-1}>0} s_n D_{KL}(z_n || \hat{z}_n)} \tag{46}$$
$$\text{where } z_{n-1} \text{ is value in parent of node } n \tag{47}$$
$$\tag{48}$$

□

For good generalization performance, we need a small expected regret with respect to distribution over data point **x**:

$$\mathbb{E}_x \max_{l=1}^{L} |y_l - \hat{y}_l| \tag{49}$$
$$\text{By concavity of square root function:} \tag{50}$$
$$\leq \sqrt{\frac{1}{2} \mathbb{E}_x \sum_{n:z_{n-1}>0} s_n D_{KL}(z_n || \hat{z}_n)} \tag{51}$$

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 **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 **x** density of at least p.

Now, we use the result in [26]. 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 **x** is bounded by norm 1, for any regressor in XReg,

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq \hat{\mathbb{E}}_x s_n D_{KL}(z_n, \hat{z}_n) + 2W\sqrt{\frac{1}{Np(1-k)}} + 2W\Delta \tag{52}$$

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:

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np(1-k)}} + 2W\Delta \tag{53}$$

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

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np(1 - 2\Delta(\sqrt{(\Delta^2 + 1)} - \Delta))}} + 2W\Delta \tag{54}$$

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

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np(1 - 2\Delta(\sqrt{(\Delta^2 + 1)} - \Delta))}} \tag{55}$$
$$+ 2W\sqrt{\frac{1}{2(\sqrt{2} - 1)^2 Np} \log\left(\frac{3L}{\delta}\right)} \tag{56}$$

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

$$\mathbb{E}_x s_n D_{KL}(z_n, \hat{z}_n) \leq 2W\sqrt{\frac{1}{Np}} + 2W\sqrt{\frac{1}{2(\sqrt{2} - 1)^2 Np} \log\left(\frac{3L}{\delta}\right)} \tag{57}$$

From (58),

$$\mathbb{E}_x \max_{l=1}^{L} |y_l - \hat{y}_l| \tag{58}$$
$$\leq \sqrt{\frac{1}{2} \mathbb{E}_x \sum_{n:z_{n-1}>0} s_n D_{KL}(z_n || \hat{z}_n)} \tag{59}$$
$$\text{Since any } \mathbf{x} \text{ has on average } \log L \text{ non-zero labels and} \tag{60}$$
$$\text{since height of the tree is } \log L \tag{61}$$
$$\text{the number of nodes with } z_{n-1} > 0 \text{ for any } \mathbf{x} \text{ is on average } \log^2 L\text{, hence:} \tag{62}$$
$$\leq \sqrt{\frac{\log^2 L}{2}\left(2W\sqrt{\frac{1}{Np}} + 2W\sqrt{\frac{1}{2(\sqrt{2} - 1)^2 Np} \log\left(\frac{3L}{\delta}\right)}\right)} \tag{63}$$
$$\leq \log L \sqrt{\frac{W}{\sqrt{Np}}} \sqrt{1 + \sqrt{5 \log\left(\frac{3L}{\delta}\right)}} \tag{64}$$

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 (58).

$$\max_{n \in S} 2|z_n - \hat{z}_n| \leq \sqrt{\frac{1}{2}\mathbb{E}_x \sum_{n:z_{n-1}>0} s_n D_{KL}(z_n || \hat{z}_n)} \tag{65}$$

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

$$\leq \log^2 L \sqrt{\frac{W}{\sqrt{Np}}} \sqrt{1 + \sqrt{5 \log\left(\frac{3L}{\delta}\right)}} \tag{66}$$