

---

This is an electronic reprint of the original article.  
This reprint may differ from the original in pagination and typographic detail.

Babbar, Rohit; Schölkopf, Bernhard

## Data scarcity, robustness and extreme multi-label classification

*Published in:*  
Machine Learning

*DOI:*  
[10.1007/s10994-019-05791-5](https://doi.org/10.1007/s10994-019-05791-5)

Published: 15/09/2019

*Document Version*  
Publisher's PDF, also known as Version of record

*Published under the following license:*  
CC BY

*Please cite the original version:*  
Babbar, R., & Schölkopf, B. (2019). Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108(8-9), 1329-1351. <https://doi.org/10.1007/s10994-019-05791-5>

---

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.



# Data scarcity, robustness and extreme multi-label classification

Rohit Babbar<sup>1</sup> · Bernhard Schölkopf<sup>2</sup>

Received: 22 October 2018 / Accepted: 19 February 2019 / Published online: 18 March 2019  
© The Author(s) 2019

## Abstract

The goal in extreme multi-label classification (**XMC**) is to learn a classifier which can assign a small subset of relevant labels to an instance from an extremely large set of target labels. The distribution of training instances among labels in XMC exhibits a long tail, implying that a large fraction of labels have a very small number of positive training instances. Detecting tail-labels, which represent diversity of the label space and account for a large fraction (upto 80%) of all the labels, has been a significant research challenge in XMC. In this work, we pose the tail-label detection task in XMC as robust learning in the presence of worst-case perturbations. This viewpoint is motivated by a key observation that there is a significant change in the distribution of the feature composition of instances of these labels from the training set to test set. For shallow classifiers, our robustness perspective to XMC naturally motivates the well-known  $\ell_1$ -regularized classification. Contrary to the popular belief that Hamming loss is unsuitable for tail-labels detection in XMC, we show that minimizing (convex upper bound on) Hamming loss with appropriate regularization surpasses many state-of-the-art methods. Furthermore, we also highlight the sub-optimality of the co-ordinate descent based solver in the LibLinear package, which, given its ubiquity, is interesting in its own right. We also investigate the spectral properties of label graphs for providing novel insights towards understanding the conditions governing the performance of Hamming loss based one-vs-rest scheme vis-à-vis label embedding methods.

**Keywords** Extreme multi-label classification · Large-scale classification · Robustness · Linear classification

---

Editors: Karsten Borgwardt, Po-Ling Loh, Evimaria Terzi, and Antti Ukkonen.

✉ Rohit Babbar  
Rohit.Babbar@aalto.fi

<sup>1</sup> Aalto University, Helsinki, Finland

<sup>2</sup> Max-Planck Institute for Intelligent Systems, Tübingen, Germany

## 1 Introduction

Extreme multi-label classification (**XMC**) refers to supervised learning with a large target label set where each training/test instance is labeled with small subset of relevant labels which are chosen from the large set of target labels. Machine learning problems consisting of hundreds of thousand labels are common in various domains such as annotating web-scale encyclopedia (Prabhu and Varma 2014), hash-tag suggestion in social media (Denton et al. 2015), and image-classification (Deng et al. 2010). For instance, all Wikipedia pages are tagged with a small set of relevant labels which are chosen from more than a million possible tags in the collection. It has been demonstrated that, in addition to automatic labelling, the framework of XMC can be leveraged to effectively address learning problems arising in recommendation systems, ranking and web-advertizing (Agrawal et al. 2013; Prabhu and Varma 2014). In the context of recommendation systems for example, by learning from similar users' buying patterns in e-stores like Amazon and eBay, this framework can be used to recommend a small subset of relevant items from a large collection in the e-store. With applications in a diverse range, designing effective algorithms to solve XMC has become a key challenge for researchers in industry and academia alike.

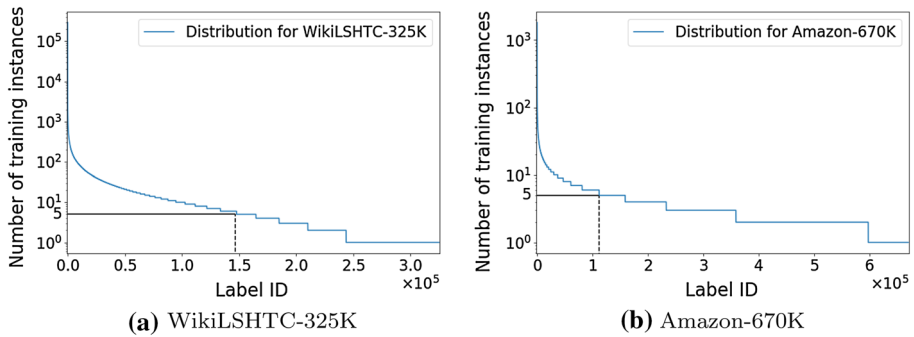
In addition to large number of target labels, typical datasets in XMC consist of a *similar scale* for the number of instances in the training data and also for the dimensionality of the input feature space. For text datasets, each training instance is a sparse representation of a few hundred non-zero features from the input space having dimensionality of the order hundreds of thousand. An example, a benchmark **WikiLSHTC-325K** dataset from the extreme classification repository (Bhatia et al. 2016) consists of 1.7 Million training instances which are distributed among 325,000 labels and each training instance sparsely spans a feature space of 1.6 Million dimensions. The challenge posed by the sheer scale of number of labels, training instances and features, makes the setup of XMC quite different from that tackled in classical literature in multi-label classification (Tsoumakas et al. 2009), and hence renders the *direct and off-the-shelf application* of some of the classical methods, such as Random Forests, Decision Trees and SVMs, non-applicable.

### 1.1 Tail labels

An important statistical characteristic of the datasets in XMC is that a large fraction of labels are tail labels, i.e., those which have very few training instances that belong to them (also referred to as power-law, fat-tailed distribution and Zipf's law). This distribution is shown in Fig. 1 for two publicly available benchmark datasets from the XMC repository,<sup>1</sup> **WikiLSHTC-325K** and **Amazon-670K** datasets, consisting of approximately 325,000 and 670,000 labels respectively. For **Amazon-670K**, only 100,000 out of 670,000 labels have more than 5 training instances in them.

Tail labels exhibit diversity of the label space, and contain informative content not captured by the head or torso labels. Indeed, by predicting well the head labels, an algorithm can achieve high accuracy and yet omit most of the tail labels. Such behavior is not desirable in many real world applications. For instance, in movie recommendation systems, the head labels correspond to popular blockbusters—most likely, the user has already watched these. In contrast, the tail corresponds to less popular yet equally favored films, like independent movies (Shani and Gunawardana 2013). These are the movies that the recommendation system should ideally focus on. A similar discussion applies to search engine development

<sup>1</sup> <http://manikvarma.org/downloads/XC/XMLRepository.html>.



**Fig. 1** Power-law distribution. Y-axis is on log-scale, and X-axis represents the labels ranked according to decreasing number of training instances in them

(Radlinski et al. 2009) and hash-tag recommendation in social networks (Denton et al. 2015), and hierarchical classification (Babbar et al. 2014).

From a statistical perspective, it has been conjectured in the recent works that Hamming loss is unsuitable for detection of tail-labels in XMC (Jain et al. 2016; Bhatia et al. 2015; Prabhu and Varma 2014). In particular, the work in Jain et al. (2016) proposes new propensity-scored loss functions (discussed in Sect. 3) which are sensitive towards the tail-labels by weighing them higher than the head/torso labels. In this work, we refute the above conjecture by motivating XMC from a robustness perspective.

## 1.2 Our contributions

In this work, our contributions are the following :

- *Statistically*, we model XMC as learning in the presence of worst-case perturbations, and demonstrate the efficacy of Hamming loss for tail-label prediction in XMC. This novel perspective stems from the observation that, *for tail labels*, there is a significant variation in the feature composition of instances in the test set as compared to the training set. We thus frame the learning problem as a *robust optimization objective* which accounts for this feature variation by considering perturbations  $\tilde{\mathbf{x}}_i$  for each input training instance  $\mathbf{x}_i$ .
- *Algorithmically*, by exploiting the label-wise independence of Hamming loss among labels, our algorithm is amenable to distributed training across labels. As a result, our forward–backward proximal gradient algorithm can scale upto hundreds of thousand labels for benchmark datasets. Our investigation also shows that the corresponding solver in the LibLinear package (“-s 5” option) yields sub-optimal solutions because of severe under-fitting. Due to its widespread usage in machine learning packages such as scikit-learn, this finding is significant in its own right.
- *Empirically*, our robust optimization formulation of XMC naturally motivates the well-known  $\ell_1$  regularized SVM, which is shown to surpass around one dozen state-of-the-art methods on benchmark datasets. For a Wikipedia dataset with 325,000 labels, we show 20% relative improvement over PFAstreXML and Parabel—leading tree based approaches, and 60% over SLEEC—a leading label embedding method.
- *Analytically*, by drawing connections to spectral properties of label graphs, we also present novel insights to explain the conditions under which Hamming loss might be suited for XMC vis-à-vis label embedding methods. We show that the algebraic connectivity of label graph can be used to explain the variation in the relative performance of

various methods as it varies from small datasets consisting of few hundred labels to the extreme regime consisting of hundreds of thousand labels.

Our robustness perspective also draws connections to recent advances in making deep networks robust to specifically designed perturbations.

## 2 Robust optimization for tail labels

In the extreme classification scenario, the tail labels in the fat-tailed distribution have very few training instances that belong to them. Also, each training instance typically represents a document of about a few hundred words from a total vocabulary of hundreds of thousand or even millions of words. The training instance is, therefore, a sparse representation with only 0.1% or even lesser non-zero features. Due to sparsity, the non-zero features/words in one training instance differ significantly from the other training instance of the same label. Furthermore, since there are only a few training instance in the tail-labels, the union of feature composition of all the training instances for a particular label does not necessarily form a good representation of that label. As a result, the feature composition of the test instance may differ significantly from that of the training instances.

On the other hand, the head labels consist of few tens or even hundreds of training instances. Therefore, the union of words/features which appear in all the training instances is a reasonably good representation of the feature composition of that label. This can also be viewed from the perspective of density of sub-space spanned by the features for a label. For the head labels, this sub-space is much more densely spanned as compared to the tail labels where it is sparsely spanned.

The above observation for the change in the feature composition of tail labels motivates a robustness approach in order to account for the distribution shift. Robustness approach for handling adversarial perturbations to features has been a subject of resurgence particularly in the context of deep learning. In the white-box setting, i.e., with access to deep network and its parameters and their gradients, it has been shown that one can make the deep network predict any desired label for a given image (Goodfellow et al. 2014; Shaham et al. 2015; Szegedy et al. 2013). In the of image classification with deep learning, the benefit of taking a robustness perspective in the presence of less training data has been demonstrated in a very recent work in the upcoming ICLR 2019 (Tsipras et al. 2019). A game-theoretic approach for robustness to feature deletion or addition has also been studied in an earlier work (Globerson and Roweis 2006).

Before, we present our robust optimization framework for handling hundreds of thousand labels in the extreme classification set up, we would like to elucidate distribution shift phenomenon with a few case scenarios.

### 2.1 Case scenarios

The distribution shift is demonstrated for two of the tail labels extracted from the raw data corresponding to **Amazon-670K** dataset [provided by the authors of Liu et al. (2017)]. The instances in the tail label in Table 1 refers to the book titles and editor reviews for books on computer vision and neuroscience, while the instances in the label in Table 2 provides similar descriptions for VHS tapes on action and adventure genre. Note that, in both cases, there is a significant variation in the features/vocabulary within training instances and also from training to test set instances. For instance, in the first example in Table 1, except the word ‘*vision*’, there is not much commonality between features of training instances, and

**Table 1** Training and test instances for label 28503

Training instances	
1.	Vision computational investigation into the human representation and processing of visual information david marr late of the massachusetts institute of technology was the author of many seminal articles on visual information processing and artificial intelligence
2.	Foundations of vision it has much to offer everyone who wonders how this most remarkable of all senses works karen de valois science
Test instance	
1.	Vision science photons to phenomenology this is monumental work covering wide range of topics findings and recent approaches on the frontiers anne princeton university stephen palmer is professor of psychology and director of the institute of cognitive studies at the university of california berkeley

those between training and test instances. Similarly, in the second instance, except for ‘*vhs*’, and the years ‘1942’ and ‘1943’, there is substantial variation in the vocabulary of instances. In other words, even though the *underlying distribution* generating the training and test set is same *in principle*, the typical assumption of the test set coming from the same distribution as the training distribution is violated substantially for tail-labels. Those features which were active in the training set might not appear in test set, and vice-versa.

### 2.2 Robust optimization

Let the training data, given by  $\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$  consist of input feature vectors  $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^D$  and respective output vectors  $\mathbf{y}_i \in \mathcal{Y} \subseteq \{0, 1\}^L$  such that  $\mathbf{y}_{i\ell} = 1$  iff the  $\ell$ -th label belongs to the training instance  $\mathbf{x}_i$ . For each label  $\ell$ , sign vectors  $\mathbf{s}^{(\ell)} \in \{+1, -1\}^N$  can be constructed, such that  $\mathbf{s}_i^{(\ell)} = +1$  if and only if  $\mathbf{y}_{i\ell} = 1$ , and  $-1$  otherwise. The *traditional goal in XMC* is to learn a multi-label classifier in the form of a vector-valued output function  $f : \mathbb{R}^D \mapsto \{0, 1\}^L$ . This is typically achieved by minimizing an empirical estimate of  $\mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}}[\mathcal{L}(\mathbf{W}; (\mathbf{x}, \mathbf{y}))]$  where  $\mathcal{L}$  is a loss function, and samples  $(\mathbf{x}, \mathbf{y})$  are drawn from some underlying distribution  $\mathcal{D}$ , and  $\mathbf{W}$  denotes the desired parameters.

Accounting for feature variations demonstrated in the above examples in Tables 1 and 2, we consider a perturbation  $\tilde{\mathbf{x}}_i \in \tilde{\mathcal{X}} \subseteq \mathbb{R}^D$  for instance  $\mathbf{x}_i$ . The corresponding robust optimization objective hence becomes

$$\min_{\mathbf{W} \in \mathcal{W}} \mathbb{E}[\sup_{\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}} \mathcal{L}(\mathbf{W}; (\mathbf{x} + \tilde{\mathbf{x}}, \mathbf{y}))] \tag{1}$$

The above formulation is intractable to optimize over an arbitrary function class  $\mathcal{W}$  even for convex loss function  $\mathcal{L}$ . For instance, in the context of deep learning methods which optimize over-parameterized non-convex function classes, the above problem in Eq. (1) cannot be solved exactly. Robustness against adversarial perturbations is therefore achieved by employing heuristics such as re-training the network to minimize the loss  $\mathcal{L}(\mathbf{W}; (\mathbf{x} + \tilde{\mathbf{x}}^*(\mathbf{W}), \mathbf{y}))$  instead of Eq. (1) (Szegedy et al. 2013; Goodfellow et al. 2014; Shaham et al. 2015). These adversarial perturbations are obtained by minimizing a linear approximation of the loss using Taylor series  $\tilde{\mathbf{x}}^*(\mathbf{W}) := \arg \max_{\|\mathbf{r}\| \leq \epsilon} \{\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{W}; (\mathbf{x}, \mathbf{y}))^T \mathbf{r}\}$ .

**Table 2** Training and test instances for label 246910

Training instances	
1.	Manhunt in the african jungle vhs 1943 an american secret agent matches wits with nazi agents in casablanca contains 15 episodes
2.	Men vs black dragon vhs 1943 fifteen episodes of 1942 serial showing government agents as they exposed the infamous black dragon society an axis spy ring intent on crippling the war effort
Test instance	
1.	and the vhs 1942 this is classic movie from 1942 with action death defying stunts and breathless cliffhangers

For XMC problems, where  $N$ ,  $D$ , and  $L$  lie in the range  $10^4 - 10^6$ , we restrict ourselves to Hamming loss for the loss function  $\mathcal{L}$  and linear function class for  $\mathcal{W}$ . These have the following statistical and computational advantages: (i) the min-max optimization problem in Eq. (1) can be solved exactly without resorting to Taylor series approximation, (ii) Hamming loss function decomposes over individual labels, thereby enabling parallel training and hence linear speedup with number of cores, (iii) linear methods are computationally efficient and statistically comparable to kernel methods for large  $D$  (Huang and Lin 2016).

We first recall Hamming loss : for predicted output vector  $\hat{\mathbf{y}}$  and the ground truth label vector  $\mathbf{y}$ , it is defined as  $\mathcal{L}_H(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{L} \sum_{\ell=1}^L I[y_\ell \neq \hat{y}_\ell]$ , where  $I[\cdot]$  is the indicator function. For linear function class, the classifier  $f$  is parameterized by  $\mathbf{W} \in \mathbb{R}^{D \times L} := [\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(L)}]$ ,  $\mathbf{w}^{(\ell)} \in \mathbb{R}^D$ . For each label  $\ell$ , by replacing the indicator 0-1 loss function in Hamming loss by hinge loss as its convex upper bound, the weight vector  $\mathbf{w}^{(\ell)}$  for label  $\ell$  with sign vector  $\mathbf{s}^{(\ell)} \in \{+1, -1\}^N$ , is learnt by minimizing the following robust optimization objective (without super-script ( $\ell$ ) for clarity)

$$\min_{\mathbf{w}} \max_{(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_N)} \sum_{i=1}^N \max[1 - \mathbf{s}_i(\langle \mathbf{w}, \mathbf{x}_i - \tilde{\mathbf{x}}_i \rangle), 0] \tag{2}$$

such that  $\tilde{\mathbf{x}}_i \in \tilde{\mathcal{X}}$ . It has been shown in Xu et al. (2009) that if  $\sum_{i=1}^N \|\tilde{\mathbf{x}}_i\| < \lambda'$ , then the objective in Eq. (2) is equivalent to regularizing with the dual norm without considering perturbations, and the corresponding equivalent problem is:

$$\min_{\mathbf{w}} \lambda' \|\mathbf{w}\|_* + \sum_{i=1}^N \max[1 - \mathbf{s}_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0] \tag{3}$$

where  $\|\cdot\|_*$  is the dual norm of  $\|\cdot\|$ , given by  $\|\mathbf{w}\|_* := \sup_{\mathbf{z}} \{\langle \mathbf{z}, \mathbf{w} \rangle, \|\mathbf{z}\| \leq 1\}$ .

From the equivalence between formulations in (2) and (3), the choice of norm in the bound on perturbations ( $\sum_{i=1}^N \|\tilde{\mathbf{x}}_i\| < \lambda'$ ) determines the regularizer in equivalent formulation in Eq. (3). For instance, considering  $\ell_2$ -bounded perturbations leads to  $\ell_2$ -regularized SVM, which is robust to spherical perturbations. It may be recalled that the dual of  $\|\cdot\|_1$ ,  $\|\cdot\|_2$ , and  $\|\cdot\|_\infty$  norms are  $\|\cdot\|_\infty$ ,  $\|\cdot\|_2$ , and  $\|\cdot\|_1$  respectively.

### 2.3 Choice of norm

As shown in the examples in Tables 1 and 2, there can be a *significant variation* in the features' distribution from the training set to the test set instances. We therefore consider the worst

case perturbations in the input, i.e.,  $\|\cdot\|_\infty$  norm. This is given by  $\|\tilde{\mathbf{x}}\|_\infty := \max_{d=1\dots D} |\tilde{\mathbf{x}}_{i_d}|$ . This choice of norm can be motivated from two perspectives.

Firstly, it may be noted that changing the input  $\mathbf{x}$  by small perturbations along each dimension, such that  $\|\tilde{\mathbf{x}}\|_\infty$  remains the same, can change the inner product evaluation of the decision function  $\mathbf{w}^T(\mathbf{x} + \tilde{\mathbf{x}})$  significantly. By accounting for such perturbations in the training data, the resulting weight vector is, therefore, robust to worst-case feature variations. This is not true for perturbations whose  $\|\cdot\|_1$  or  $\|\cdot\|_2$  norms are bounded.

Since the dual of  $\|\cdot\|_\infty$  is  $\|\cdot\|_1$  norm, this choice of norm for the perturbations in formulation (2) leads to the  $\ell_1$  regularized SVM in the optimization problem in (3). As this is known to yield sparse solutions, the above choice of norm shows an equivalence between sparsity and robustness. Both of these are desirable properties in the context of XMC, since we want models which are robust to perturbations due to the tail-label effect, and also sparse which leads to compact models for fast prediction in real-times applications of XMC.

## 2.4 Optimization

However, both  $\ell_1$ -norm and hinge-loss  $\max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0]$  are non-smooth, which is undesirable from the view-point of efficient optimization. In the following theorem, we prove that one can replace hinge loss by its squared version given by  $(\max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0])^2$  for a different choice of the regularization parameter  $\lambda$  instead of  $\lambda'$ . The statistically equivalent problem results in objective function in Eq. (5), which is easier to optimize. The proof of the theorem follows the same technique as in Xu et al. (2010).

**Theorem 1** *The following objective function with hinge loss and regularization parameter  $\lambda'$*

$$\min_{\mathbf{w}} \lambda' \|\mathbf{w}\|_1 + \sum_{i=1}^N \max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0] \quad (4)$$

*is equivalent, upto a change in the regularization parameter, to the objective function below with squared hinge loss for some choice of  $\lambda$*

$$\min_{\mathbf{w}} \lambda \|\mathbf{w}\|_1 + \sum_{i=1}^N (\max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0])^2 \quad (5)$$

**Proof** We first start with a definition. Let  $g(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}$  and  $h(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}$  be two functions. Then  $\mathbf{w}^*$  is called *weakly efficient* if at least one of the following holds, (i)  $\mathbf{w}^* \in \arg \min_{\mathbf{w} \in \mathbb{R}^D} g(\mathbf{w})$ , (ii)  $\mathbf{w}^* \in \arg \min_{\mathbf{w} \in \mathbb{R}^D} h(\mathbf{w})$ , and (iii)  $\mathbf{w}^*$  is Pareto efficient, which means that  $\nexists \mathbf{w}'$  such that  $g(\mathbf{w}') \leq g(\mathbf{w}^*)$  and  $h(\mathbf{w}') \leq h(\mathbf{w}^*)$  with at least one holding with strict inequality.

A standard result from convex analysis states that for convex functions  $g(\mathbf{w})$  and  $h(\mathbf{w})$ , the set of optimal solutions for the weighted sum,  $\min_{\mathbf{w}} (\lambda_1 g(\mathbf{w}) + \lambda_2 h(\mathbf{w}))$  where  $\lambda_1, \lambda_2 \in [0, +\infty)$  and not being zero together, coincides with the set of weakly efficient solutions. This means that the set of optimal solutions of  $\min_{\mathbf{w}} (\lambda' \|\mathbf{w}\|_1 + \sum_{i=1}^N \max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0])$ , where  $\lambda'$  ranges in  $[0, +\infty)$  is the set of weakly efficient solution of  $\|\mathbf{w}\|_1$  and  $\sum_{i=1}^N \max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0]$ . On similar lines, the set of optimal solutions of  $\min_{\mathbf{w}} (\lambda \|\mathbf{w}\|_1 + \sum_{i=1}^N (\max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0])^2)$  where  $\lambda$  ranges in  $[0, +\infty)$  is the set of weakly efficient solution of  $\|\mathbf{w}\|_1$  and  $\sum_{i=1}^N (\max[1 - s_i(\langle \mathbf{w}, \mathbf{x}_i \rangle), 0])^2$ . Since taking the square for non-negatives is a monotonic function, it implies that these two sets are identical, and hence the two formulations given in Eqs. (4) and (5) are statistically equivalent upto change in the regularization parameter.  $\square$



### 2.5 Sub-optimality of Liblinear solver (Fan et al. 2008)

The formulation in Eq. (5) lends itself to easier optimization and an efficient solution is implemented in the Liblinear package (as `-s 5` argument) by using a cyclic co-ordinate descent (CCD) procedure. Liblinear has been included in machine learning packages such as scikit-learn and Cran LibLinear for solving large-scale linear classification and regression tasks. A natural question to ask is - *why not use this solver directly if the modeling of XMC under the worst-case perturbation setting and the resulting optimization problem are indeed correct.*

We applied the CCD based implementation in LibLinear and found that it gives sub-optimal solution. In particular, the CCD solution, (i) underfits the training data, and (ii) does not give good generalization performance. For concreteness, let  $\mathbf{w}_{CCD} \in \mathbb{R}^D$  be the minimizer of the objective function in Eq. (5) and  $opt_{CCD} \in \mathbb{R}^+$  be the corresponding optimal value of the objective value attained. We demonstrate under-fitting by producing a certificate  $\mathbf{w}_{Prox} \in \mathbb{R}^D$  with the corresponding objective function value  $opt_{Prox} \in \mathbb{R}^+$  such that  $opt_{Prox} < opt_{CCD}$ . The construction of the certificate of sub-optimality is obtained by following a proximal gradient procedure in the next section. The inferior generalization performance of Liblinear is shown in Table 4, which among other methods, provides comparison on the test set of the models learnt by CCD and that learnt by our proximal gradient procedure in Algorithm 1.

### 2.6 Certificate construction by proximal gradient

---

**Algorithm 1** Proximal gradient method to optimize objective (5) for learning  $\mathbf{w}_{Prox}$  for label  $\ell$

---

**Require:** Binary training data  $(\mathbf{X}, \mathbf{s})$  and initialize  $\mathbf{w}_0 = \mathbf{0}$

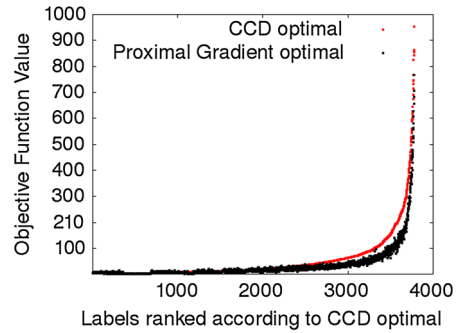
**Ensure:** Learnt weight vector  $\mathbf{w}_{Prox}$  for each label independently

- 1:  $t=0$
  - 2: **while** not converged **do**
  - 3:    $\mathbf{u}_t = \mathbf{w}_t - \gamma_t \nabla_{\mathbf{w}} \sum_{i=1}^N (\max[1 - \mathbf{s}_i \langle \mathbf{w}, \mathbf{x}_i \rangle], 0)^2$  ▷ Forward step
  - 4:    $\mathbf{w}_{t+1} = \arg \min_{\mathbf{w}} \left[ \frac{\lambda \gamma_t}{2} \|\mathbf{w}\|_1 + \frac{1}{2} \|\mathbf{w} - \mathbf{u}_t\|_2^2 \right]$  ▷ Backward step
  - 5:    $t = t + 1$
  - 6: **end while**
  - 7:  $\mathbf{w}_{Prox} = \mathbf{w}_t$  ; return  $\mathbf{w}_{Prox}$
- 

Proximal methods have been effective in addressing large-scale non-smooth convex problems which can be written as sum of a differentiable function with Lipschitz-continuous gradient and a non-differentiable function. We therefore use the forward-backward proximal procedure described in Algorithm 1 to construct the certificate  $\mathbf{w}_{Prox}$  by solving the optimization problem in Eq. (5). The two main steps in the algorithm are in lines 3 and 4. In line 3 (called the forward step), gradient with respect to the differentiable part of the objective is taken, which we denote by  $\mathcal{H}(\mathbf{w})$  for squared hinge loss, i.e.,  $\mathcal{H}(\mathbf{w}) := \sum_{i=1}^N (\max[1 - \mathbf{s}_i \langle \mathbf{w}, \mathbf{x}_i \rangle], 0)^2$ . The step size  $\gamma_t$  in the forward step, which can be thought as inverse of the Lipschitz constant of  $\mathcal{H}'(\mathbf{w}_t)$ , is estimated for a new weight  $\mathbf{w}'$  by starting at a high value and decreasing fractionally until the following condition is satisfied (Bach et al. 2011):

$$\mathcal{H}(\mathbf{w}') \leq \mathcal{H}(\mathbf{w}_t) + \mathcal{H}'(\mathbf{w}_t)^T (\mathbf{w}' - \mathbf{w}_t) + 1/(2\gamma_t) \|\mathbf{w}' - \mathbf{w}_t\|_2^2$$

**Fig. 2** Comparison of  $opt_{CCD}$  and  $opt_{Prox}$  over individual labels for **EUR-Lex** dataset



Line 4 is the backward or proximal step in which minimization problem involving the computation of the proximal operator has a closed-form solution for  $\|\mathbf{w}\|_1$ . It is given by the *soft-thresholding operator*, which for the  $j$ -th dimension at the  $t$ -th iterate is :

$$\mathbf{w}_{j,t+1} = \text{sign}(\mathbf{u}_j) \times \max [ (|\mathbf{u}_j| - \lambda), 0 ] \tag{6}$$

Note the forward–backward procedure detailed in Algorithm 1 learns the weight vector corresponding to each label. Similar to DiSMEC (Babbar and Schölkopf 2017), since the computations are independent for each label, it can be invoked in parallel over as many cores as are available for computation to learn  $\mathbf{W}_{Prox} = [\mathbf{w}_{Prox}^{(1)}, \dots, \mathbf{w}_{Prox}^{(L)}]$ . We call our proposed method  $PR_{OXML}$ , which stands for parallel Robust extreme multi-label classification. The convergence of the forward–backward scheme for proximal gradient has been studied in Combettes and Pesquet (2007).

For **EUR-Lex** dataset, Fig. 2 compares the variation in the optimization objective for the weight  $\mathbf{w}_{CCD}$  learnt by the LibLinear CCD solver and proximal gradient solver  $\mathbf{w}_{Prox}$ . For approximately 90% of the labels, the objective value obtained by Algorithm 1, was lower than that obtained by LibLinear, which in some cases could be as low as half for  $\mathbf{w}_{Prox}$ .

The outlier points with the corresponding optimal value among the highest ones correspond to the labels which mostly head labels. The head-labels are typically linearly non-separable from the others, and hence this leads to significant values for squared hinge loss  $\sum_{i=1}^N (\max[1 - s_i((\mathbf{w}, \mathbf{x}_i)), 0])^2$ . Furthermore, the weight vectors for the head labels are relatively much less sparse compared to the tail labels which further adds to the contribution towards the overall objective value through the regularization term  $\lambda\|\mathbf{w}\|_1$ .

### 3 Experimental analysis

*Dataset description and evaluation metrics* We perform empirical evaluation on publicly available datasets from the XMC repository curated from sources such as Wikipedia and delicious (Mencia and Fürnkranz 2008; McAuley and Leskovec 2013). The detailed statistics of the datasets are shown in Table 3. The datasets exhibit a wide range of properties in terms of number of training instances, features, and labels. **MediaMill** and **Bibtex** datasets are small scale datasets and do not exhibit tail-label behavior. The last column shows the algebraic connectivity of label graph, which measures the degree of connectedness of labels based on their co-occurrences in the training data. The calculation of algebraic connectivity based on algebraic graph theoretic considerations, is described in Sect. 4.

**Table 3** Multi-label datasets from XMC repository

Dataset	# Training (N)	# Features (D)	# Labels (L)	APpL	ALpP	Algebraic connectivity, $\lambda_2(G)$
Mediamill	30,993	120	101	1902.1	4.4	0.46
Bibtex	4,880	1,836	159	111.7	2.4	0.30
EUR-Lex	15,539	5,000	3,993	25.7	5.3	0.22
WikiLSHTC-325K	1,778,351	1,617,899	325,056	17.4	3.2	0.002
Wiki-500K	1,813,391	2,381,304	501,070	24.7	4.7	0.001
Amazon-670K	490,499	135,909	670,091	3.9	5.4	0.0001

APpL and ALpP represent average points per label and average labels per point respectively. Mediamill and Bibtex do not have tail-labels. The algebraic connectivity is calculated in Sect. 4

With applications in recommendation systems, ranking and web-advertizing, the objective of the machine learning system in XMC is to correctly recommend/rank/advertize among the top-k slots. Propensity scored variants of precision@k and nDCG@k capture prediction accuracy of a learning algorithm at top-k slots of prediction, and also the diversity of prediction by giving higher score for predicting rarely occurring tail-labels. As detailed in Jain et al. (2016), the propensity  $p_\ell$  of label  $\ell$ , is related to number of its positive training instances  $N_\ell$  such that  $p_\ell \approx 1$  for head-labels and  $p_\ell \ll 1$  for tail-labels. Let  $\mathbf{y} \in \{0, 1\}^L$  and  $\hat{\mathbf{y}} \in \mathbb{R}^L$  denote the true and predicted label vectors respectively, then the propensity scored variants of  $P@k$  and  $nDCG$  are given by

$$PSP@k(\hat{\mathbf{y}}, \mathbf{y}) := \frac{1}{k} \sum_{\ell \in \text{rank}_k(\hat{\mathbf{y}})} \mathbf{y}_\ell / p_\ell \quad (7)$$

$$PSnDCG@k(\hat{\mathbf{y}}, \mathbf{y}) := \frac{PSDCG@k}{\sum_{\ell=1}^{\min(k, \|\mathbf{y}\|_0)} \frac{1}{\log(\ell+1)}} \quad (8)$$

where  $PSDCG@k := \sum_{\ell \in \text{rank}_k(\hat{\mathbf{y}})} [\frac{\mathbf{y}_\ell}{p_\ell \log(\ell+1)}]$ , and  $\text{rank}_k(\mathbf{y})$  returns the  $k$  largest indices of  $\mathbf{y}$ .

To match against the ground truth, as suggested in Jain et al. (2016), we use  $100 * \mathbb{G}(\{\hat{\mathbf{y}}\}) / \mathbb{G}(\{\mathbf{y}\})$  as the performance metric. For  $M$  test samples,  $\mathbb{G}(\{\hat{\mathbf{y}}\}) = \frac{1}{M} \sum_{i=1}^M \mathbb{L}(\hat{\mathbf{y}}_i, \mathbf{y})$ , where  $\mathbb{G}(\cdot)$  and  $\mathbb{L}(\cdot, \cdot)$  signify gain and loss respectively. The loss  $\mathbb{L}(\cdot, \cdot)$  can take two forms, (i)  $\mathbb{L}(\hat{\mathbf{y}}_i, \mathbf{y}) = -PSnDCG@k$ , and (ii)  $\mathbb{L}(\hat{\mathbf{y}}_i, \mathbf{y}) = -PSP@k$ . This leads to the two metrics which are finally used in our comparison in Table 4 (denoted by  $N@k$  and  $P@k$ ).

### 3.1 Methods for comparison

We compare PR<sub>O</sub>XML against eight algorithms including the state-of-the-art label-embedding, tree-based and linear methods:

#### Label embedding methods

- (I) LEML (Yu et al. 2014)—It learns a low-rank embedding of the label space and is shown to work well for datasets with high label correlation in the presence of moderate number of labels.
- (II) SLEEC (Bhatia et al. 2015)—It learns sparse *locally low-rank* embeddings and captures non-linear correlation among the labels. This method has been shown to scale for XMC problems by applying data clustering as an initialization step.

### Tree-based methods

- (I) **FastXML** (Prabhu and Varma 2014)—This is a scalable tree-based method which partitions the feature space and optimizes vanilla nDCG metric at each node ( $p_\ell = 1$  in equation (8) above).
- (II) **PFastreXML** (Jain et al. 2016)—This method is designed for better classification of tail-labels. It learns an ensemble of **PFastXML** classifier which optimizes propensity-scored nDCG metric in Eq. 8 and Rocchio classifier ([https://en.wikipedia.org/wiki/Rocchio\\_algorithm](https://en.wikipedia.org/wiki/Rocchio_algorithm)) applied on the top 1,000 labels predicted by **PFastXML**. It is shown to out-perform the production system used in Bing Search [c.f. Section 7 in Jain et al. (2016)] and reviewed in detail in Sect. 5.
- (III) **Parabel** (Prabhu et al. 2018)—This is recently proposed method which learns label partitions by a balanced 2-means algorithm, followed by learning one-vs-rest classifier at each node.

### Linear methods

- (I) **PD-Sparse** (Yen et al. 2016)—It uses elastic net regularization with multi-class hinge loss and exploits the sparsity in the primal and dual problems for faster convergence.
- (II) **DiSMEC** (Babbar and Schölkopf 2017)—This is distributed one-vs-rest method which achieves state-of-the-art results on vanilla  $P@k$  and  $nDCG@k$ . It minimizes Hamming loss with  $\ell_2$  regularization with weight pruning *heuristic* for model size reduction.
- (IV) **CCD-L1**—This is the in-built sparse solver as part of the **LibLinear** package which optimizes Eq. 5 using co-ordinate descent.

The proposed method **PROXML** was implemented in C++ on 64-bit Linux system using **openMP** for parallelization. For **PROXML**, the regularization parameter  $\lambda$  was cross-validated for smaller datasets, **MediaMill**, **Bibtex**, and **EUR-Lex** and it was fixed to 0.1 for all bigger datasets. Due to computational constraints in **XMC** consisting of hundreds of thousand labels, keeping fixed values for hyper-parameters is quite standard [c.f. Hyper-parameters setting, Section 7 in Jain et al. (2016), and Section 3 in Bhatia et al. (2015), and Section 5 in Prabhu and Varma (2014)]. For all other approaches, the results were reproduced as suggested in the papers.

## 3.2 Prediction accuracy

The relative performance of various methods on propensity scored metrics for nDCG@k (denoted N@k) and precision@k (denoted P@k) is shown in Table 4. The important observations from these are summarized below:

(A) *Larger datasets* We first look at the results for the datasets falling in the extreme regime such as **Amazon-670K**, **Wiki-500K** and **WikiLSHTC-325K** with large number of labels, and a large fraction of them are tail-labels. Under this regime, **PROXML** performs substantially better than both embedding-schemes and tree-based methods such as **PFastreXML**. For instance, for **WikiLSHTC-325K**, the improvement in P@5 and N@5 over **SLEEC** and **PFastreXML** is approx. 60% and 25% respectively. It is important to note that our method works better on propensity scored metrics than **PFastreXML** even though its training process is optimizing another metric, namely, a convex upper bound on Hamming loss. On the other hand, **PFastreXML** is minimizing the same metric on which the performance is evaluated. Due to its robustness properties, **PROXML** also performs better than **DiSMEC** which also minimizes Hamming loss but employs  $\ell_2$  regularization instead.

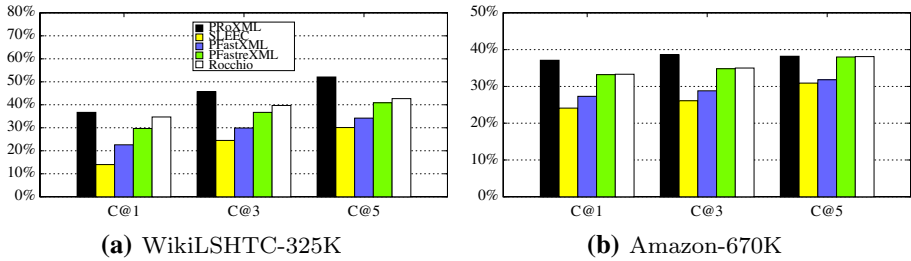
**Table 4** Comparison of N@k for k = 1,3 and 5

Dataset	Proposed approach		Embedding methods		Tree-based methods			Linear methods		
	PROXML		SLEEC	LEMIL	FastXML	PFAstreXML	Parabel	PDSparse	CCD-L1	DISMEC
<b>MediaMill</b>										
N@1	64.3	<b>70.1</b>		66.3	66.1	66.8	66.5	62.2	63.9	66.5
N@3	63.6	<b>72.3</b>		65.7	66.1	66.5	65.9	61.0	62.8	65.5
N@5	62.8	<b>73.1</b>		64.7	65.2	65.7	65.2	57.2	62.0	65.2
P@1	64.3	<b>70.1</b>		66.3	66.6	66.8	63.4	62.2	63.6	66.5
P@3	61.3	<b>72.7</b>		65.1	65.4	66.5	62.8	59.8	60.2	65.1
P@5	60.8	<b>74.0</b>		63.6	64.3	64.7	62.1	54.0	59.7	63.7
<b>Bibtex</b>										
N@1	50.1	51.1		47.9	48.5	<b>52.2</b>	50.8	48.3	49.9	50.2
N@3	52.1	52.9		50.2	51.1	<b>53.6</b>	51.9	48.5	51.6	52.0
N@5	55.1	56.0		53.5	54.3	<b>56.9</b>	54.5	50.7	54.9	55.7
P@1	50.1	51.1		47.9	48.5	<b>52.3</b>	41.2	48.3	49.9	50.2
P@3	52.0	53.9		51.4	52.3	<b>54.3</b>	45.8	48.7	52.1	52.2
P@5	58.3	59.5		57.5	58.8	<b>60.5</b>	54.5	52.9	57.9	58.6
<b>EURLex-4K</b>										
N@1	<b>45.2</b>	35.4		24.1	26.6	43.8	37.7	38.2	37.8	41.2
N@3	<b>47.5</b>	38.8		26.3	32.0	45.2	43.4	40.9	41.6	44.3
N@5	<b>49.1</b>	40.3		27.6	35.2	46.0	46.1	42.8	44.1	46.9
P@1	<b>45.2</b>	35.4		24.1	27.6	43.8	37.7	38.2	37.8	41.2
P@3	<b>48.5</b>	39.8		27.2	35.3	46.4	44.7	42.7	41.6	45.4
P@5	<b>51.0</b>	42.7		29.1	39.9	47.3	48.8	44.8	44.1	49.3

Table 4 continued

Dataset	Proposed approach		Embedding methods		Tree-based methods			Linear methods		
	ProXML		SLEEC	LEML	FastXML	PFastreXML	Parabel	PDSparse	CCD-L1	DiSMEC
<b>Wiki-325K</b>										
<i>N@1</i>	<b>34.8</b>		20.2	3.4	16.3	30.6	28.7	28.3	27.8	29.1
<i>N@3</i>	<b>38.7</b>		22.2	3.6	19.5	31.2	35.2	31.9	30.6	35.9
<i>N@5</i>	<b>41.5</b>		23.3	3.9	21.0	32.0	38.1	33.6	33.9	39.4
<i>P@1</i>	<b>34.8</b>		20.5	3.4	16.5	30.8	28.7	28.3	27.8	29.1
<i>P@3</i>	<b>37.7</b>		23.3	3.7	21.2	31.5	35.0	33.5	30.6	35.6
<i>P@5</i>	<b>41.0</b>		25.2	4.2	23.7	33.0	38.6	36.6	33.9	39.4
<b>Wiki-500K</b>										
<i>N@1</i>	<b>33.1</b>		21.1	3.2	22.5	29.2	28.8	–	29.8	31.2
<i>N@3</i>	<b>35.2</b>		21.0	3.4	21.8	27.6	31.2	–	30.2	33.4
<i>N@5</i>	<b>39.0</b>		20.8	3.5	22.4	27.7	35.5	–	33.1	37.0
<i>P@1</i>	<b>33.1</b>		21.1	3.2	22.5	29.2	28.8	–	29.8	31.2
<i>P@3</i>	<b>35.0</b>		21.0	3.4	21.8	27.6	31.9	–	30.2	33.4
<i>P@5</i>	<b>39.4</b>		20.8	3.5	22.4	27.7	34.6	–	33.1	37.0
<b>Amazon-670K</b>										
<i>N@1</i>	<b>30.8</b>		20.6	2.07	19.3	29.3	28.0	–	19.4	27.8
<i>N@3</i>	<b>31.7</b>		22.6	2.21	22.2	30.4	28.8	–	21.2	28.8
<i>N@5</i>	<b>32.6</b>		24.4	2.35	24.6	31.4	29.4	–	22.7	30.7
<i>P@1</i>	<b>30.8</b>		20.6	2.0	20.2	28.0	27.6	–	19.4	27.8
<i>P@3</i>	<b>32.8</b>		23.3	2.2	23.8	29.5	31.0	–	21.2	30.6
<i>P@5</i>	<b>35.1</b>		26.0	2.4	27.2	30.1	34.1	–	22.7	34.2

**Bold** font denotes the best performing method. PDSparse did not scale to **Wiki-500K**, and **Amazon-670K**



**Fig. 3** Label Coverage for various methods on **WikiLSHTC-325K** and **Amazon-670K** datasets

(B) *Smaller datasets* We now consider smaller datasets which consist of no tail-labels. These include **Mediamill** and **Bibtex** datasets consisting of 101 and 159 labels respectively. Under this regime, embedding based methods SLEEC and LEMML perform better or at par with Hamming loss minimizing methods. As explained in Sect. 4, this is due to high algebraic connectivity of label graphs in smaller datasets, leading to high correlation between labels. This behavior is in stark contrast to datasets in the extreme regime such as **WikiLSHTC-325K** and **Amazon-670K** in which Hamming loss minimizing methods significantly outperform label-embedding methods. The above differences observed in the performance of small-scale problems vis-à-vis large-scale problems are indeed quite contrary to the remarks in recent works [c.f. abstract of Jain et al. (2016)].

(C) *Label coverage* This metric is a measure of the fraction of unique labels correctly predicted by an algorithm. It is shown in Fig. 3 (denoted by C@1, C@3, and C@5) for **WikiLSHTC-325K** and **Amazon-670K**. It is clear that PROXML performs better than state-of-the-art methods in detecting more unique and correct labels. From Fig. 3, it may also be noted that Rocchio classifier applied on top-1000 labels short-listed by PFastXML, does better than PFastXML classifier itself. This indicates that the performance of PFastreXML depends heavily on the good performance of Rocchio classifier, which in turn is learnt from the top labels predicted by PFastXML classifier. On the other hand, our method despite not having any such ensemble effects, performs better than PFastreXML and its components PFastXML and Rocchio.

(D) *Vanilla metrics* The results for vanilla precision@k and nDCG@k (in which the label propensity  $p_\ell = 1, \forall \ell$ ) are shown in the appendix. For these metrics, PROXML performs slightly worse than DiSMEC. However, this is expected since these metrics give equal weight to all the labels. As a result, those methods which are biased towards head-labels tend to perform better, but tend to yield less diverse predictions.

### 3.3 Model size, and training/prediction time

Due to the sparsity inducing  $\ell_1$  regularization, the obtained models are quite sparse and light-weight. For instance, the model learnt by PROXML is 3 GB in size for **WikiLSHTC-325K**, compared to 30 GB for PFastreXML on this dataset. In terms of training time, PROXML uses a distributed training framework thereby exploiting any number of cores as are available for computation. The training can be done offline on a distributed/cloud based system for large datasets such as **WikiLSHTC-325K** and **Amazon-670K**. Faster convergence can be achieved by other methods such as sub-sampling negative examples or warm-starting the optimization with the weights learnt by DiSMEC algorithm to warm-start for faster convergence, via better initialization instead of initializing with an all-zeros solution in Algorithm 1.

Prediction speed is more critical for most applications of XMC which demand low latency in domains such as recommendation systems and web-advertising. The compact model learnt by PROXML can be easily evaluated for prediction on streaming test instances. This is further aided by distributed model storage which can exploit the parallel architecture for prediction. On **WikiLSHTC-325K**, it takes 2 ms per test instance on average which is thrice as fast as SLEEC, and at par with tree-based methods.

## 4 Discussion: what works, what doesn't and why?

We now analyze the empirical results shown in the previous section by drawing connections to spectral properties of label graphs, and determine data-dependent conditions under which Hamming loss minimization is more suited compared to label embedding methods and vice-versa. This section also sheds light on qualitative differences between data properties when one moves from small-scale to the extreme regime.

### 4.1 Algebraic connectivity of label graphs

For the output vectors in the training data  $\{(\mathbf{y}_i)\}_{i=1}^N$  such that  $\mathbf{y}_{i\ell} = 1$  iff the  $\ell$ -th label belongs to the training instance  $\mathbf{x}_i$ , and 0 otherwise. Consider the adjacency matrix  $A(G)$  corresponding to the label graph  $G$ , whose vertex set  $V(G)$  is the set of labels in training set, and the edge weights  $a_{\ell,\ell'}$  are defined by  $a_{\ell,\ell'} = \sum_{i=1}^N \left[ (\mathbf{y}_{i\ell} = 1) \wedge (\mathbf{y}_{i\ell'} = 1) \right]$ , where  $\wedge$  represents the logical and operator. The edge between labels  $\ell$  and  $\ell'$  is weighted by the number of times  $\ell$  and  $\ell'$  co-occur in the training data. By symmetry,  $a_{\ell,\ell'} = a_{\ell',\ell} \forall \ell, \ell' \in V(G)$ . Let  $d(\ell)$  denote the degree of label  $\ell$ , where  $d(\ell) = \sum_{\ell' \in V(G)} a_{\ell,\ell'}$ , and  $D(G)$  be the diagonal degree matrix  $d_{\ell,\ell} = d(\ell)$ . The entries of normalized Laplacian matrix,  $L(G)$  is given by:

$$L_{\ell,\ell'} = \begin{cases} 1 - \frac{a_{\ell,\ell'}}{d_{\ell}} & \text{if } \ell = \ell' \text{ and } d_{\ell} \neq 0 \\ -\frac{a_{\ell,\ell'}}{\sqrt{d_{\ell}d_{\ell'}}} & \text{if } \ell \text{ and } \ell' \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

Let  $\lambda_1(G), \dots, \lambda_L(G)$  be the eigen-values of  $L(G)$ . From spectral graph theory,  $\lambda_2(G) \leq \nu(G) \leq \eta(G)$ , where  $\nu(G)$  and  $\eta(G)$  are respectively the vertex and edge connectivity of  $G$ . i.e. minimum of vertices and edges to be removed from  $G$  to make it disconnected (Chung 1997). Being a lower bound on  $\nu(G)$  and  $\eta(G)$ ,  $\lambda_2(G)$  gives an estimate on the connectivity of the label graph. The higher the algebraic connectivity, the more densely connected the labels are in the graph  $G$ . The last column of Table 3 shows algebraic connectivity for the normalized Laplacian matrix for various datasets. Higher values of algebraic connectivity, indicating high degree of connectivity and correlation between labels, are observed for smaller datasets such as **MediaMill** which consist of only a few hundreds labels. Lower value is observed for datasets in the extreme regime such as **WikiLSHTC-325K**, **WikiLSHTC-500K** and **Amazon-670K**.

*Why Hamming loss works for extreme classification?*

Contrary to the assertions in Jain et al. (2016), as shown in Table 4, Hamming loss minimizing one-vs-rest, which trains an independent classifier for every label, works well on datasets in the extreme regime such as **WikiLSHTC-325K** and **Amazon-670K**. In this regime, there is very little correlation between labels that could potentially be exploited in the first place



by schemes such as LEML and SLEEC. The extremely weak correlation is indicated by crucial statistics shown in Table 3, which include: lower value of the algebraic connectivity of the label graph  $\lambda_2(G)$ , fat-tailed distribution of instances among labels and lower values of average number of labels per instance. The virtual non-existence of correlation indicates that the presence of a given label does not really imply the presence of other labels. It may be noted that though there may be underlying semantic similarity between labels, but there is not enough data, especially for tail-labels, to support that. This inherent separation in label graph for larger datasets leads to better performance of one-vs-rest scheme.

*Why label-embedding is suitable for small datasets ?*

For smaller datasets that consist of only a few hundred labels (such as **MediaMill**) and relatively large value for average number of labels per instance, the labels tend to co-occur more often than for datasets in extreme regime. In this situation, label correlation is much higher that can be easily exploited by label-embedding approaches leading to better performance compared to one-vs-rest approach. This scale of datasets, as is common in traditional multi-label problems, has been marked by the success of label-embedding methods. Therefore, it may be noted that conclusions drawn on this scale of problems, such as on the applicability of learning algorithms or suitability of loss functions for a given problem, may not necessarily apply to datasets in XMC.

*What about  $PSP@k$  and  $PSPnDG@k$  ?*

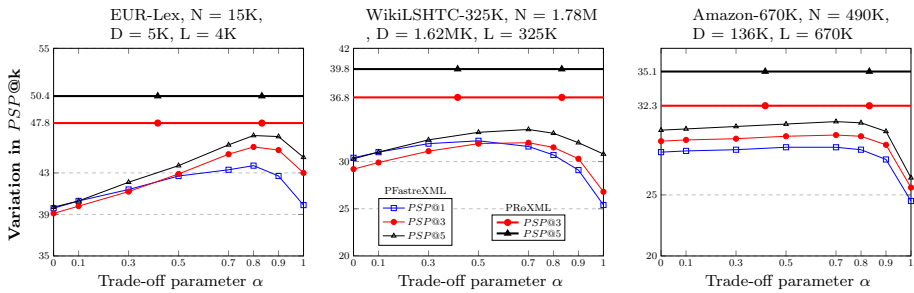
Though  $PSP@k$  and  $PSPnDG@k$  are appropriate for performance evaluation, these may not right metrics to optimize over during training. For instance, if a training instance has fifteen positive labels and we are optimizing  $PSP@5$ , then as soon as it has correctly classified five out of the fifteen labels correctly, the training process will stop trying to change the decision hyper-plane for this training instance. As a result, the information regarding the remaining ten labels is not captured while optimizing the  $PSP@5$  metric. It is possible that at test time, we get a similar instance which has some or all the remaining ten labels which were *not optimized* during training. On the other hand, one-vs-rest which minimizes Hamming loss would try to independently align the hyper-planes for all the fifteen labels until these are separated from the rest. Overall, the model learnt by optimizing is richer compared to that learnt by optimizing  $PSP@k$  and  $PSPnDG@k$ . Therefore, it leads to better performance on  $P@k$  and  $nDG@k$  as well as  $PSP@k$  and  $PSPnDG@k$ , when regularized properly.

## 5 Related work

To handle the large scale of labels in XMC, most methods have focused on two of the main strands, (i) Tree-based methods, and (ii) Label-embedding based methods.

*Label-embedding approaches* assume that the output space has an inherently low rank structure. These approaches have been at the fore-front in multi-label classification consisting of few hundred labels (Bhatia et al. 2015; Hsu et al. 2009; Yu et al. 2014; Tai and Lin 2012; Bi and Kwok 2013; Zhang and Schneider 2011; Chen and Lin 2012; Bengio et al. 2010; Lin et al. 2014; Tagami 2017). However, this assumption can break down in presence of large-scale power-law distributed category systems, and hence leading to high prediction error (Babbar et al. 2013, 2016).

*Tree-based approaches* (Jain et al. 2016; Prabhu and Varma 2014; Si et al. 2017; Niculescu-Mizil and Abbasnejad 2017; Daume III et al. 2016; Jernite et al. 2016; Jasinska et al. 2016) are aimed towards faster prediction which can be achieved by recursively dividing the space of labels or features. Due to the cascading effect, the prediction error made at a top-level



**Fig. 4** Variation of  $PSP@k$  with the trade-off parameter  $\alpha$  for (i) **EUR-Lex**, (ii) **WikiLSHTC-325K**, and (iii) **Amazon-670K** datasets. For PFastreXML,  $\alpha = 0.8$ . On the left, ( $\alpha = 0$ ), represents Rocchio<sub>0,1,000</sub> classifier, and on the right ( $\alpha = 1$ ), represents PFastXML classifier without re-ranking step. PRoXML works better than PFastreXML for all ranges of  $\alpha$  for  $PSP@3$  and  $PSP@5$ .  $PSP@1$  is not shown for clarity, and it is 44.3, 32.4, and 30.3 respectively

cannot be corrected at lower levels. Typically, such techniques trade-off prediction accuracy for logarithmic prediction speed which might be desired in some applications.

Recently, there has been interest in developing distributed linear methods (Babbar and Schölkopf 2017; Yen et al. 2016) which can exploit distributed hardware, and deep learning methods (Liu et al. 2017; Nam et al. 2017). From a probabilistic view-point, bayesian approaches for multi-label classification have been developed in recent works such as Jain et al. 2017; Gaure et al. 2017 and Labeled LDA (Papanikolaou and Tsoumakas 2017). Here, we only discuss PFastreXML in detail here as this is the method designed specifically for better prediction of tail-labels.

### 5.1 PFastreXML (Jain et al. 2016)

PFastreXML is a state-of-the-art tree-based method which outperformed a highly specialized production system for Bing search engine consisting of ensemble of a battery of ranking methods [cf. Section 7 in Jain et al. (2016)]. Learning the PFastreXML classifier primarily involves learning two components, (i) PFastXML classifier—which is an ensemble of trees which minimize propensity scored loss functions, and (ii) a re-ranker which attempts to recover the tail labels missed by PFastXML. The re-ranker is essentially Rocchio classifier, also called the nearest centroid classifier [Equation 7, Section 6.2 in Jain et al. (2016)], which assigns the test instance to the label with closest centroid among the top 1,000 labels predicted by PFastXML. The final score  $s_l$  assigned to label  $l$  for test instance  $\mathbf{x}$  is given by a convex combination of scores PFastXML and the Rocchio classifier for top 1000 label (Equation 8, Section 6.2 in Jain et al. (2016)) as follows:

$$s_l = \alpha \log P_{PFast}(\mathbf{y}_l = 1|\mathbf{x}) + (1 - \alpha) \log P_{Roc}(\mathbf{y}_l = 1|\mathbf{x})$$

For PFastreXML,  $\alpha$  is fixed to 0.8; setting  $\alpha = 1$  gives the scores from PFastXML classifier only and  $\alpha = 0$  gives the scores from Rocchio<sub>0,1,000</sub> classifier only. It may be recalled that, akin to FastXML, PFastXML is also an ensemble of a number of trees, which is typically set to 50. Some of its shortcomings in addition to the relatively poorer performance compared to PRoXML are:

- (I) **Standalone PFastXML**—Fig. 3 shows the variation of  $PSP@k$  of PFastreXML with change in  $\alpha$  which includes the two extremes (PFastXML,  $\alpha = 1$ ) and (Rocchio<sub>0,1,000</sub> classifier,  $\alpha = 0$ ) on three datasets from Table 3. Clearly, the per-

formance of PFastreXML depends heavily on good performance of *Rocchio*<sub>1,000</sub> classifier. It may be recalled that one of the main goals of propensity based metrics and PFastXML was better coverage of tail labels. However, PFastXML itself needs to be supported by the additional *Rocchio*<sub>1,000</sub> classifier for better tail label coverage. To the contrary, our method does not need additional such auxiliary classifier.

- (II) **Need for propensity estimation from meta-data**—To estimate propensities  $p_\ell$  using  $p_\ell := 1 / (1 + Ce^{-A \log(N_\ell + B)})$ , one needs to compute parameters  $A$  and  $B$  from some meta-information of the data-source such as Wikipedia or Amazon taxonomies. Furthermore, it might not even be possible on some datasets to have this side information, in which case the authors in Jain et al. (2016) set it to average of Wikipedia and Amazon datasets, which is quite ad-hoc. Our method does not need propensities for training and hence is also applicable to other metrics for tail-label coverage.
- (III) **Large model sizes**—PFastreXML leads to large model size such as 30 GB (for 50 trees) for **WikiLSHTC-325K** data, and 70 GB (for 20 trees) for **Wiki-500K**. Such large model sizes can be difficult to evaluate for making real-time predictions in recommendation systems and web-advertising. For larger datasets such as **WikiLSHTC-325K**, the model sizes learnt by PROXML is around 3 GB which is an order of magnitude smaller than PFastreXML.
- (IV) **Lots of hyper-parameters**—PFastreXML has around half a dozen hyper-parameters such as  $\alpha$ , number of trees in ensemble, and number of instances in the leaf node etc. Also, there is no reason apriori to fix  $\alpha = 0.8$  even though it gives better generalization performance as shown in Fig. 4. To the contrary, our method has just one hyper-parameter which is the regularization parameter.

## 6 Conclusion

In this work, we motivated the importance of effective tail-label discovery in XMC. We approached this problem from a novel perspective of robustness which has not been considered so far in the domain of XMC. We show that this view-point naturally motivates the well known  $\ell_1$ -regularized SVM, and demonstrate its surprising effectiveness over state-of-the-art methods while scaling to problems with millions of labels. To provide insights into the observations, we explain the performance gain of one-vs-rest scheme vis-à-vis label embedding methods, using tools from graph theory. We hope that synergizing with recent progress on robustness of deep learning methods will open new research avenues for future research.

**Acknowledgements** Open access funding provided by Aalto University. Funding was provided by Aalto Yliopisto (Grant No. TT Package). The authors wish to acknowledge CSC - IT Center for Science, Finland, for computational resources, and Triton cluster team at Aalto University.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## Appendix: Further experimental results

In this section, we present results on the vanilla precision@k (Table 5) and nDCG@k (Table 6). Since these metrics weigh all labels equally, the prediction algorithms have an incentive to predict the head labels. The important observations are summarized below:

**Table 5** Comparison of N@k for k = 1,3 and 5

Dataset	Proposed approach		Embedding methods		Tree-based methods			Linear methods		
	ProXML		SLEEC	LEML	FastXML	PFastreXML	Parabel	PD-Sparse	L1-SVM	DiSMEC
<b>MediaMill</b>										
<i>P@1</i>	86.5	<b>87.2</b>	84.0	84.0	84.2	84.2	83.4	81.8	85.8	87.2
<i>P@3</i>	68.4	<b>73.4</b>	67.2	67.2	67.3	67.3	66.3	62.5	67.4	69.3
<i>P@5</i>	53.2	<b>59.1</b>	52.8	52.8	53.0	53.0	51.7	45.1	52.5	54.1
<b>Bibtex</b>										
<i>P@1</i>	64.4	<b>65.0</b>	62.5	62.5	63.4	62.8	64.4	61.2	64.1	64.5
<i>P@3</i>	39.0	<b>39.6</b>	38.4	38.4	39.2	39.6	38.5	35.9	38.7	39.2
<i>P@5</i>	28.2	28.8	28.2	28.2	28.8	<b>28.9</b>	27.9	25.8	28.4	28.4
<b>EURLex-4K</b>										
<i>P@1</i>	<b>83.4</b>	79.2	63.1	63.1	71.6	75.8	80.3	76.2	80.8	82.2
<i>P@3</i>	<b>70.9</b>	64.3	50.3	50.3	62.0	62.2	68.9	60.9	67.6	68.3
<i>P@5</i>	<b>59.1</b>	52.3	41.6	41.6	51.2	52.0	57.7	49.8	55.1	57.9
<b>Wiki-325K</b>										
<i>P@1</i>	63.6	54.8	19.8	19.8	49.3	56.4	64.7	61.2	60.6	<b>64.9</b>
<i>P@3</i>	41.5	33.8	11.4	11.4	32.7	36.4	<b>42.9</b>	36.3	38.6	42.5
<i>P@5</i>	30.8	24.0	8.4	8.4	24.0	27.0	<b>31.6</b>	28.7	28.5	31.5
<b>Wiki-500K</b>										
<i>P@1</i>	69.0	48.2	41.3	41.3	54.1	59.2	67.5	–	65.3	<b>70.2</b>
<i>P@3</i>	49.1	29.4	30.1	30.1	35.5	40.3	48.7	–	46.1	<b>50.6</b>
<i>P@5</i>	38.8	21.2	19.8	19.8	26.2	30.7	37.7	–	35.3	<b>39.7</b>
<b>Amazon-670K</b>										
<i>P@1</i>	43.5	35.0	8.1	8.1	33.3	28.6	44.0	–	39.8	<b>44.7</b>
<i>P@3</i>	38.7	31.2	6.8	6.8	29.3	24.9	39.4	–	34.3	<b>39.7</b>
<i>P@5</i>	35.3	28.5	6.0	6.0	26.1	22.3	36.0	–	30.1	<b>36.1</b>

**Bold** font denotes the best performing method. PDSparse did not scale to **Wiki-500K**, and **Amazon-670K**

**Table 6** Comparison of N@k for k = 1,3 and 5

Dataset	Proposed approach		Embedding methods		Tree-based methods			Linear methods		
	ProXML	SLEEC	SLEEC	LEML	FastXML	PFastreXML	Parabel	PD-Sparse	L1-SVM	DiSMEC
<b>MediaMill</b>										
N@1	86.5	<b>87.2</b>	84.0	84.0	84.2	84.2	83.4	81.8	85.8	87.2
N@3	77.3	<b>81.5</b>	75.2	75.2	75.4	75.6	74.4	70.2	76.4	78.5
N@5	75.6	<b>79.2</b>	71.8	71.8	72.3	72.4	70.9	63.7	74.7	76.5
<b>Bibtex</b>										
N@1	64.4	<b>65.0</b>	62.5	62.5	63.4	62.8	64.4	61.2	64.1	64.5
N@3	59.2	<b>60.4</b>	58.2	58.2	59.5	60.0	59.3	55.8	59.2	59.4
N@5	61.5	<b>62.6</b>	60.5	60.5	61.7	62.0	61.0	57.3	61.3	61.6
<b>EURLex-4K</b>										
N@1	<b>83.4</b>	79.2	63.1	63.1	71.6	75.8	80.3	76.2	80.8	82.2
N@3	<b>74.2</b>	68.1	53.5	53.5	61.2	65.9	71.8	64.3	71.2	72.5
N@5	<b>68.2</b>	61.6	48.4	48.4	52.3	60.7	66.1	58.7	64.9	66.7
<b>Wiki-325K</b>										
N@1	63.6	54.8	19.8	19.8	49.3	56.4	64.7	61.2	60.6	<b>64.9</b>
N@3	57.4	47.2	14.4	14.4	33.1	50.4	58.3	55.3	55.2	<b>58.5</b>
N@5	57.1	46.1	13.4	13.4	24.4	50.0	58.1	54.7	55.0	<b>58.4</b>
<b>Wiki-500K</b>										
N@1	69.0	48.2	41.3	41.3	54.1	59.2	67.5	–	65.3	<b>70.2</b>
N@3	39.8	22.6	18.7	18.7	26.4	30.1	38.5	–	36.1	<b>42.1</b>
N@5	38.7	21.4	17.1	17.1	24.7	28.7	36.3	–	34.3	<b>40.5</b>
<b>Amazon-670K</b>										
N@1	43.5	35.0	8.1	8.1	33.3	28.6	44.0	–	39.8	<b>44.7</b>
N@3	41.1	32.7	7.3	7.3	33.2	35.8	41.5	–	36.8	<b>42.1</b>
N@5	39.7	31.5	6.8	6.8	30.5	33.2	39.8	–	35.2	<b>40.5</b>

**Bold** font denotes the best performing method. PDSparse did not scale to **Wiki-500K**, and **Amazon-670K**

- DiSMec which employs  $\ell_2$ -regularization tends to perform slightly better than PROXML on all datasets except for EURLex dataset.
- PROXML performs at par with Parabel which uses 2-means to hierarchically partition the data, and then employs  $\ell_2$ -regularized SVM at each node. However, as shown in Table 4 of the main paper, PROXML performs upto 25% better than Parabel on propensity scored metrics.

These results further reinforce the robustness properties introduced by our method, and its impact on effective prediction of tail-label-labels.

## References

- Agrawal, R., Gupta, A., Prabhu, Y., & Varma, M. (2013). Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW* (pp. 13–24).
- Babbar, R., & Schölkopf, B. (2017). Dismec: Distributed sparse machines for extreme multi-label classification. In *WSDM* (pp. 721–729).
- Babbar, R., Partalas, I., Gaussier, E., & Amini, M.-R. (2013). On flat versus hierarchical classification in large-scale taxonomies. In *Advances in Neural Information Processing Systems* (pp. 1824–1832).
- Babbar, R., Metzger, C., Partalas, I., Gaussier, E., & Amini, M.-R. (2014). On power law distributions in large-scale taxonomies. *ACM SIGKDD Explorations Newsletter*, 16(1), 47–56.
- Babbar, R., Partalas, I., Gaussier, E., Amini, M.-R., & Amblard, C. (2016). Learning taxonomy adaptation in large-scale classification. *The Journal of Machine Learning Research*, 17(1), 3350–3386.
- Bach, F., Jenatton, R., Mairal, J., & Obozinski, G. (2011). Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*, 5, 19–53.
- Bengio, S., Weston, J., & Grangier, D. (2010). Label embedding trees for large multi-class tasks. In *Neural information processing systems* (pp. 163–171).
- Bhatia, K., Jain, H., Kar, P., Varma, M., & Jain, P. (2015). Sparse local embeddings for extreme multi-label classification. In C. Cortes, N. D. Lawrence, D.D. Lee, M. Sugiyama, & R. Garnett (Eds.), *NIPS* (pp. 730–738).
- Bhatia, K., Dahiya, K., Jain, H., Prabhu, Y., & Varma, M. (2016). The extreme classification repository: Multi-label datasets and code. <http://manikvarma.org/downloads/XC/XMLRepository.html>.
- Bi, W., & Kwok, J. (2013). Efficient multi-label classification with many labels. In *Proceedings of The 30th international conference on machine learning* (pp. 405–413).
- Chen, Y.-N., & Lin, H.-T. (2012). Feature—aware label space dimension reduction for multi-label classification. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *NIPS* (pp. 1529–1537).
- Chung, F. R. (1997). *Spectral graph theory*. Providence: American Mathematical Society.
- Combettes, P. L., & Pesquet, J.-C. (2007). A douglas-rachford splitting approach to nonsmooth convex variational signal recovery. *IEEE Journal of Selected Topics in Signal Processing*, 1(4), 564.
- Daume III, H., Karampatziakis, N., Langford, J., & Mineiro, P. (2016). Logarithmic time one-against-some. arXiv preprint. [arXiv:1606.04988](https://arxiv.org/abs/1606.04988).
- Deng, J., Berg, A. C., Li, K., & Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? In K. Daniilidis, P. Maragos, & N. Paragios (Eds.), *ECCV* (pp. 71–84). Berlin: Springer.
- Denton, E., Weston, J., Paluri, M., Bourdev, L., & Fergus, R. (2015). User conditional hashtag prediction for images. In *KDD*.
- Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9, 1871–1874.
- Gaure, A., Gupta, A., Verma, V. K., & Rai, P. (2017). A probabilistic framework for zero-shot multi-label learning. In *UAI*.
- Globerson, A., & Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In W. W. Cohen & A. Moore (Eds.), *Proceedings of the 23rd international conference on machine learning* (pp. 353–360). ACM.
- Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint. [arXiv:1412.6572](https://arxiv.org/abs/1412.6572).
- Hsu, D., Kakade, S., Langford, J., & Zhang, T. (2009). Multi-label prediction via compressed sensing. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems* (pp. 772–780).

- Huang, H.-Y., & Lin, C.-J. (2016). Linear and kernel classification: When to use which? In *SDM* (pp. 216–224). SIAM.
- Jain, H., Prabhu, Y., & Varma, M. (2016). Extreme multi-label loss functions for recommendation, tagging, ranking and other missing label applications. In *KDD*.
- Jain, V., Modhe, N., & Rai, P. (2017). Scalable generative models for multi-label learning with missing labels. In *ICML*.
- Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfanschmidt, K., Klerx, T., & Hüllermeier, E. (2016). Extreme f-measure maximization using sparse probability estimates. In *ICML*.
- Jernite, Y., Choromanska, A., Sontag, D., & LeCun, Y. (2016). Simultaneous learning of trees and representations for extreme classification, with application to language modeling. arXiv preprint. [arXiv:1610.04658](https://arxiv.org/abs/1610.04658).
- Lin, Z., Ding, G., Hu, M., & Wang, J. (2014). Multi-label classification via feature-aware implicit label space encoding. (pp. 325–333).
- Liu, J., Chang, W.-C., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *SIGIR* (pp. 115–124). ACM.
- McAuley, J., & Leskovec, J. (2013). Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys* (pp. 165–172). ACM.
- Mencia, E. L., & Fürnkranz, J. (2008). Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML-PKDD*.
- Nam, J., Mencia, E. L., Kim, H. J., & Fürnkranz, J. (2017). Maximizing subset accuracy with recurrent neural networks in multi-label classification. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 5413–5423).
- Niculescu-Mizil, A., & Abbasnejad, E. (2017). Label Filters for large scale multilabel classification. In *AIS-TATS, proceedings of machine learning research*. (pp. 1448–1457). Fort Lauderdale.
- Papanikolaou, Y., & Tsoumakas, G. (2017). Subset labeled l1 for large-scale multi-label classification. arXiv preprint. [arXiv:1709.05480](https://arxiv.org/abs/1709.05480).
- Prabhu, Y., & Varma, M. (2014). Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD*, (pp. 263–272). ACM.
- Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., & Varma, M. (2018). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*.
- Radlinski, F., Bennett, P. N., Carterette, B., & Joachims, T. (2009). Redundancy, diversity and interdependent document relevance. In *ACM SIGIR Forum*.
- Shaham, U., Yamada, Y., & Negahban, S. (2015). Understanding adversarial training: Increasing local stability of neural nets through robust optimization. arXiv preprint. [arXiv:1511.05432](https://arxiv.org/abs/1511.05432).
- Shani, G., & Gunawardana, A. (2013). Tutorial on application-oriented evaluation of recommendation systems. *AI Communications*, 26(2), 225–236.
- Si, S., Zhang, H., Keerthi, S. S., Mahajan, D., Dhillon, I. S., & Hsieh, C.-J. (2017). Gradient boosted decision trees for high dimensional sparse output. In *ICML*.
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint. [arXiv:1312.6199](https://arxiv.org/abs/1312.6199).
- Tagami, Y. (2017). *Annexml: Approximate nearest neighbor search for extreme multi-label classification*. In *KDD*, ACM.
- Tai, F., & Lin, H.-T. (2012). Multilabel classification with principal label space transformation. *Neural Computation*, 24(9), 2508–2542.
- Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2019). Robustness may be at odds with accuracy. In *International conference on learning representations*. <https://openreview.net/forum?id=SyxAb30cY7>.
- Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). Mining multi-label data. In *Data mining and knowledge discovery handbook*, Springer.
- Xu, H., Caramanis, C., & Mannor, S. (2009). Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10, 1485–1510.
- Xu, H., Caramanis, C., & Mannor, S. (2010). Robust regression and lasso. *IEEE Transactions on Information Theory*, 7(56), 3561–3574.
- Yen, I. E., Huang, X., Ravikumar, P., Zhong, K., & Dhillon, I. S. (2016). Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*.

- Yu, H.-F., Jain, P., Kar, P., & Dhillon, I. (2014). Large-scale multi-label learning with missing labels. In *ICML* (pp. 593–601).
- Zhang, Y., & Schneider, J. (2011). Multi-label output codes using canonical correlation analysis. (pp. 873–882).

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.