KHOURY COLLEGE OF COMPUTER SCIENCES
NORTHEASTERN UNIVERSITY

PhD Thesis Proposal

Efficiency and Effectiveness in Large-Scale Learning

by

Bingyu Wang

Supervisor: **Javed A. Aslam**
Committee Members: **Javed A. Aslam**
Virgil Pavlu
Jennifer G. Dy
Helen H. Suh

May, 2019

Abstract

Efficiency and Effectiveness in Large-Scale Learning

In the past decade, the data have grown faster than ever across many domains. For example, in survival analysis, there are more than 60 million Medicare beneficiaries across 40 thousand ZIP Code areas in United States from 2000 to 2012, which is up to 5.7 billion person-months of follow-up. In multi-label classification, Wikipedia data contains more than 500 thousand labels, millions of features and instances. For many of such datasets, machine learning models are facing unprecedented challenges associated with effectiveness and efficiency in both time and memory. This thesis aims to develop learning models that scale well on large data while being able to maintain or even increase its level of performance based on the inherent structures of the dataset and learning algorithms. By working on the following key questions for each model: 1) how to adapt to the intrinsic structure of the dataset and 2) how to take into account the special design of the learning formula and algorithm, we develop state-of-the-art algorithms for both regression and classification problems, and scale such algorithms well on multiple real-world datasets, such as millions of Medicare enrollees in survival analysis, Wikipedia articles and Amazon products categorization in multi-label classification.

Contents

Chapter 1

Introduction

Machine learning (ML) has grown rapidly in popularity in the past two decades, and is almost everywhere nowadays. By employing ML algorithms, computer is able to detect patterns and make decisions based on the historical and real-time data. It is obvious that ML has a significant impact on a variety of industries and researches. The field of ML has improved companies' operating efficiency and profitability [1], and also has a broad range of effects across sundry disciplines, including but not limited to public health, policy, biology, finance, neuroscience and etc., that utilize ML to analyze and learn from data [2, 3]. Furthermore, accompanied with the increasing business scale and data accumulation, we are witnessing the big data era and some of us have to run analysis to obtain useful insights and make right decisions from such large datasets. With abundant data, on the one hand, ML algorithms are more guaranteed to extract underlying patterns and generalize better; on the other hand, it becomes harder for many ML algorithms to tackle the computationally tractable and underperforming issues.

Motivated by the new opportunities and challenges with the big data ascent, we aim to develop 'smart' learning techniques, efficiently and effectively, for large-scale data to obtain underlying knowledge, trends and patterns. It is worth emphasizing that we focus on two most common supervised learning tasks: *classification* and *regression*. More specifically, we address the large-scale data learning problems by follows:

- Utilizing the inherent structures of the given data, such as compactness and sparsity;
- Exploiting the intrinsic properties of the learning objective and turning it to tractable learning process, such as class imbalance and mixture structures.
- Further improving performance by designing a faster multi-label wrapper on top of single-label learning methods.

Following the above key ideas, we develop such 'smart' machine learning algorithms for two main real-world applications: **Multi-label classification** in text categorization and image object recognition, and **regression in Survival Analysis** to estimate the association between air pollution and mortality in epidemiological study. Since those two problems are totally disparate domains and the methodologies are different as well, we split them into two chapters for making the content more reader-friendly. Extreme Multi-label Classification and Survival Analysis can be found in Chapter 2 and Chapter 3 respectively. For each application, we introduce its background, along with its essential challenges in the task with large-scale

dataset, and also review current popular state-of-the-art methods that may overcome partial of those challenges. In the end of each chapter, we present our key ideas and contribution, which further overcomes the challenges and maximizes the benefit for algorithms.

Chapter 2

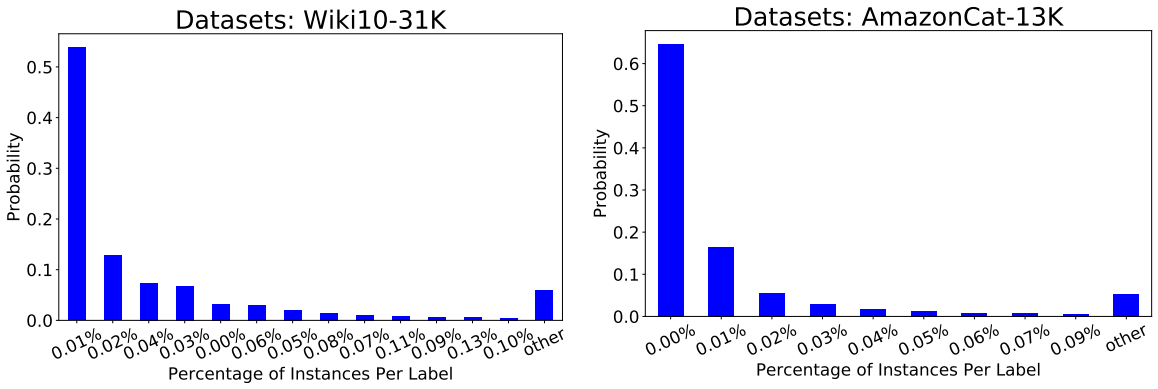
Extreme Multi-label Classification

In multi-label classification (**MLC**) [4, 5], a classifier is trained on instances, each of which is associated with a set of labels, and then is used to predict the label sets of unseen instance with known labels. The difference between multi-label classification and binary / multi-class classification is that in the later problems, the number of labels is restrict to one, thus labels are mutually exclusive, whereas in MLC tasks some of labels can be mutually exclusive and others may be highly correlated. Multi-label classification, as one of the most important machine learning tasks, is omnipresent in real-world applications, especially in Internet industries. For example, in Wikipedia, each article is tagged by several topics; for any recommendation system, one is expected to predict the most relevant items (songs, movies, news and products) for a user; in the movie / television database system, one film is associated with multiple genres; in an image object detection, more than one object can be identified. In these classification tasks, labels often exhibit complex dependencies: for example, in movie genres, *Documentary* and *Sci-Fi* are usually mutually exclusive movie genres, while *Horror* and *Thriller* are typically highly correlated. Therefore, predicting labels independently fails to capture these dependencies and suffers suboptimal performance [4, 6, 7]. In the early stage of solving MLC, several methods, which only focus on capturing label dependencies, have been proposed, including Conditional Random Fields (CRF) [6, 8], Classifier Chain (CC) [9, 10] and several neural networks, such as Structured Prediction Energy Networks (SPEN) [11], and Canonical Correlated AutoEncoder (C2AE) [12]. However, these methods are typically only work well on small-to-medium scale multi-label datasets due to their sophisticated model structures for capturing correlations between labels.

Because the ubiquity of MLC in Internet industries, accompanied with the increasing business scale / scope and data accumulation, extreme multi-label classification (**XML**) is more and more on demand. XML is such a multi-label classification task in which the size of **features**, **instances** and **labels** are very huge, often on the order of thousands to millions [13, 14]. As a side note, it is assumed that either all three (a large number of features, instances and labels) or any one of them needs to be met for multi-label classification as a XML problem. For example, as a online movie database, IMDb[1] contains approximately 5.3 million titles (including episodes) by 2018[2], while the label vocabulary (movie genres) size is still under a hundred. Another example of Wikipedia multi-label data with social tags [13], wherein the tags are aggregated by different users in a collaborative way, includes thirty

[1] `www.imdb.com`

[2] `https://en.wikipedia.org/wiki/IMDb`



(a) Wiki10-31K (b) AmazonCat-13K

Figure 2.1: Label Imbalance on Wiki10-31K and AmazonCat-13K Datasets. The x-axis presents the percentage of instances in which each label is present. y-axis stands for the probability of labels located in that bar. For example, the first bar of Wiki10-31K plot says that more than half labels are present in 0.01% documents, and the second bar says that more than 10% labels are only present in 0.02% documents.

thousand labels tagged by users but only twenty thousand instances available. The number of features, for these two examples and general text categorization, is usually the vocabulary size, varying from thousands to millions. Such large scale datasets require efficient and feasible multi-label methods.

With the increasing scale of multi-label datasets, efficiency and feasibility become not the only challenge for learning algorithms. A series of issues of data quality have also been discovered, including data complexity, label noise and imbalance etc. Besides, with the increasing significance of large-scale multi-label in different domains, people apply multi-label classifiers to solve their specific tasks, leading to a diverse array of evaluation metrics. By investigating all those challenges of utilizing extreme multi-label classification for practitioners, we lay out five fundamental challenges for XML algorithms:

1. **Efficiency and Feasibility.** With rapid data accumulation, whether a multi-label classifier is feasible and efficient enough turns to be the essential challenge above others. For example, Youtube-8M[3] data has millions of data points, billions of audio / visual features and roughly 4 thousand classes. Delicious-200K [15], a social bookmark dataset, contains 300 thousand instances, 780 thousand features and 200 thousand labels. Training such large dataset, even with the simplest method *Binary Relevance* (BR: one-versus-rest), gets to be intractable with linear computational cost in $\#data \times \#features \times \#labels$ [16] . Even though one can efficiently train BR model, he / she may probably encounter memory and storage issues for the linear complexity of model in $\#features \times \#labels$, especially when both the number of features and labels are on the sheer magnitude. One simple but effective way is to apply feature reduction techniques to compress model, leading to sparse model solution, which can be utilized to in turn accelerate training procedure [16, 17].

2. **Label Dependencies.** The second major challenge of XML is to capture the label de-

[3] `https://research.google.com/youtube8m/index.html`

pendencies. Usually, label sets for given instance exhibit complex dependency structures, for instance including mutually exclusive relation (*Documentary* vs *Sci-Fi*) and strong dependence (*Horror* vs *Thriller*) in movie genres. So independent predictions per label (Binary Relevance method) is unlikely to work well [4, 6, 7]. Recently, researchers in machine learning are formalizing these dependencies explicitly or implicitly, aiming to design algorithms that are either provably or practically effective to improve the overall performance [18]. It is worth noting that most of the proposed algorithms estimate the label dependencies from data during the *training* process [6, 7, 10, 11], which is the major efficiency bottleneck and is also why those complicated algorithms are only suitable to small-to-medium scale multi-label datasets. On the other hand, in order to make algorithms feasible for large scale datasets, machine-learning practitioners normally neglect the label dependencies and train classifiers for labels independently [16, 19–22], however, yielding to suboptimal performance.

3. **Label Noises.** Label noises are frequently observed in extreme multi-label data [23–25], especially for large collection of labels. Since the label vocabulary is large and mistakes by human annotators are inevitable, it is quite common for an annotator to miss some relevant tags out of a large collection of labels. And another situation for a human annotator, but less common, is to tag some invalid labels. As far as we are concerned in this thesis, the most frequent multi-label noises comprise both of these two noises or either one of them. With noise labeling data, sometimes when the classifier's prediction (which might be correct) disagrees with the annotation, one can potentially assign an unbounded penalty to the classifier during training procedure.

4. **Label Imbalance.** Although the label vocabulary is large, typically each instance only matches a few labels. In other words, some labels are irrelevant to more instances than other labels, called label imbalance, demonstrated in Figure 2.1. The label imbalance problem exists in both small-to-medium scale and large scale multi-label data. However, most of the current multi-label classifiers ignore such issue, leading to performance degradation [26, 27].

5. **Task-Specific Metric.** There exist many different evaluation metrics for multi-label classification, such as hamming loss, subset accuracy, F-measure, ranking-based metrics and so on [5]. First, hamming loss (HL), a fraction of the wrong labels to the total number of labels, is often optimized by binary relevance method [7, 18]. Since the optimal classifier for HL simply predicts each label independently, it fails to capture instinct label dependencies in multi-label datasets. Besides, another major disadvantage of HL is that it totally ignores label imbalance. As mentioned above, each instance often matches a few labels out of huge number of classes, thus simply predicting each instance an empty set could also achieve a decent hamming loss. Although this is a statement of fact, most of current popular and efficient extreme multi-label classifiers are essentially binary relevance methods [16, 19–22]. On the other hand, the subset accuracy measure, aiming for label dependencies, gives for each instance a score of 1 if the exact label set is predicted and 0 otherwise, and it is usually optimized by maximum likelihood over sets. Typically, methods for optimizing the likelihood over label sets do not scale to large datasets due to the exponential number of potential label subsets. Furthermore, with large number of label candidates, matching the exact

label set is a insoluble task. Therefore, in practice, the F-measure, which gives partial rewards for subset predictions based on overlap with the correct subset, is much better suited for many multi-label tasks than strict subset accuracy. For example, in a medical note, a patient may present with multiple illness or undergo a procedure with multiple billing codes; predicting five out of six codes correctly is a considerable help to medical billing systems. At last, the ranking-based metrics (e.g. precision and nDCG [14]), which are based on the fraction of correct predictions in the top predicted scoring labels, have also been heavily favored in industries, especially for recommendation systems.

For a better story structure, we divide the work into two parts based on these two specific tasks: set prediction and ranking, since they involve different related works, measure metrics and methodologies. We will first introduce set prediction related work, our contribution and future work, and then focus on ranking related tasks.

2.1 Label Set Prediction

For tag/label set prediction task, the classifier is asked to return all relevant labels for a given document. This is often more difficult than ranking tasks, as the classifier not only needs to compute the relevance of each label w.r.t the given instance, but also needs to make a final decision on how many labels and which labels to be included in the predicted set. This decision in general cannot be simply made by picking the top K labels with the highest relevance scores, as the number of relevant labels per instance varies a lot and thus this prediction strategy may be suboptimal for some evaluation metrics, like set accuracy and F-measure.

2.1.1 Notation

Formally, in a multi-label classification problem, we are given a set of label candidates $\mathcal{Y} = \{1, 2, \ldots, L\}$. The dataset consists of features and labels: $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$, wherein N is number of data, and each instance $\boldsymbol{x} \in \mathbb{R}^D$ (D is the feature dimension) matches a label subset $\boldsymbol{y} \subseteq \mathcal{Y}$, which can be written as a binary vector $\boldsymbol{y} = \{0, 1\}^L$, with each bit y_l representing the presence or absence of label l. Given such dataset, our goal is to build a multi-label classifier $\boldsymbol{y} = C(\boldsymbol{x})$: $\mathbb{R}^D \to \{0, 1\}^L$, mapping an instance to a subset of labels. And the size of label subset $\boldsymbol{y}$ can be of arbitrary integer number, written as $|\boldsymbol{y}| \in [0, L]$.

One common way to evaluate the multi-label classifier is called **set accuracy**, which measures the predicted label set $\boldsymbol{y}$ exactly matches the ground truth set $\boldsymbol{y}$:

$$Acc(\boldsymbol{y}, \boldsymbol{y}') = I(\boldsymbol{y}, \boldsymbol{y}') \tag{2.1}$$

Set accuracy is often used when label size is small and predicting label set is desired. For example in Movie genre prediction, predicting all the relevant movie genres is adopted. However, set accuracy, exactly matching ground truth set, becomes impractical when label size is getting larger. In this case, **F-measure** is often more favored by practitioners than other metrics. Thus, in our tag/label set prediction tasks, we focus on optimizing F-measure and scaling it well on large scale dataset.

2.1.2 Related Work for Optimizing F-Measure

It has not escaped our notice that most of current extreme multi-label classifiers ignores label dependencies but can be trained for each label independently in parallel, and thus are computationally efficient in practice. However, ignoring the label dependencies limits the prediction accuracy, especially when the desired measure metric is not decomposable over individual labels. In practice, one of such non-decomposable metrics is **instance-F1**, assigning partial credits for subset predictions based on overlap with the correct subset and handling label imbalance well. Instance-F1 has been widely employed for tag/label prediction in industries, such as the Yelp business categorization [28] and the Greek Media [29]. The **F1-measure** for each instance is defined as

$$F(\boldsymbol{y}, \boldsymbol{y}') = \frac{2\sum_{l=1}^{L} y_l y'_l}{\sum_{l=1}^{L} y_l + \sum_{l=1}^{L} y'_l} \tag{2.2}$$

which is the harmonic mean between precision and recall. The reported instance-F1 is the average of F1-measure values over test instances. It is worth noting that **instance-F1** is different from **macro-averaged F-measure (macro-F1)**, which computes the F1-measure for each binary label across the test set first and then averages across labels. Since macro-F1 is decomposable over labels, obviously, optimizing macro-F1 suffices to maximize the binary F1-measure for each label separately [30, 31]. In this proposal, we only focus on **instance-F1** measure.

There exist a few methods which explicitly take into account the F1 metric during training [32–34], but most of the popular methods that provide a joint estimation in the form of $p(\boldsymbol{y}|\boldsymbol{x})$ are trained by standard maximum likelihood estimation without considering F1 metric as objective. For such methods, it is still possible to use an F1 optimal prediction strategy post-training, that is, output $\boldsymbol{y}^*$ which maximizes the expected F1 score:

$$\begin{aligned} \boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}'} \boldsymbol{E}_{\boldsymbol{y}\sim p(\boldsymbol{y}|\boldsymbol{x})} \left[F(\boldsymbol{y}, \boldsymbol{y}')\right] \\ &= \arg\max_{\boldsymbol{y}'} \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) F(\boldsymbol{y}, \boldsymbol{y}') \end{aligned} \tag{2.3}$$

Besides, for most of multi-label classifiers, training time is usually the bottleneck, thus post-training strategy is more attractive for extreme multi-label classification tasks. The General F-measure Maximizer (**GFM**) [35] is first proposed to find the instance-F1 optimal prediction ($\boldsymbol{y}^*$) for a given instance based on probability estimations. However, the GFM algorithm does not work directly with a joint estimation $p(\boldsymbol{y}|\boldsymbol{x})$, but rather, some L^2 marginal distributions (defined in Eq (2.4)), which can be read as "the probability of the given instance having s relevant labels and y_l is one of them".

$$p(y_l = 1, |\boldsymbol{y}| = s \mid \boldsymbol{x}), \forall l, \forall s \in \{1, \ldots, L\} \tag{2.4}$$

The original paper [35] proposes two ways of obtaining these L^2 marginals (probabilities per instance): I) a model which directly estimates L^2 marginals from data, and II) the use of a probabilistic joint classifier/estimator $p(\boldsymbol{y}|\boldsymbol{x})$ and sampling to generate the required L^2 probabilities. However, we find that option I) is very difficult, perhaps unsolvable although it is indeed appealing as a theoretical exercise; the option II) sampling method is neither

Algorithm 1 General F-Measure Maximizer with Support Inference

Input: support sets $\boldsymbol{\mathcal{Y}}$ and joint estimation $p(\boldsymbol{y}|\boldsymbol{x})\ \forall \boldsymbol{y} \in \boldsymbol{\mathcal{Y}}$
Marginalize $p(\boldsymbol{y}|\boldsymbol{x})$ to obtain $P \in \mathbb{R}^{L\times L}$ matrix, wherein $p_{ls} = p(y_l = 1, |\boldsymbol{y}| = s|\boldsymbol{x})$
Compute $p(\boldsymbol{y} = \mathbf{0}|\boldsymbol{x})$
Define $W \in \mathbb{R}^{L\times L}$ matrix with $w_{s,k} = \frac{2}{s+k}$
Compute $\Delta = PW \in \mathbb{R}^{L\times L}$
for $s = 1, 2, ..., L$ **do**
    Compute $\boldsymbol{y}^s$ with $y_l^s = 1$ if $l \in rank_s(\Delta_s)$, otherwise $y_l^s = 0$ for $l = 1 \ldots, L$
    Compute $\boldsymbol{E}[F(\boldsymbol{y}, \boldsymbol{y}^s)] = \sum_{l=1}^{L} y_l^s \Delta_{ls}$
Let $\boldsymbol{E}[F(\boldsymbol{y}, \boldsymbol{y}^0)] = p(\boldsymbol{y} = \mathbf{0}|\boldsymbol{x})$
Output: $\mathbf{y}^* = \arg\max_{0\le s\le L} \boldsymbol{E}[F(\boldsymbol{y}, \boldsymbol{y}^s)]$

efficient nor effective, for large number of labels and in particular low-confidence (flat) joint that allows probability mass to spread over many label combinations.

There are several approaches that seek to optimize the F-measure directly during training. [34] provides an up-to-date overview on different F-measure maximization methods. [36] uses a graph-cut algorithm and has poor scalability on high dimensional text datasets. There are two methods that use a cost-sensitive approach to optimize F-measure score during training, such as Condensed Filter Tree method (**CFT**) [37] and the cost-sensitive label embedding with multidimensional scaling method (**CLEMS**) [38], both of which perform poorly and their training are also slow on large datasets. [33] studies F-measure maximization with conditionally independent label subsets. This method has a strong assumption which makes it hard to apply to real data.

2.1.3 A Wrapper for Optimal F-measure

In our work [39], we focus on tag/label prediction metric **instance-F1** with medium-to-large scale multi-label text datasets, such as medical billing codes, movie genres, review objects, patent classification and news categorization. Our main work is based on the General F-measure Maximizer (GFM) method [35]. The original GFM is proposed to either directly estimate L^2 marginals from data (we name it as **LSF**) or use a probabilistic joint classifier $p(\boldsymbol{y}|\boldsymbol{x})$ to sample the required L^2 probabilities (**GFM-sample**). However, the first option of estimating L^2 marginals directly performs poorly in practice. Our speculation is that even though we can use logistic regression to estimate $p(y_l = 1, |\boldsymbol{y}| = s \mid \boldsymbol{x})$ directly, predicting the number of relevant labels $|\boldsymbol{y}|$ by a classifier is a very hard and unnatural task. The second proposed sampling method by the use of a joint estimation $p(\boldsymbol{y}|\boldsymbol{x})$, in practice, is neither efficient nor effective. It turns out that for large number of labels, sampling usually generates low-confidence (flat) joint that allows probability mass to spread over many label combinations.

[Our Contribution]. We instead develop an efficient solution based on the sampling strategy with a critical change, after training the joint estimator $p(\boldsymbol{y}|\boldsymbol{x})$, we derive the required L^2 marginals using **support inference**. With support inference, one can limit the space of possible label sets $\boldsymbol{y}$ to only the observed ones in the training set, and evaluate their probabilities and then marginalize. These marginals are then fed into GFM to produce the F1-optimal prediction, see Algorithm 1.

Method	Bibtex	IMDb	Ohsumed	RCV1	WISE	WIPO
SPEN	39.0	61.1	61.7	65.3	-	65.9
PDsparse	40.7	62.3	67.3	75.0	74.5	67.5
CFT	23.5	-	-	53.5	-	62.7
CLEMS	42.5	-	52.6	72.4	-	67.1
LSF	43.9	59.8	65.0	73.6	76.7	71.1
BR+GFM-Sample	40.2	61.0	64.3	74.9	73.0	70.0
CBM+GFM-Sample	40.4	64.8	65.4	77.9	73.6	70.3
BR+GFM-Support	48.1	63.8	71.0	76.1	80.1	68.0
CRF+GFM-Support	49.5	**67.1**	70.5	76.1	79.4	**72.5**
CBM+GFM-Support	**50.4**	66.2	**72.6**	**78.7**	**81.5**	71.3

Table 2.1: F-measure comparisons with other methods. Note: '-': indicates failed runs with 56 core and 256GB RAM. The statistics of datasets are shown in [39].

In our previous work [39], we demonstrate that, even with the simplest model (binary relevance (BR)), one can train the model using the existing and feasible algorithms, but only take into account the target measure during the prediction. In addition to BR method, we adopt some other multi-label classifiers, such as Conditional Bernoulli Mixture (CBM) [7] and Conditional Random Fields (CRF) [6], and apply GFM with support inference post-training procedure on top of them. Our experiments show that GFM with support inference work extreme well with all those multi-label classifiers and can outperform recent sophisticated methods (PD-sparse [16], SPEN [11]) and models (GFM-Sample, LSF [35], CFT [37] and CLEMS [38]) designed specifically to be multi-label F-optimal (see Table 2.1).

We highlight the support inference, which not only allows us to infer the required marginals from the joint, but also provides some additional regularization effect on the label structures. With support inference, we have put all the probability mass on only support combinations, and give an infinite strong prior which regularizes the posterior to only consider observed label sets. Besides, using these limited number (usually no more than a few thousands for medium-to-large scale multi-label text datasets) of support label sets and their associated probabilities, we can then easily and efficiently infer the label set $\boldsymbol{y}^*$ for optimal instance-F1. Admittedly, support inference has the limitation in that no new label combination will be considered during marginalization. However, this limitation of support inference by itself is largely mitigated by GFM procedure, which could potentially be an unobserved label set $\boldsymbol{y}^* \notin \boldsymbol{\mathcal{Y}}$.

The proposed algorithm of GFM with support inference can be an extension to any given probabilistic multi-label classifiers, as long as the joint $p(\boldsymbol{y}|\boldsymbol{x})$ can be estimated. The algorithm is a post-training method and independent from model training, which costs no extra time in training. Moreover, although Algorithm 1 has a worse case complexity of $\mathcal{O}(L^3)$ in prediction time, it can be reduced to $\mathcal{O}(LT^2)$, wherein T is the average number of labels per instance. Because in practice, each instance only matches a few labels out of L labels, T is usually bounded up to 10. In another word, the complexity of the Algorithm 1 can be reduced to linear time in $\mathcal{O}(L)$, thus GFM with support inference is very efficient even for dataset with large L.

2.1.4 Compactness and Sparsity in CBM

[Our Contribution]. In our previous work [7], we propose the Conditional Bernoulli Mixtures (CBM) model, which also aims to optimize the likelihood over set prediction. The model represents the joint as a mixture of K components, each with independent label classifiers, with the following form:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{k=1}^{K} \pi(z = k|\boldsymbol{x}; \boldsymbol{\alpha}) \prod_{l=1}^{L} b(y_l|\boldsymbol{x}; \boldsymbol{\beta}_l^k), \tag{2.5}$$

wherein the multi-class classifier, e.g. a Multinomial logistic regression π decides the mixing coefficient for each mixture component. Inside each component, the joint is factorized into marginals, estimated by L binary classifier b, e.g. binary logistic regression. After the joint estimation with π, the joint $p(\boldsymbol{y})$ is not a product of marginals anymore. In other words, labels are not independent and thus the label dependencies can be captured by CBM.

Since the model contains hidden variables, both the multi-class classifier and the binary classifiers can be trained jointly by Expectation Maximization (EM) algorithm, which minimizes the the negative log likelihood:

$$\sum_{i=1}^{N} \mathbb{KL}(\Gamma(z_i)||\pi(z_i|\boldsymbol{x_i}; \boldsymbol{\alpha})) + \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{i=1}^{N} \gamma_i^k \mathbb{KL}(Ber(Y_{il}; y_{il})||b(Y_{il}|\boldsymbol{x}_n; \boldsymbol{\beta}_l^k)) \tag{2.6}$$

wherein capital Y indicates the unknown random labels, while lowercase y is the specific labels from training data; $\Gamma(z_i) = (\gamma_i^1, \gamma_i^2, \ldots, \gamma_i^K)$ is the posterior membership distribution $p(z_i|\boldsymbol{x}_i, \boldsymbol{y}_i)$; $Ber(Y_{il}; y_{il})$ is the Bernoulli distribution with head probability y_{il}.

During the training procedure, we apply EM by following:

E Step: Estimate the posterior membership probability of each instance (i) from each component (k):

$$\gamma_i^k = \frac{\pi(z_i = k|\boldsymbol{x}_i; \boldsymbol{\alpha}) \prod_{l=1}^{L} b(y_{il}|\boldsymbol{x}_i; \boldsymbol{\beta}_l^k)}{\sum_{k=1}^{K} \pi(z_i = k|\boldsymbol{x}_i; \boldsymbol{\alpha}) \prod_{l=1}^{L} b(y_{il}|\boldsymbol{x}_i; \boldsymbol{\beta}_l^k)} \tag{2.7}$$

M Step: Update parameters for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}_l^k$, $\forall l \in \{1, \ldots, L\}\, \forall k \in \{1, \ldots, K\}$, which can be perfectly decomposed into a series of separate optimization problem:

$$\boldsymbol{\alpha}_{new} = \arg\min_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \mathbb{KL}(\Gamma(z_i)||\pi(z_i|\boldsymbol{x}_i; \boldsymbol{\alpha})), \tag{2.8}$$

$$\boldsymbol{\beta}_{l\ new}^k = \arg\min_{\boldsymbol{\beta}_l^k} \sum_{i=1}^{N} \gamma_i^k \mathbb{KL}(Ber(Y_{il}; y_{il})||b(Y_{il}|\boldsymbol{x}_i; \boldsymbol{\beta}_l^k)) \tag{2.9}$$

The optimization problem in (2.8) is essentially a multi-class classification problem with a soft target class distribution $\Gamma(z_i)$, while (2.9) is a weighted (membership estimation from E Step) binary classification problem for each label l in component k.

The original work [7] has demonstrated the effectiveness of CBM against current competitive methods, such as Classifier Chain [9, 10], Conditional Random Fields [6] and so on. However, one major issue of CBM is the computational complexity in $\mathcal{O}(KLDN)$, which is

Dataset	D	L	N	N of Test	inst/label	Cardinality	time (hour)	memory(GB)
Bibtex	1,836	159	4,880	2,515	111.71	2.4	0.002	1
Mediamill	120	101	30,993	12,914	1902.15	4.38	0.009	3
Delicious	500	983	12,920	3,185	311.61	19.03	0.036	10
EURLex-4K	5,000	3,993	15,539	3,809	25.73	5.31	0.178	50
Wiki10-31K	101,938	30,938	14,146	6,616	8.52	18.64	1.253	350
RCV1-2K	47,236	2,456	623,847	155,962	1218.56	4.79	4.388	1,226
AmazonCat-13K	203,882	13,330	1,186,239	306,782	448.57	5.04	45.287	12,651
Delicious-200K	782,585	205,443	196,606	100,095	72.29	75.54	115.68	32,313
AmazonCat-14K	597,540	14,588	4,398,050	1,099,725	1330.1	3.53	183.75	51,330
Amazon-670K	135,909	670,091	490,449	153,025	3.99	5.45	941.235	262,917
WikiLSHTC-325K	1,617,899	325,056	1,778,351	587,084	17.46	3.19	1655.565	462,452
Wikipedia-500K	2,381,304	501,070	1,813,391	783,743	24.75	4.77	2602.318	726,910
Ads-1M	164,592	1,082,898	3,917,928	1,563,137	7.07	1.95	12151.071	3,394,176
Amazon-3M	337,067	2,812,281	1,717,899	742,507	31.64	36.17	13836.52	3,864,973
Ads-9M	2,082,698	8,838,461	70,455,530	22,629,136	14.32	1.79	1783455.494	498,174,810

Table 2.2: Datasets Characteristics. **D**: number of features, **L**: number of labels, **N**: number of training samples, **N of Test**: number of test points, **inst/label**: the average number of training instances per label, **Cardinality**: the average number of labels per instance, **time (h)**: estimated training time for CBM based on its time complexity ($\mathcal{O}(KLDN)$), **memory (GB)**: estimated running memory for CBM training based on its model complexity ($\mathcal{O}(KLD)$). The estimation is based on $K = 20$ in CBM.

K times slower than Binary relevance method ($\mathcal{O}(LDN)$). Besides, the model complexity of CBM ($\mathcal{O}(KLD)$) is at least K times larger than BR method in $\mathcal{O}(LD)$, which further forbids CBM to scale on the large-scale datasets. Because the time complexity of Multinomial logistic regression is $\mathcal{O}(KDN)$, which is much smaller than the complexity of binary classifiers $\mathcal{O}(KLDN)$ for large L thus is eliminated here, and the same to model complexity. Consequently, whether one can solve the training time and memory consumption in binary classifiers will be the bottleneck, determining where computational resources should be focused.

[Proposed Work]. For the large scale datasets, we can approximate the CBM running time and memory according to its complexity, see Table 2.2 [4]. When the number of labels exceeds 10 thousand, the estimated running time can be over than 50 hours and the memory may take up to hundred gigabit even with a small number of clusters ($K = 20$). Therefore, currently CBM is only able to work on datasets with small number of labels and reasonable data scale. However, this proposal will demonstrate CBM has a potential to be extended to larger scale datasets while being able to maintain its performance. We will exploit the sparsity in the CBM structure regarding to solve the time and memory efficiency issues respectively.

Compactness in CBM. In CBM (see Eq (2.5)), each component is forced to train a separated binary classifier for each label l, thus K different binary classifiers for l are generated, which is parameterized by $\boldsymbol{\beta}_l^k \quad \forall k = \{1, \ldots, K\}$. It is quite natural to ask: if CBM really needs to build K different binary classifiers for each label. In practice, we have the following interesting observations on CBM model:

1. Most of the labels only belong to few specific clusters found by CBM. In other words, only few binary classifiers per label are activated. Therefore, we could assume the

[4] The datasets can be found from this Extreme Multi-Label Repository:`http://manikvarma.org/downloads/XC/XMLRepository.html`

rest of inactive binary classifiers provide 0 probability in $b(y_l|\boldsymbol{x};\boldsymbol{\beta}_l^k)$ for $k \in$ *inactive clusters for label* l.

2. During training, each binary classifier of label l in cluster k is only trained on a partial data with membership γ_i^k is larger than 0. However, we observe that most of the data point only belong to part of the clusters. Therefore, with the label imbalance fact, each cluster may not get enough training samples, especially positive instances, thus CBM may fail to estimate the binary classifier in that component.

Inspired by the above two observations, we can train the CBM model in the following form with shared a binary classifier per label:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{k=1}^{K} \pi(z=k|\boldsymbol{x};\boldsymbol{\alpha}) \prod_{l=1}^{L} b(y_l|\boldsymbol{x},\boldsymbol{z};\boldsymbol{\beta}_l), \tag{2.10}$$

To differentiate the binary classifier among different components, we concatenate the feature $\boldsymbol{x}$ with the component indicator as a binary vector $\boldsymbol{z}$. Comparing (2.10) with (2.5), each label classifier is forced to share the same parameters from the features $\boldsymbol{x}$ part, while at the same time take account for the distinct adjustment inside each component from $\boldsymbol{z}$. With the new formula, one can have significant memory saving and avoid the data scarcity problem:

1. The model complexity has been reduced to $\mathcal{O}(KD+L(D+K))$, which can be written as $\mathcal{O}(\mathcal{LD})$ because $K << L \& K << D$ in general, which is very attractive because of the same model complexity as binary relevance method.

2. The label l binary classifier now is trained on the instances that not just appear in one cluster but all clusters. This is a very effective way to utilize the positive instances for a label, alleviating the previous label imbalance issue among each component.

Surprisingly, by shrinking K binary classifiers into one single shared parameterized one, we not just reduce the model complexity, but also have significant speedup that we did not expect. First, this new model does not affect the **E step** and $\boldsymbol{\alpha}_{new}$ optimization in **M Step**, thus we can only focus on training binary classifiers with logistic regression function:

$$b(Y_{il}=1|\boldsymbol{x}_i,\boldsymbol{z}) = \frac{1}{1+\exp\left\{-\left[\sum_{d=1}^{D}\beta_l^d x_i^d + \sum_{k=1}^{K}\beta_l^{k+D} z^k\right]\right\}} \tag{2.11}$$

According to Eq (2.9), we aim to minimize the following objective function for each label l respectively:

$$\ell_l = \sum_{k=1}^{K}\sum_{i=1}^{N} \gamma_i^k \log b(y_{il}|\boldsymbol{x}_i,\boldsymbol{z};\boldsymbol{\beta}_l) \tag{2.12}$$

wherein the gradient can be computed as:

$$\frac{\partial \ell_l}{\partial \beta_l^d} = \sum_{i=1}^{N} y_{il} x_i^d - \sum_{i=1}^{N} x_i^d \sum_{k=1}^{K} \gamma_i^k b(Y_{il}=1|\boldsymbol{x}_n,\boldsymbol{z}) \;\; \forall d \in \{1,\ldots,D\} \tag{2.13}$$

$$\frac{\partial \ell_l}{\partial \beta_l^{k+D}} = \sum_{i=1}^{N} y_{il}\gamma_i^k - \sum_{i=1}^{N} \gamma_i^k b(Y_{il}=1|\boldsymbol{x}_n,\boldsymbol{z}) \quad \forall k \in \{1,\ldots,K\} \tag{2.14}$$

The training complexity has been reduced to $\mathcal{O}(LN(D+K))$, which is surprisingly equivalent to Binary Relevance method $\mathcal{O}(LND)$ since usually $K << D$. Thus, increasing the number of components is not necessary to have a linear increase in training time.

Sparsity in CBM. Even the complexity of CBM can be reduced to the same as binary relevance, for XML datasets with millions of labels / features / instances, a sub-linear time complexity is still highly demanded. We realize sparsity in CBM model, which can be further used as a practical way to massively speed up the model training. More specifically, we can further save the computation based on the sparsity of the output γ_i^k. In practice, we observe that the posterior membership probability of each instance is very sparse, meaning that for most of the components γ_i^k equals or extremely closes to zero. Considering CBM with shared parameters that increasing components without sacrificing training time and memory, we can further produce a sparser $\boldsymbol{\gamma} \in \mathbb{R}^{N \times K}$ matrix with larger K. Noticing the model formula, we do not need to consider such points within that component if the posterior probability is close to zero $\gamma^k \approx 0$. As a result, within a component k, the actual number of training instances N_k can be much smaller than N. One simple and efficient way to implement this sparsity is to only keep the top $\hat{K}$ posterior probabilities for each instance. In this case, each instance is forced to belong up to $\hat{K}$ components. Suppose each component receives similar numbers of training samples, the number of instances in a component can be reduced by a factor $\frac{\hat{K}}{K}$, e.g. $N_k \approx \frac{\hat{K}}{K}N$. Therefore, the CBM's complexity can be theoretically reduced to $\mathcal{O}(LN_kD)$, wherein $N_k \approx \frac{\hat{K}}{K}N$. Again, with a larger number of components K, it not only increases the model capacity, but also reduces the training samples within each component thus is more effective.

Furthermore, we notice that the reduction factor $\frac{\hat{K}}{K}$ is not a best solution when having label imbalance issues in the datasets, see Figure 2.1. In CBM, the primary time consumption is to train each label a binary classifier. Due to the label imbalance, each label usually does not have enough positive instances for training, in other words, the majority of the instances is negative samples. One simple way is to apply undersampling technique, which randomly removes negative samples from each component. The undersampling strategy for negative instances can be easily further improved by considering the instance weight γ_i^k among all other negative instances weights:

$$p_{il} = \frac{\gamma_i^k}{\sum_{n:y_{nl}=0} \gamma_i^k} \quad \forall i : y_{il} = 0. \tag{2.15}$$

We have implemented the sparse version of CBM, based on the proposed sparsity in CBM. The preliminary results (see Table 2.3) show the training time (in seconds) comparisons between BR and sparse CBM on 6 multi-label datasets (Table 2.2). First, sparse CBM can achieve less training time than BR, especially on datasets with larger number of labels. Besides, by exploiting the sparsity inherent in CBM structure, CBM has training time growing sub-linearly with number of labels while keeping competitive performance.

With binary / Multinomial logistic regression as the base leaner in CBM and the success of sparse optimization techniques [16, 17] in extreme classification, we can also extend such techniques to CBM base learners, which can achieve sub-linear complexity in both number of features and instances.

	Bibtex	Delicious	Mediamill	EURLex-4K	RCV1-2K	Wiki10-31K
Training Time (seconds)						
BR	**5**	50	**20**	1800	5600	9300
Sparse-CBM	8	**40**	60	**900**	**1800**	**2200**
Precision@1						
BR	0.644	0.673	0.838	0.718	**0.904**	**0.854**
Sparse-CBM	**0.651**	**0.675**	**0.845**	**0.794**	0.900	0.820

Table 2.3: Training time and Precision performance comparisons between Binary Relevance (BR) and Sparse CBM. The time is reported in second. $K = 20$ is fixed in Sparse-CBM. Precision@3 and Precision@5 have similar conclusion with Precision@1, thus are eliminated here.

2.2 Label Ranking

As the size of labels increases, predicting a label set often becomes impracticable, wherein even F-measure may approximate to zero. In this section, we focus on the ranking tasks, which are more popular in practice when label size is extreme large.

2.2.1 Related Work for Label Ranking

In the extreme multi-label classification (**XML**), most of the models focus on tag/label ranking tasks since the size of label collection is large. Ranking metrics [14, 40], such as **Precision** at top K ($P@K$) and the **Normalized Discounted Cummulated Gains** at top K ($n@K$), are more practicable than predicting label set out of a large range of labels, thus they have been widely used in XML. With the increasing scale of labels, data quality issues, such as label noise and imbalance, are getting more unfavorable. The fundamental data illness issue as well as efficiency problem, have been addressed partially by researchers in the literature. We group them into different categories and describe representative methods in each category.

$$P@K = \frac{1}{K} \sum_{l \in rank_K(\hat{\boldsymbol{y}})} y_l \tag{2.16}$$

$$DCG@K = \sum_{l \in rank_K(\hat{\boldsymbol{y}})} \frac{y_l}{log(l+1)} \tag{2.17}$$

$$nDCG@K = \frac{DCG@K}{\sum_{l=1}^{\min(K,|\boldsymbol{y}|)} \frac{1}{\log(l+1)}} \tag{2.18}$$

(wherein the $rank_K$ returns K largest indices of the prediction $\hat{\boldsymbol{y}}$ in a descending order, and $|\boldsymbol{y}|$ is the number of positive labels in ground truth.)

Binary Relevance Methods: A popular method is to divide the multi-label classification problem into multiple binary classification problems [4, 16, 19, 20, 41]. A typical implementation is to treat labels independently and train one-vs-all classifiers for each of the labels. These independent classifiers can be trained in parallel and thus are computationally efficient in practice. Ignoring the inter-label dependency also enables efficient optimization algorithm, which further reduces computational cost. However, ignoring label dependency

inherently limits prediction accuracy. A competitive method in this category is called PD-Sparse [16], with a variant of the Block-Coordinate Frank-Wolfe training algorithm that exploits data sparsity and achieves complexity sub-linear in the number of primal and dual variables. PD-Sparse [16] shows better performance with less training and prediction time than 1-vs-all Logistic Regression or SVM on extreme multi-label datasets.

Tree Based Classifiers: Following the success of tree-based algorithms in binary classification problems, people also proposed tree-based algorithms for multi-label classification [40, 42, 43], which achieve promising prediction accuracy. Similar to decision trees, these methods make classification decisions in each branch split. Different from decision trees, each split evaluates all features, instead of one, to make a decision. Also, each decision is for a subset of labels rather than one label. Finally, via ensemble and parallel implementation, trees can boost their prediction accuracy with practically affordable computational cost. Among these tree based classifiers, FastXML [40] further optimizes an nDCG-based ranking loss function and achieves significantly higher accuracy than other peer methods.

Embedding: A major difficulty of extreme multi-label classification is the large number of labels. When labels are inter-dependent, one can attempt to find a lower dimensional latent label space from which one can fully reconstruct the original label space. Over the past decade, many methods were proposed to find this latent label space. In early work, methods were proposed to linearly project the original label space into a lower-dimension space and reconstruct predictions from that space [44, 45]. However, there are two assumptions: (1) the label dependency is linear and (2) the label matrix is low-rank, which do not always hold, as reflected by the low prediction accuracy of these methods. To overcome the limitation of the linear assumption, different methods were proposed using non-linear embeddings, including kernels, sub-sampling [24], feature-aware [12, 46] and pairwise distance preservation [14]. Among these methods, SLEEC [14] stands out for less training time and higher accuracies. SLEEC introduces a method for learning a small ensemble of local pairwise distance preserving embeddings which allows it to avoid the low-rank and linear-dependency assumption.

Deep Learning: Deep learning has not been well studied for XML, although it has achieved great successes in binary and multi-class classification problems [47, 48].

FastText [49] reconstructs a document representation by averaging the embedding of the words in the document, followed by a softmax transformation. It is a simple but very effective and accurate multi-class text classifier, as demonstrated in both sentiment analysis and multi-class classification [49]. However, FastText may not be directly applicable for more complicated problems, like XML.

BoW-CNN [50] learns powerful embedding of small text regions by applying CNN to high-dimensional text data. The embedding of all regions are sent to one or multiple convolutional layers, a pooling layer and the output layer at the end.

XML-CNN [51] achieves computational efficiency by training a deep neural network with a hidden bottleneck layer much smaller than the output layer. However, this method has a few drawbacks. First, it is trained using the binary cross entropy loss. This loss tends to be sensitive to label noise, which is frequently observed in extreme multi-label data. Since the label vocabulary is large, it is quite common for human annotator to miss relevant tags. When the classifier's prediction (which might be correct) disagrees with the annotation, the cross entropy loss can potentially assign an unbounded penalty to the classifier during training procedure. The second issue is that because labels are trained independently as

separate binary classification tasks, their prediction probabilities/scores may not be directly comparable. This is problematic because in many applications the requirement is to rank all labels according to their relevance, as opposed to making an independent binary decision on each label.

2.2.2 Ranking-based Autoencoder

In this work [52], we concentrate on tag/label ranking metrics with large scale multi-label datasets, such as image object detection, Wikipedia article tagging and products categorization. In those problems, the number of potential labels is usually from tens of thousands to hundreds of thousands. A method, even with linear complexity in the number of labels (binary relevance), may not scale well to such problems. Many methods have been proposed to reduce the complexity to sub-linear in number of labels, see Section 2.2.1, but few of them pay attention to other challenges, like label dependencies and label noises. In this work, we proposed a new deep learning method, incorporating capturing label dependencies and tolerating label noise for large scale multi-label datasets.

Inspired by C2AE [12], a Ranking-based Auto-Encoder (**Rank-AE**), as depicted in Figure 2.2, has been proposed in our previous work [52]. Rank-AE includes three mapping functions to be trained: a mapping from input features $\boldsymbol{x}$ to feature embeddings $\boldsymbol{x}_h$, denoted as $\mathcal{F}(\boldsymbol{x})$, where h is the embedding size; an encoder from output labels $\boldsymbol{y}$ to label embeddings $\boldsymbol{y}_h$ as $\mathcal{E}(\boldsymbol{y})$; a decoder from label embeddings $\boldsymbol{y}_h$ to output labels $\boldsymbol{y}'$, written as $\mathcal{D}(\boldsymbol{y}_h)$. The proposed model is built on two assumptions: first, each instance can be represented from two different aspects, features $\boldsymbol{x}$ and labels $\boldsymbol{y}$, so there exists a common latent space between $\boldsymbol{x}$ and $\boldsymbol{y}$; second, labels can be reproduced by an autoencoder. Based on these two assumptions, we design the object function as below:

$$\mathcal{L} = \min_{\mathcal{F},\mathcal{E},\mathcal{D}} \mathcal{L}_h(\boldsymbol{x}_h, \boldsymbol{y}_h) + \lambda \mathcal{L}_{ae}(\boldsymbol{y}, \boldsymbol{y}') \tag{2.19}$$

wherein loss $\mathcal{L}_h(\boldsymbol{x_h}, \boldsymbol{y}_h)$ aims to find the common latent space for input $\boldsymbol{x}$ and output $\boldsymbol{y}$ and $\mathcal{L}_{ae}(\boldsymbol{y}, \boldsymbol{y}')$ enforces the output to be reproducible. λ is a hyper-parameter to balance these two losses. During the training, the model learns a joint network including $\mathcal{F}, \mathcal{E}$ and $\mathcal{D}$ to minimize the empirical loss Eq (2.19).

During inference, a given input $\hat{\boldsymbol{x}}$ will be first transformed into a vector in latent space $\hat{\boldsymbol{x}}_h = \mathcal{F}(\hat{\boldsymbol{x}})$, which will then be fed into the label decoder to compute the predictions $\hat{\boldsymbol{y}} = \mathcal{D}(\hat{\boldsymbol{x}}_h)$. It is worth mentioning that although the label encoder $\mathcal{E}$ is ignored during the prediction, it is able to exploit cross-label dependency during the label embedding stage [12]. Recent work also shows that using co-occurring labels information to initialize the neural network can further improve accuracy in multi-label classification [53, 54].

[Main Contribution]. We adopt the mean squared loss in $\mathcal{L}_h(\boldsymbol{x_h}, \boldsymbol{y}_h)$. In $\mathcal{L}_{ae}(\boldsymbol{y}, \boldsymbol{y}')$, in order to alleviate the label noise issue and train the objective in linear time, we propose a

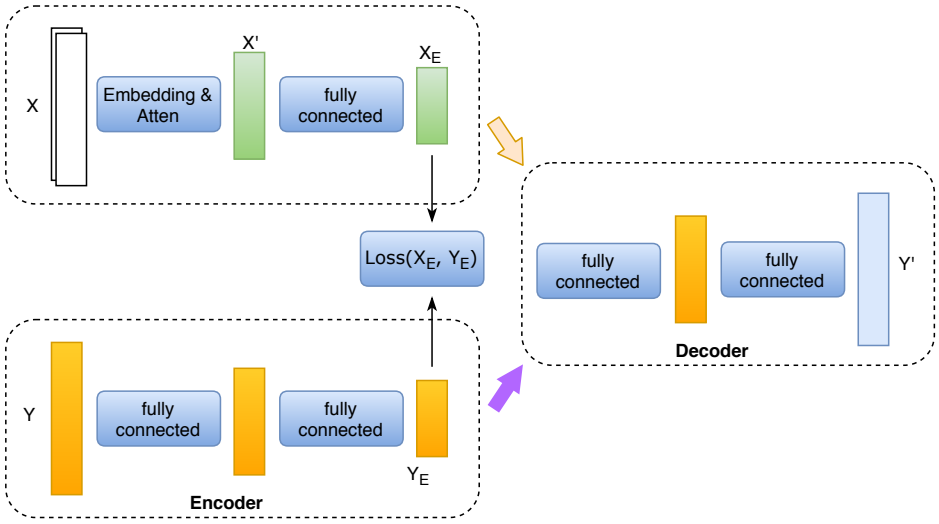


Figure 2.2: Ranking-based AutoEncoder for XML.

marginal-based ranking loss as follows:

$$\mathcal{L}_{ae}(\boldsymbol{y}, \boldsymbol{y}') = \mathcal{L}_P(\boldsymbol{y}, \boldsymbol{y}') + \mathcal{L}_N(\boldsymbol{y}, \boldsymbol{y}') \tag{2.20}$$

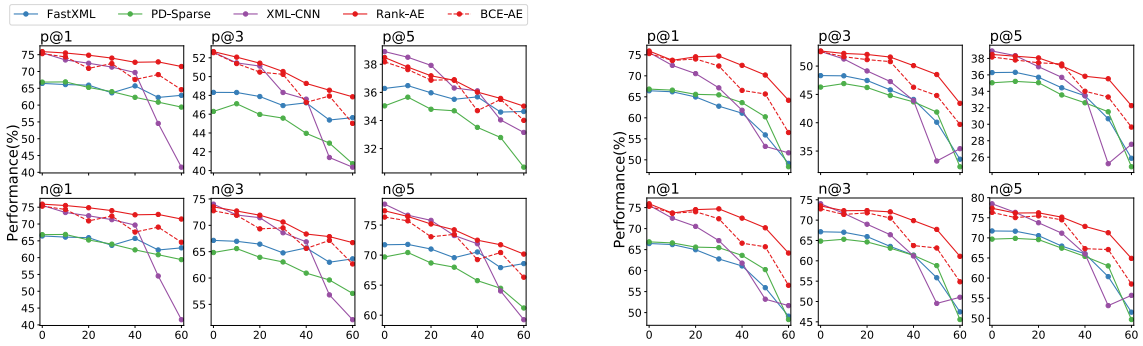
$$\mathcal{L}_P(\boldsymbol{y}, \boldsymbol{y}') = \sum_{n \in N(\boldsymbol{y})} \max_{p \in P(\boldsymbol{y})} (m + y'_n - y'_p)_+ \tag{2.21}$$

$$\mathcal{L}_N(\boldsymbol{y}, \boldsymbol{y}') = \sum_{p \in P(\boldsymbol{y})} \max_{n \in N(\boldsymbol{y})} (m + y'_n - y'_p)_+ \tag{2.22}$$

wherein $N(\boldsymbol{y})$ is the set of negative label indexes, $P(\boldsymbol{y})$ is the complement of $N(\boldsymbol{y})$, and margin $m \in [0, 1]$ is a hyper-parameter for controlling the minimal distance between positive and negative labels. The loss consists of two parts: 1) $\mathcal{L}_P$ targets to raise the minimal score from positive labels over all negative labels at least by m; 2) $\mathcal{L}_N$ aims to penalize the most violated negative label under all positive labels by m. The proposed loss has the following attractive properties: 1) having linear complexity in number of labels $\mathcal{O}(L)$; 2) capturing the relative rankings between positive and negative labels; 3) tolerating the noisy labels with a tunable hyper-parameter m.

We first compare our proposed Rank-AE method with other six outstanding methods in state-of-the-art for large multi-label problems. In the experiment (see Table 2 in [52]), Rank-AE usually achieves the best or the second best on six benchmark XML datasets. We observe that the binary relevance method (PD-sparse [16]), because ignoring label dependencies, often obtains worse performance than other ones. Besides, embedding methods, such as SLEEC [14] and Rank-AE, usually achieve a better results than other BR and tree-based method (Fast-XML [40]).

Furthermore, we conduct an ablation study on label noise. As claiming that the proposed marginal-based ranking loss (Eq 2.20) is more robust to label noise, we control the noise labels in two different ways: 1) missing labels: changing each positive label from $y_l = 1$ to $y_l = 0$ with certain rate, 2) both missing and invalid labels: flipping either from positive to negative or from negative to positive with a noise rate. The noise rates are varied from 0% to 60% on 80% of the training set, and the rest of 20% is noise-free validation set for model



(a) Missing Rate (b) Missing and Invalid Rate

Figure 2.3: Comparisons on noisy labelling IMDb data.

selection. We select five algorithms: FastXML, PD-Sparse, XML-CNN [51], Rank-AE and BCE-AE, wherein BCE-AE is our proposed method but using binary cross-entropy loss in $\mathcal{L}_{ae}(\boldsymbol{y}, \boldsymbol{y}')$. Comparing BCE-AE with Rank-AE can be used to verify whether the robustness to label noise is due to the use of marginal ranking loss.

The performance are reported on the same testing set, shown in Figure 2.3. Rank-AE consistently outperforms other four approaches and has the best robustness tolerating noise labels. Besides, FastXML and PD-Sparse are more tolerant to missing noises than XML-CNN, which may due to XML-CNN has greater capacity and thus more prone to over-fitting the noise. Furthermore, when comparing Rank-AE with BCE-AE, both of which share the same structure but only have different loss functions, the proposed marginal-based ranking loss seems to be robuster than binary cross-entropy loss, which is yet widely used in multi-label classification (e.g. XML-CNN).

However, we observe that the proposed Rank-AE has an efficiency bottleneck regarding the number of labels L, thus it is limited to multi-label datasets with less than hundreds of thousands labels. In order to run datasets with more than hundreds of thousands or millions labels, an algorithm of computational complexity in sub-linear time is on demand.

Chapter 3

Survival Analysis

3.1 Challenges

Survival analysis is a very active research field, wherein the goal is to analyze relationships between single or multiple explanatory variables and outcome variable, where the outcome is the time until the occurrence of an event of interest [55]. The event of interest can be death, or usually might be extended to occurrence of a disease, reliability of a product, divorce in a marriage, click through rate and so on. The time to event or survival time can be measured by hours, days, months, years and etc. Survival analysis has been applied in various real-world domains, such as but not limited to healthcare, epidemiology, economics, engineering [56–58] and so forth. One of the main challenges in this literature is to apply existing survival analysis methods to the large-scale data [59, 60], due to the development of data acquisition techniques to collect a wide variety of data over long-term periods. In our epidemiology study, we aim to estimate the relationships between air pollution (explanatory variables) and mortality (outcome of interest), but there are more than 60 million Medicare beneficiaries across 40 thousand ZIP Code areas in United States from 2000 to 2012, which is up to 5.7 billion person-months of follow-up. Such prohibitively large data poses significant challenge for most of the conventional methods. Moreover, observing the time of occurrence of an event also presents a very common issue: a large amount of censored observations, wherein the observations are called censored when the time of the event is incomplete during the study period. The censored observations are usually caused by the time limitation of the study period, or losing track during the observation period. For example, in the mortality dataset of our study, because the mortality is only tracked from 2000 to 2012, 20 million deaths are observed (uncensored instances), while the information of deaths for the remaining population is unknown (called censored instances). In other words, censored instances can represent a particular type of missing or latent data. Because of this censored issue, many conventional regression models are not directly suitable for analyzing survival data, thus survival methods that integrate both censored and uncensored data are highly on demand.

Due to the difficulty in handling censored and large scale data for many algorithms, we focus on extending current effective survival methods to large scale data, e.g. Cox model [61]. It is worth noting that there are many methods, that have been applied to Survival Analysis, from statistical models, such as non-parametric, semi-parametric and parametric

models, to machine learning based methods (Bayesian methods, Survival Trees, Neural Network and etc.), summarized in a recent survey [62]. However, over the past half century, Cox model, as a semi-parametric statistical model, still has reigned over the research for analyzing time-to-event data [59]. Comparing with non-parametric models, Cox is much easier to interpret by the estimated regression coefficients, and yields more accurate estimation [62]. While analogized to full parametric methods, although being little harder to interpret, Cox model does not give a strong assumption of the distribution on survival time, thus is not necessary to violate the assumption and suffer sub-optimal estimation.

In spite of those virtue, the traditional Cox model is failed to handle correlated variables which are commonly seen in many practical problems. In our air-pollution and mortality example, air pollution causes, such as $PM_{2.5}$ (refers to atmospheric particulate matter that have a diameter of less than 2.5 micrometers), Nitrogen Dioxide (NO_2) and Ozone (O_3), are highly correlated to each other, i.e. the presence of collinearity. Assessing the true effects of each of the correlated variables in a flexible multivariable survival analysis is quite challenging, often leading to unreliable and unstable estimated coefficients.

To address the above challenges, we summarize the main goal in this proposal as follows:

1. How to extend existing survival algorithms to scale on extreme large scale dataset.

2. How to handle the collinearity issue on large scale dataset.

3.2 Related Work

Since focusing on two main issues–scalability and collinearity, in this proposal, we only review some relevant work in these two aspects.

3.2.1 Scalability

In the early stage of survival analysis, researchers often work on datasets with only a few hundred or thousand of predictors and observations. However, the well developed data acquisition techniques have motivated people to analyze data with hundreds of thousands of predictors and millions even billions of observations across different fields. Thus, large scale survival datasets can be recognized in two facets: high-dimensional predictors and massive sample-size [59]. High-dimensional predictors are oftentimes seen in genomics study with multiple gene expression as variables, which may exceed 10^6 [59, 60]. While in our epidemiology analysis, more than 60 million Medicare beneficiaries are included, and they are tracked over 13 years, which is up to 5.7 billion person-month follow-up observations.

To find the scalable solutions, people have been working on either developing new and efficient methodologies for large data analysis or extending the existing survival models. For example, due to the success of deep learning, [63] extends the deep neural network with Bayesian optimization to handle high-dimensional data; [64] proposes a novel model in a Bayesian framework, called deep survival analysis, for large survival analysis; [59, 60] explore the sparsity of high-dimensional data using a variant of the cyclic coordinate descent optimization for Cox model; some other regularized Cox models [65] address the high-dimensional but low sample-size data to avoid over-fitting issue.

To the best of our knowledge, most extensions of Cox model focus on either selecting the best of predictors from a large range of variables or applying novel optimization techniques in order to be scaled well on large dataset. However, for datasets with low-dimensional variables but heavily massive sample-size (billions of observations), are rarely explored in this domain. The reasons for the lack of exploration of such dataset are mainly twofold: I) tracking more than billions of observations is an extremely hard and time consuming task; II) some data information are very sensitive, thus not available for most of researchers.

3.2.2 Collinearity

In statistics, collinearity is a phenomenon where one predictor, in a multivariable regression model, can be linearly predicted from the others. In this situation, the coefficient estimates of the multivariable regression may change erratically in response to small changes in the model or data [66]. It is noteworthy that When evaluating the model as a whole, e.g. predictive power, collinearity does not usually have any influence; collinearity often affects the estimation for individual predictor. In our study, we target on the associations between each air pollution cause and mortality. Therefore, when multiple air pollution causes are included in one regression model, assessing the true effects of each variables becomes unreliable and unstable [67].

Ridge regularization, also called L2 norm, is widely used in multivariable regression model, including Cox regression model, when covariates are highly correlated [68]. Ridge estimator spreads the weights across all correlated features, and this normally makes the model more robust and generalized to new data. However, ridge does not provide the true effects of each variable, instead which is exactly what we want. In addition to ridge method, Cluster analysis and Principal Component Analysis (PCA) are also often applied to address collinearity issue [69] by combining / removing the correlated variables. Nevertheless, either combing or removing variables leads to lack of interpretability of the results. A most recent work [70] proposes the **deconfounder** algorithm for scientific studies involve multiple causes, when different variables whose effects are simultaneously of interest. The deconfounder can infer a latent variable as a substitute for unobserved correlations and then uses that substitute to perform downstream tasks.

3.3 Contribution and Future Work

In this section, we will claim our main contribution how to scale existing methods on extreme large scale survival datasets with massive sample-size but low-dimensional features. In particular, we aim to assess the associations between mortality and air pollution, such as $PM_{2.5}$, NO_2, O_3 and etc, with about 5.7 billion observations.

3.3.1 Introduction to Cox Proportional Hazard Model

In survival analysis, **survival time** T is the main concept, which tracks the occurrence of an event of interest. Given the epidemiological study as an example, we often follow people and wait until some event happens to them, e.g. deaths. One can define a **survival function**

of T:

$$S(t) = p(T \geq t) \tag{3.1}$$

which represents the cumulative probability that the time to the event of interest does not occur earlier than a specific time t [62]. Alternatively, one can also write the **failure function**:

$$F(t) = 1 - S(t) = p(T < t) \tag{3.2}$$

which is the cumulative probability that the event of interest is earlier than t. There are many proposed methods that directly model survival function, which is equivalent to model the corresponding failure function or the failure density function $f(t) = \frac{d}{dt}F(t)$.

In Cox model, instead we estimate another commonly used function, called **hazard function**, which is an instantaneous rate of failure at time t conditional on that no interest event happened before time t:

$$h(t) = \lim_{\Delta t -> 0} \frac{p(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \tag{3.3}$$

Due to the existence of censored observations, who have not experienced a failure or quit during the study period, we can write the censoring time as C, and then we rewrite the survival times by a new variable $y = \min(T, C)$. Along with an input of the subject's features $\boldsymbol{x} \in \mathbb{R}^D$ with D as feature dimension, the survival dataset $\mathbb{D}$ usually consists of the follows: $\mathbb{D} = \{(y_i, \delta_i, \boldsymbol{x}_i) : i = 1, ..., N\}$, wherein $\delta_i = I(T_i \leq C_i)$, $y = \min(T_i, C_i)$, and N is the total number of observations.

Based on the proportionality assumption, Cox Proportional Hazard model (**CPH**) [61] is based on each observation i, defined as:

$$h(y_i | \boldsymbol{x}_i; \boldsymbol{\beta}) = h_0(y_i) \exp(\boldsymbol{\beta}^T \boldsymbol{x}_i) \tag{3.4}$$

wherein h_0 is the **baseline hazard function**, an unspecified function of the survival time y. CPH is a semi-parametric algorithm because it contains both non-parametric and parametric functions.

Because there is no explicit formula for the baseline hazard function, estimating Eq 3.4 directly by standard maximum likelihood is very difficult. Instead, Cox [61] proposed to maximize the following partial likelihood:

$$L(\boldsymbol{\beta} | \mathbb{D}) = \prod_{i=1}^{N} \left(\frac{\exp(\boldsymbol{\beta}^T \boldsymbol{x}_i)}{\sum_{t \in R(y_i)} \exp(\boldsymbol{\beta}^T \boldsymbol{x}_t)} \right)^{\delta_i} \tag{3.5}$$

$R(y_i) = \{t : y_t \geq y_i\}$ is the risk set of observations who are still survival and at risk at time y_i.

Maximizing the partial likelihood is obviously equivalent to maximize the following log-likelihood:

$$\ell(\boldsymbol{\beta} | \mathbb{D}) = \sum_{i=1}^{N} \delta_i \left(\boldsymbol{\beta}^T \boldsymbol{x}_i - \log \sum_{t \in R(y_i)} \exp(\boldsymbol{\beta}^T \boldsymbol{x}_t) \right) \tag{3.6}$$

To simplify the model explanation, we omit the L2 penalty term of $\boldsymbol{\beta}$ here. It is well known that Eq 3.6, even with L2 norm, is a smooth convex function, and a wide range of optimization algorithms can be utilized. However, for optimization algorithms such as batch gradient descent, Newton methods and etc., the biggest computational challenge is due to the mathematical operations on millions or billions of subjects with time complexity $\mathcal{O}(N(D + |\bar{R}|D)) = \mathcal{O}(N|\bar{R}|D)$, where $\bar{R}$ is the average size of risk sets. We will demonstrate how to deal with the massive sample-size issue by exploiting the compactness in our dataset and the CPH algorithms.

3.3.2 Compactness in Data

In our epidemiological survival analysis, we examine associations between air pollution ($PM_{2.5}, NO_2, O_3$) and mortality. To perform such analysis, we consider over 60 million Medicare enrollees across 40 thousand of ZIP Code areas in the conterminous United States from 2000 to 2012. Each Medicare individual has been measured in these aspects: gender, age, race, timeline (monthly), indicator for being alive or dead at that time, living ZIP Code, and the corresponding monthly air pollution values in that ZIP Code and so on. And each individual has been tracked through 13 years with monthly report. As a result, we have massive sample-size (N), approximately 5.7 billion person-months followups, but low-dimensional features (D), less than 20. Besides, each feature, such as gender, age and race, only contains few possible values, except for air pollution feature, which is continuous. However, for individuals living in the same ZIP Code area, they also share the same air pollution feature. In this case, the product operation on $\boldsymbol{\beta}^T \boldsymbol{x}_i$ will be repeated many times, i.e. the number of individuals with exactly the same properties.

In practice, one can avoid the duplicated computation by grouping individuals, who have the same properties, into one group with two additional counts: the total number of deaths in this group and the size of the group. Normally, air pollution is varied month to month for each ZIP Code area, and each area has different level pollutant. Therefore, we could group the epidemiology survival data according to different locations and months. We write a new location-time related feature as X_t^l, indicating the variables for people who have same properties living in location l at time t. And the number of deaths and the size of the group are denoted as δ_t^l and Y_t^l respectively. By exploiting the compactness in the given data, we shrink lots of duplicated row data into one grouped row. In our example, the original person-month data has roughly 5.7 billion rows, but after the grouping, the number has been reduced to 0.5 billion. The data storage has also been compressed to 30 Gigabit.

3.3.3 Cox PH model with Compact Data

The original partial log-likelihood (Eq 3.6) is based on individual observation, and its gradient can be written as:

$$\frac{\partial}{\partial \boldsymbol{\beta}} \ell(\boldsymbol{\beta}|\mathbb{D}) = \sum_{i=1}^{N} \delta_i \left(\boldsymbol{x}_i - \sum_{m \in R(y_i)} \boldsymbol{x}_m \frac{\exp(\boldsymbol{\beta}^T \boldsymbol{x}_m)}{\sum_{t \in R(y_i)} \exp(\boldsymbol{\beta}^T \boldsymbol{x}_t)} \right) \tag{3.7}$$

The gradient is usually used to update the coefficient vector $\boldsymbol{\beta}$. Without compressing the data structure, the updating can be realized in time $\mathcal{O}(N|\bar{R}|D)$, which is an extremely time-

consuming process with large scale sample size. However, after introducing the shared location-time related features x_t^l and accumulated counts for deaths and local population (δ_t^l and Y_t^l), we show that the updating rule can be transformed into the following format:

$$\frac{\partial}{\partial \boldsymbol{\beta}} \ell(\boldsymbol{\beta}|\mathbb{D}) = \sum_{l \in L} \left[\sum_{t=1}^{T} \delta_t^l X_t^l \left(Y_t^l - \exp(\boldsymbol{\beta}^T X_t^l) \sum_{m \in R(t)} \frac{Y_m^l}{\sum_{j \in R(m)} \exp(\boldsymbol{\beta} X_j^l)} \right) \right] \tag{3.8}$$

wherein the L is the set of locations, i.e. 40 thousand ZIP Code areas, and T is the monthly timeline from 2000 to 2012 (The size of T is $13 \times 12 = 156$). One can find the proof of the transformation in Appendix A. The time complexity for Eq 3.8 is $\mathcal{O}(LT|\bar{R}|D)$. Comparing it with Eq 3.7 $\mathcal{O}(N|\bar{R}|D)$, one can see that the time has been reduced by a factor $\frac{N}{LT} \approx 900$, when $N \approx 5.7$ million, $L \approx 40$ thousand, and $T = 156$. In other words, the shared updating rule allows CPH model to be up to 900 times faster during training.

We choose two different optimization algorithms, Batch Gradient Descent and Limited-memory BFGS(L-BFGS) [71], to maximize the objective function according to Eq 3.8. The second derivative of Eq 3.6 has the same virtue when the data is compressed.

We first verify the correctness of our CPH implementation by comparing with some existing packages in *R* and *SAS*. We sample a small subset (1 million person-month followups) and use the same settings in different packages. The experiment result shows that our implementation can provide almost identical estimations to those obtained from other packages, see Appendix B.

For testing CPH on the whole dataset, we do not have any fair comparisons because almost all of the available packages are not feasible for such large scale data. And there is no other researchers reporting the estimation based on 5.7 billion person-months of follow-up before. To our best knowledge, Di et al. [72] is the closest analysis, wherein only 0.5 billion person-years of follow-up are considered. In their study, they claimed that running Cox PH model with larger scale data would be computationally infeasible. However, we demonstrate that our improved Cox PH model is able to analyze this data within 10 minutes on one machine (Intel Xeon CPU E5-2680 v4 2.4GHz and 56 logical cores).

In the previous studies, researchers often consider smaller set of population for the estimating the associations between air pollution and mortality. Therefore, their studies usually provide key evidence of air pollution's causal impacts on mortality from specific causes of deaths, or people living in urban areas only. While our proposed CPH model allows us to examine pollution impacts on the mortality and health experience of extremely large cohorts, thus allowing examination of understudied groups. Based on this new implementation, a few of our work have been published on either journals or conferences [56, 73–75].

3.3.4 Collinearity

Although we estimate similarly consistent and strong associations between single air pollution exposure and increased mortality, our study has a critical issue when including multiple air pollution exposures into CPH model. This limitation is usually caused by the strong correlations between air pollution exposures, e.g. the Pearson correlation [1] between $PM_{2.5}$ and NO_2 is 0.60. In Table 3.1, when modeling pollution separately, the estimation for each

[1] `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`

	Single Pollutant Model		Two Pollutant Model	
pollutant	$PM_{2.5}$	NO_2	$PM_{2.5}$	NO_2
$\boldsymbol{\beta}^*$	0.0211	0.0245	0.0001	0.0026

Table 3.1: Comparisons between one-pollutant Cox Proportional Hazard and two-pollutant CPH Model. Single Pollutant Model is CPH model with one pollution exposure only, while Two Pollutant Model is the one with two pollution exposures the same time.

pollutant is strong positive, but the association for $PM_{2.5}$ is almost attenuated to zero when NO_2 is included together. As mentioned before, collinearity often affects the estimation for individual predictor and the assessment becomes unreliable and unstable [67] because of the overlapping information the predictors share.

Strategy I-Ridge Norm. The adverse impact of collinearity in regression has been observed, but there are not much attention since most of the time survival analyzes focus on a single exposure of interest. Besides, dropping some of the correlated variables (i.e. including only one correlated variable) is also often recommended [76, 77]. This is why we start with assessing the effect on one single pollution exposure. However, for the sake of scientific advancement and model completeness, sometimes we still need to keep correlated variables to build a more accurate model. One of the most common ways to deal with correlated variables is to apply ridge (L2) regression [68, 76]. We implemented the CPH model with L2 norm, unfortunately, we observed that the estimations for collinear variables could be easily manipulated by tuning the regularization strength, resulting in biased estimation [76].

Strategy II-Residual Model. Another often used strategy to handle collinearity is called residual model [77–79]. For example, $PM_{2.5}$, which correlated with NO_2, is not directly included in the regression model. Instead, we fit a linear regressor for $PM_{2.5}$ using NO_2 as predictor, and add the residuals from the fitted regressor into the Cox regression model. In this case, the residual of $PM_{2.5}$ is orthogonal to No_2, thus reduce the issue from collinearity. However, this residual model has been criticized for an overestimation or difficulty in interpretation [77].

Strategy III-Deconfounder. Yixin et al. [70] recently propose a deconfounder algorithm to find and fit a latent-variable model to capture the dependence among variables, e.g. the dependence in air pollution exposures. This estimated variable is a substitute for unobserved confounders, and used in the causal inference. The deconfounder aims to discover the hidden confounders, variables that affect both other variables and the outcome. For example, any location with high $PM_{2.5}$ may also have high NO_2 pollutant, such as urban areas. And $PM_{2.5}$ has an effect on both other pollutant and the increasing mortality. With all assumptions hold (see [70], the deconfounder can provide an unbiased estimation. Therefore, we plan to implement the deconfounder algorithm to solve the collinear variables issue in our survival analysis. However, before the implementation, we have to ask ourselves the following two questions: 1) can we extend the deconfounder method to survival analysis, instead of the original application causal inference; 2) is it able to scale deconfounder algorithm well on our big dataset.

Chapter 4

Timeline

Timeline	Task	Progress
Fall 2019	Speeding up CBM algorithm and handling label noise and label imbalance issues	ongoing
Spring 2020	Optimizing task specific metrics for Extreme Multi-label classifiers	planning
Spring 2020	Handling collinearity issue in Survival analysis	ongoing
Summer 2020	Thesis writing and defense.	planning

Bibliography

[1] E. Brynjolfsson, L. M. Hitt, and H. Kim. *Strength in Numbers: How does data-driven decision-making affect firm performance?* In *ICIS*, (2011).

[2] M. I. Jordan and T. M. Mitchell, *Machine learning: Trends, perspectives, and prospects*, Science **349**, 255–260 (2015).

[3] Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, *Big data opportunities and challenges: Discussions from data analytics perspectives*, IEEE Computational Intelligence Magazine **9**, 62–74 (2014).

[4] G. Tsoumakas and I. Katakis, *Multi-label classification: An overview*, International Journal of Data Warehousing and Mining (IJDWM) **3**, 1–13 (2007).

[5] M.-L. Zhang and Z.-H. Zhou, *A review on multi-label learning algorithms*, IEEE transactions on knowledge and data engineering **26**, 1819–1837 (2014).

[6] N. Ghamrawi and A. McCallum. *Collective multi-label classification*. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM, (2005).

[7] C. Li, B. Wang, V. Pavlu, and J. Aslam. *Conditional bernoulli mixtures for multi-label classification*. In *International Conference on Machine Learning*, pages 2482–2491, (2016).

[8] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, (2001). Morgan Kaufmann Publishers Inc.

[9] W. Cheng, E. Hüllermeier, and K. J. Dembczynski. *Bayes optimal multilabel classification via probabilistic classifier chains*. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 279–286, (2010).

[10] J. Read, B. Pfahringer, G. Holmes, and E. Frank, *Classifier chains for multi-label classification*, Machine learning **85**, 333 (2011).

[11] D. Belanger and A. McCallum. *Structured prediction energy networks*. In *International Conference on Machine Learning*, pages 983–992, (2016).

[12] C.-K. Yeh, W.-C. Wu, W.-J. Ko, and Y.-C. F. Wang. *Learning deep latent space for multi-label classification*. In *Thirty-First AAAI Conference on Artificial Intelligence*, (2017).

[13] A. Zubiaga, *Enhancing Navigation on Wikipedia with Social Tags*, CoRR **abs/1202.5469** (2012).

[14] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. *Sparse local embeddings for extreme multi-label classification*. In *Advances in neural information processing systems*, pages 730–738, (2015).

[15] R. Wetzker, C. Zimmermann, and C. Bauckhage. *Analyzing social bookmarking systems: A del. icio. us cookbook*. In *Proceedings of the ECAI 2008 Mining Social Data Workshop*, pages 26–30, (2008).

[16] I. E.-H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. Dhillon. *Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification*. In *International Conference on Machine Learning*, pages 3069–3077, (2016).

[17] Q. Lei, I. E. Yen, C.-y. Wu, I. S. Dhillon, and P. Ravikumar. *Doubly greedy primal-dual coordinate descent for sparse empirical risk minimization*. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2034–2042. JMLR. org, (2017).

[18] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, *On label dependence and loss minimization in multi-label classification*, Machine Learning **88**, 5–45 (2012).

[19] I. E. Yen, X. Huang, W. Dai, P. Ravikumar, I. Dhillon, and E. Xing. *Ppdsparse: A parallel primal-dual sparse method for extreme classification*. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM, (2017).

[20] R. Babbar and B. Schölkopf. *Dismec: Distributed sparse machines for extreme multi-label classification*. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 721–729. ACM, (2017).

[21] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. *Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising*. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 993–1002. International World Wide Web Conferences Steering Committee, (2018).

[22] H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. *Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches*. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 528–536. ACM, (2019).

[23] S. Nowak and S. Rüger. *How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation*. In *Proceedings of the international conference on Multimedia information retrieval*, pages 557–566. ACM, (2010).

[24] H.-F. Yu, P. Jain, P. Kar, and I. Dhillon. *Large-scale multi-label learning with missing labels*. In *International conference on machine learning*, pages 593–601, (2014).

[25] I. Misra, C. Lawrence Zitnick, M. Mitchell, and R. Girshick. *Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2939, (2016).

[26] M.-L. Zhang, Y.-K. Li, and X.-Y. Liu. *Towards class-imbalance aware multi-label learning*. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, (2015).

[27] F. Charte, A. Rivera, M. J. del Jesus, and F. Herrera. *A first approach to deal with imbalance in multi-label datasets*. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 150–160. Springer, (2013).

[28] Yelp. *Automatically Categorizing Yelp Businesses*, (2015). https://engineeringblog.yelp.com/amp/2015/09/automatically-categorizing-yelp-businesses.html.

[29] WISE. *Greek Media Monitoring Multilabel Classification (WISE 2014)*, (2014). www.kaggle.com/c/wise-2014.

[30] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hullermeier. *Extreme f-measure maximization using sparse probability estimates*. In *International Conference on Machine Learning*, pages 1435–1444, (2016).

[31] X.-Z. Wu and Z.-H. Zhou. *A unified view of multi-label performance measures*. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3780–3788. JMLR. org, (2017).

[32] S. P. Parambath, N. Usunier, and Y. Grandvalet. *Optimizing F-measures by cost-sensitive classification*. In *Advances in Neural Information Processing Systems*, pages 2123–2131, (2014).

[33] M. Gasse and A. Aussem. *F-measure maximization in multi-label classification with conditionally independent label subsets*. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 619–631. Springer, (2016).

[34] I. Pillai, G. Fumera, and F. Roli, *Designing multi-label classifiers that maximize F measures: State of the art*, Pattern Recognition **61**, 394–404 (2017).

[35] W. Waegeman, K. Dembczyński, A. Jachnik, W. Cheng, and E. Hüllermeier, *On the bayes-optimality of f-measure maximizers*, The Journal of Machine Learning Research **15**, 3333–3388 (2014).

[36] J. Petterson and T. S. Caetano. *Submodular multi-label learning*. In *Advances in Neural Information Processing Systems*, pages 1512–1520, (2011).

[37] C.-L. Li and H.-T. Lin. *Condensed filter tree for cost-sensitive multi-label classification*. In *International Conference on Machine Learning*, pages 423–431, (2014).

[38] K.-H. Huang and H.-T. Lin, *Cost-sensitive label embedding for multi-label classification*, Machine Learning **106**, 1725–1746 (2017).

[39] B. Wang, C. Li, V. Pavlu, and J. Aslam. *A Pipeline for Optimizing F1-Measure in Multi-label Text Classification*. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 913–918. IEEE, (2018).

[40] Y. Prabhu and M. Varma. *Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning*. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272. ACM, (2014).

[41] B. Hariharan, S. Vishwanathan, and M. Varma, *Efficient max-margin multi-label classification with applications to zero-shot learning*, Machine learning **88**, 127–155 (2012).

[42] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. *Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages*. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24. ACM, (2013).

[43] J. Weston, A. Makadia, and H. Yee. *Label partitioning for sublinear ranking*. In *International Conference on Machine Learning*, pages 181–189, (2013).

[44] F. Tai and H.-T. Lin, *Multilabel classification with principal label space transformation*, Neural Computation **24**, 2508–2542 (2012).

[45] K. Balasubramanian and G. Lebanon. *The Landmark Selection Method for Multiple Output Prediction*. In *ICML*, (2012).

[46] Z. Lin, G. Ding, M. Hu, and J. Wang. *Multi-label classification via feature-aware implicit label space encoding*. In *International conference on machine learning*, pages 325–333, (2014).

[47] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, *A Structured Self-attentive Sentence Embedding*, CoRR **abs/1703.03130** (2017).

[48] Y. Kim, *Convolutional Neural Networks for Sentence Classification*, CoRR **abs/1408.5882** (2014).

[49] E. Grave, T. Mikolov, A. Joulin, and P. Bojanowski. *Bag of tricks for efficient text classification*. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 3–7, (2017).

[50] R. Johnson and T. Zhang. *Effective Use of Word Order for Text Categorization with Convolutional Neural Networks*. In *HLT-NAACL*, (2015).

[51] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang. *Deep learning for extreme multi-label text classification*. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM, (2017).

[52] B. Wang, L. Chen, W. Sun, K. Qin, K. Li, and H. Zhou. *Ranking-Based Autoencoder for Extreme Multi-label Classification*. In *2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, (2019).

[53] G. Kurata, B. Xiang, and B. Zhou. *Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence*. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526, (2016).

[54] S. Baker and A. Korhonen. *Initializing neural networks for hierarchical multi-label text classification*. In *BioNLP 2017*, pages 307–315, (2017).

[55] J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data*, John Wiley & Sons (2011).

[56] K.-D. Eum, F. Kazemiparkouhi, B. Wang, J. Manjourides, V. Pun, V. Pavlu, and H. Suh, *Long-term NO2 exposures and cause-specific mortality in American older adults*, Environment international **124**, 10–15 (2019).

[57] J. Heckman and B. S. Singer. *Longitudinal analysis of labor market data*. Technical report, Cambridge University Press, (2008).

[58] D. W. Hosmer Jr, S. Lemeshow, and S. May, *Applied survival analysis: regression modeling of time-to-event data*, Wiley-Interscience (2008).

[59] S. Mittal, D. Madigan, R. S. Burd, and M. A. Suchard, *High-dimensional, massive sample-size Cox proportional hazards regression for survival analysis*, Biostatistics **15**, 207–221 (2013).

[60] S. Mittal, D. Madigan, J. Q. Cheng, and R. S. Burd, *Large-scale parametric survival analysis*, Statistics in medicine **32**, 3955–3971 (2013).

[61] C. R. David, *Regression models and life tables (with discussion)*, Journal of the Royal Statistical Society **34**, 187–220 (1972).

[62] P. Wang, Y. Li, and C. K. Reddy, *Machine Learning for Survival Analysis: A Survey*, CoRR **abs/1708.04649** (2017).

[63] S. Yousefi, F. Amrollahi, M. Amgad, C. Dong, J. E. Lewis, C. Song, D. A. Gutman, S. H. Halani, J. E. V. Vega, D. J. Brat, and L. A. D. Cooper. *Predicting clinical outcomes from large scale cancer genomic profiles with deep survival models*. In *Scientific Reports*, (2017).

[64] R. Ranganath, A. J. Perotte, N. Elhadad, and D. M. Blei. *Deep Survival Analysis*. In *MLHC*, (2016).

[65] D. Engler and Y. Li, *Survival analysis with high-dimensional covariates: an application in microarray studies*, Statistical applications in genetics and molecular biology **8**, 1–22 (2009).

[66] Wikipedia contributors. *Multicollinearity — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Multicollinearity&oldid=888064443, (2019). [Online; accessed 30-April-2019].

[67] J. Marsh, J. Hutton, and K. Binks, *Removal of radiation dose response effects: an example of over-matching*, Bmj **325**, 327–330 (2002).

[68] X. Xue, M. Y. Kim, and R. E. Shore, *Cox regression analysis in presence of collinearity: an application to assessment of health risks associated with occupational radiation exposure*, Lifetime data analysis **13**, 333–350 (2007).

[69] A. S. Gwelo et al., *Principal Components To Overcome Multicollinearity Problem*, Oradea Journal of Business and Economics **4**, 79–91 (2019).

[70] Y. Wang and D. M. Blei, *The Blessings of Multiple Causes*, arXiv e-prints **1805.06826** (2019).

[71] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical programming **45**, 503–528 (1989).

[72] Q. Di, Y. Wang, A. Zanobetti, Y. Wang, P. Koutrakis, C. Choirat, F. Dominici, and J. D. Schwartz, *Air pollution and mortality in the Medicare population*, New England Journal of Medicine **376**, 2513–2522 (2017).

[73] F. Kazemiparkouhi, K.-D. Eum, B. Wang, J. Manjourides, and H. H. Suh, *Long-term ozone exposures and cause-specific mortality in a US Medicare cohort*, Journal of exposure science & environmental epidemiology , 1 (2019).

[74] F. Kazemiparkouhi, K.-D. Eum, B. Wang, J. Manjourides, and H. Suh. *Effect of Confounding, Effect Modification, and Exposure Measures on the Association of Long-Term Ozone Exposure and Cause-Specific Mortality*. In *ISEE Conference Abstracts*, volume 2018, (2018).

[75] B. Wang, K.-D. Eum, J. Manjourides, F. Kazemiparkouhi, H. Suh, and V. Pavlu. *Effect Modification of the Association of Long-Term PM2. 5 Exposure and Cause-Specific Mortality: An Analysis of 64 Million US Medicare Beneficiaries*. In *ISEE Conference Abstracts*, volume 2018, (2018).

[76] D. Schreiber-Gregory. *Multicollinearity: What Is It, Why Should We Care, and How Can It Be Controlled*, (2017).

[77] F. Tomaschek, P. Hendrix, and R. H. Baayen, *Strategies for addressing collinearity in multivariate linguistic data*, Journal of Phonetics **71**, 249–267 (2018).

[78] U. C. Priva, *Informativity affects consonant duration and deletion rates*, Laboratory Phonology **6**, 243–278 (2015).

[79] A. Tremblay and B. V. Tucker, *The effects of N-gram probabilistic measures on the recognition and production of four-word sequences*, The Mental Lexicon **6**, 302–324 (2011).

Appendix A

Cox Proportional Hazard Model

Air Pollution Studies

Bingyu Wang

April 12, 2017

1 Case Crossover

1.1 General Framework

Let X_{it}^l be the exposure for person i belonging to location l and in interval t, $t = 1, \ldots, T$ and let Y_{it}^l indicates whether person i has the event at location l in interval t(1 - yes, 0 - no). Assume that the outcome Y_{it}^l is rare and that the probability that subject i fails in interval t at location l is given by the relative risk model:

$$\lambda_i(t, X_{it}^l) = \lambda_{it} \exp(\beta X_{it}^l) = \lambda_i \exp(\beta X_{it}^l + \gamma_{it}) \tag{1}$$

Each person is assumed to have his own baseline risk λ_{it} at time t consisting of two parts:

1. λ_i is a constant frailty for person i;
2. $\exp(\gamma_{it})$ is the effect of unmeasured time-varying factors on his risk.

1.2 Case-crossover design

In the case-crossover approach, the exposure of cases in interval t_i is compared to the exposures from a set of references periods, where t_i is event interval and $W(t_i)$ is a set of references periods. For example, $t_i = 8$ indicates the event was on the 8th day and $W(8) = \{7, 8, 9\}$ means the day before and the day after, including itself as the reference periods. The only assumption of a case-crossover design is that the time-varying effect γ_{it} is constant for all $t \in W(t_i)$.

Conditional on an individual being a case within a pre-specified reference windo $W(t_i)$, the probability $p_{it_i}^l$ that subject i belonging to l location and fails at time t_i is

$$\begin{aligned}
p_{it_i}^l &= P(T_i = t_i | X, W(t_i), \sum_{m=1}^{T} Y_{im}^l = 1, L_i = l) && (2)\\
&= \frac{P(T_i = t_i, \sum_{m=1}^{T} Y_{im}^l = 1, L_i = l | X, W(t_i))}{\sum_{j \in W(t_i)} P(T_i = j, \sum_{m=1}^{T} Y_{im}^l = 1, L_i = l | X, W(t_i))} && (3)\\
&= \frac{\lambda_i \exp\{\beta X_{it_i}^l + \gamma_{it_i}\}}{\sum_{j \in W(t_i)} \lambda_i \exp\{\beta X_{ij}^l + \gamma_{ij}\}} && (4)\\
&= \frac{\exp\{\beta X_{it_i}^l\}}{\sum_{j \in W(t_i)} \exp\{\beta X_{ij}^l\}} && (5)
\end{aligned}$$

which is free of terms λ_i and γ_{it_i}.

1.3 Derivation

The likelihood is defined as following, assuming subjects are independent.

$$\mathcal{L}(\beta) = \prod_{i=1}^{n} p_{it_i}^{l} = \prod_{i=1}^{n} \left(\frac{\exp\{\beta X_{it_i}^{l}\}}{\sum_{j \in W(t_i)} \exp\{\beta X_{ij}^{l}\}} \right) \tag{6}$$

Log-likelihood:

$$\ell(\beta) = \sum_{i=1}^{n} \log p_{it_i}^{l} = \sum_{i=1}^{n} \left(\beta X_{it_i}^{l} - \log \sum_{j \in W(t_i)} \exp\{\beta X_{ij}^{l}\} \right) \tag{7}$$

1.3.1 Derivation I

Take derivation directly:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{n} \left(X_{it_i}^{l} - \frac{1}{\sum_{j \in W(t_i)} \exp\{\beta X_{ij}^{l}\}} \sum_{m \in W(t_i)} \exp\{\beta X_{im}^{l}\} X_{im}^{l} \right) \tag{8}$$

$$= \sum_{i=1}^{n} \left(X_{it_i}^{l} - \sum_{m \in W(t_i)} X_{im}^{l} \frac{\exp\{\beta X_{im}^{l}\}}{\sum_{j \in W(t_i)} \exp\{\beta X_{ij}^{l}\}} \right) \tag{9}$$

This can be used as the updating rule for β. The complexity of this updating rule is correlated to number of subjects(persons) and the size of window, which can be written as $O(n|W|)$

1.3.2 Derivation II

This derivation will be based on group information. Denote the observed number of events Y_t^l at location l in interval time t is $Y_t^l = \sum_{i \in \mathcal{I}} Y_{it}^l$, where $\mathcal{I}$ the subjects satisify the same time t, same location l and same exposures X.

If we assume the group subjects share the same exposure, $X_{it}^l = X_t^l$, the Log-likelihood could be written as

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{n} \left(X_{it_i}^l - \sum_{m \in W(t_i)} X_{im}^l \frac{\exp\{\beta X_{im}^l\}}{\sum_{j \in W(t_i)} \exp\{\beta X_{ij}^l\}} \right) \tag{10}$$

$$= \sum_{l \in L} \sum_{i \in l} \left(X_{it_i}^l - \sum_{m \in W(t_i)} X_{im}^l \frac{\exp\{\beta X_{im}^l\}}{\sum_{j \in W(t_i)} \exp\{\beta X_{ij}^l\}} \right) \tag{11}$$

$$= \sum_{l \in L} \left[\sum_{i \in l} \left(X_{t_i}^l - \sum_{m \in W(t_i)} X_m^l \frac{\exp\{\beta X_m^l\}}{\sum_{j \in W(t_i)} \exp\{\beta X_j^l\}} \right) \right] \tag{12}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} Y_t^l \left(X_t^l - \sum_{m \in W(t)} X_m^l \frac{\exp\{\beta X_m^l\}}{\sum_{j \in W(t)} \exp\{\beta X_j^l\}} \right) \right] \tag{13}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} Y_t^l X_t^l - \sum_{t=1}^{T} Y_t^l \sum_{m \in W(t)} X_m^l \frac{\exp\{\beta X_m^l\}}{\sum_{j \in W(t)} \exp\{\beta X_j^l\}} \right] \tag{14}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} Y_t^l X_t^l - \sum_{t=1}^{T} \sum_{m=1}^{T} Y_t^l X_m^l \frac{I(m \in W(t)) \exp\{\beta X_m^l\}}{\sum_{j=1}^{T} I(j \in W(t)) \exp\{\beta X_j^l\}} \right] \tag{15}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} Y_t^l X_t^l - \sum_{m=1}^{T} \left(X_m^l \sum_{t=1}^{T} Y_t^l \frac{I(m \in W(t)) \exp\{\beta X_m^l\}}{\sum_{j=1}^{T} I(j \in W(t)) \exp\{\beta X_j^l\}} \right) \right] \tag{16}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} Y_t^l X_t^l - \sum_{t=1}^{T} \left(X_t^l \sum_{m=1}^{T} Y_m^l \frac{I(t \in W(m)) \exp\{\beta X_t^l\}}{\sum_{j=1}^{T} I(j \in W(m)) \exp\{\beta X_j^l\}} \right) \right] \tag{17}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} X_t^l \left(Y_t^l - \sum_{m=1}^{T} Y_m^l \frac{I(t \in W(m)) \exp\{\beta X_t^l\}}{\sum_{j=1}^{T} I(j \in W(m)) \exp\{\beta X_j^l\}} \right) \right] \tag{18}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} X_t^l \left(Y_t^l - \sum_{m \in R(t)} Y_m^l \frac{\exp\{\beta X_t^l\}}{\sum_{j \in W(m)} \exp\{\beta X_j^l\}} \right) \right] \tag{19}$$

$$= \sum_{l \in L} \left[\sum_{t=1}^{T} X_t^l \left(Y_t^l - \exp\{\beta X_t^l\} \sum_{m \in R(t)} \frac{Y_m^l}{\sum_{j \in W(m)} \exp\{\beta X_j^l\}} \right) \right] \tag{20}$$

Now the updating rule has been transformed into the one related to number of locations, times and references window size. The time complexity is $O(LT|W|)$. The advantage of this updating method is that we shrink lots of duplicated persons row data into much smaller number of groups, which share the same location l and exposures(features) X_t^l at the same time t.

2 Cox Proportional Hazards

2.1 Model

The hazard function for the Cox proportional hazard model has the form:

$$h(y_i|\boldsymbol{\beta}) = h_0(y_i|\boldsymbol{\beta}) \exp(\boldsymbol{\beta}^T \boldsymbol{x}_i)$$

where

- $h_0(y_i|\boldsymbol{\beta})$: the unspecified baseline hazard function.

- $i \in [1, n]$: each individual and n is the total number of individuals. .
- $y_i = \min(t_i, c_i)$: t_i is time-to-event(failure time) and c_i is right-censoring time.
- $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$: p-vector of features for the individual i.
- $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^T$: p-vector of underlying model parameters.

The n observed data $\boldsymbol{D} = \{(y_i, \delta_i, \boldsymbol{x}_i) : i = 1, \ldots, n\}$, where $\delta_i = I(t_i \leq c_i)$ is an indicator variable such that $\delta_i = 1$ if the observation is not censored and 0 otherwise.

2.2 Partial Likelihood

To estimate the underlying parameters $\boldsymbol{\beta}$, the original likelihood $L(\boldsymbol{\beta}|\boldsymbol{D})$ is hard to maximize. Cox proposed to maximize the partial likelihood:

$$L_p(\boldsymbol{\beta}|\boldsymbol{D}) = \prod_{i=1}^{n} \left(\frac{\exp(\boldsymbol{\beta}^T \boldsymbol{x}_i)}{\sum_{t \in R(y_i)} \exp(\boldsymbol{\beta}^T \boldsymbol{x}_t)} \right)^{\delta_i}$$

where $R(y_i)$ is the risk set of the i-th observation, defined as $R(y_i) = \{t : y_t \geq y_i\}$

2.3 Estimate Parameters

Maximizing partial likelihood is equivalent to maximize log-partial likelihood:

$$l_p(\boldsymbol{\beta}|\boldsymbol{D}) = \sum_{i=1}^{n} \delta_i \left\{ \boldsymbol{\beta}^T \boldsymbol{x}_i - \log \left(\sum_{t \in R(y_i)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_t\right) \right) \right\}$$

The negated log-partial likelihoods are convex, and a wide range of optimization algorithms can be utilized. In our experiments, we apply Limited-memory BFGS algorithm to minimize the negated log-partial likelihoods:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} -l_p(\boldsymbol{\beta}|\boldsymbol{D})$$

To apply L-BFGS, we have to calculate the first derivatives of $-l_p(\boldsymbol{\beta}|\boldsymbol{D})$ with respect to $\boldsymbol{\beta}$:

$$-l_p'(\beta_j) = -\sum_{i=1}^{n} \delta_i \left(x_{ij} - \frac{\sum_{t \in R(y_i)} x_{tj} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_t\right)}{\sum_{t \in R(y_i)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_t\right)} \right)$$

To produce approximate standard errors for the regression coefficients, we need to calculate the second derivatives:

$$-l_p''(\beta_j) = \sum_{i=1}^{n} \delta_i \left\{ \frac{\sum_{t \in R(y_i)} x_{tj}^2 \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_t\right)}{\sum_{t \in R(y_i)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_t\right)} - \left(\frac{\sum_{t \in R(y_i)} x_{tj} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_t\right)}{\sum_{t \in R(y_i)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}_t\right)} \right)^2 \right\}$$

3 Proportional Hazard Model with Frailties

3.1 model

For estimation of zip code specific long-term air-pollution mortality risks, we will consider proportional hazard model with multivariate random effects(frailties). For this model, event times from the same group (zip code area) are likely to be correlated. Suppose there are C number of clusters(zip code areas). Then the proportional hazard model with frailties has the form:

$$h(y_{ij}|\boldsymbol{\beta}_i) = h_0(y_{ij}|\boldsymbol{\beta}_i) \exp\left(\boldsymbol{\beta}_i^T \boldsymbol{x}_{ij} + b_i\right)$$

where

- $h_0(y_{ij}|\boldsymbol{\beta}_i)$: the baseline hazard function.
- $i \in [1, C]$: each clusters, C is the total number clusters(zip code areas).
- $j \in [1, n_i]$: each individual from cluster i, and n_i is the total number of individual from ith cluster.
- $y_{ij} = \min(t_{ij}, c_{ij})$: where t_{ij} is the failure time, and c_{ij} is the right-censorinbg time.
- δ_{ij}: is an indicator variable such that $\delta_{ij} = 1$ if the observation is not censored and 0 otherwise.
- $\boldsymbol{x}_{ij}$: p-vector of features for the individual j in cluster i.
- $\boldsymbol{\beta}_i = (\beta_{i1}, \beta_{i2}, \ldots, \beta_{ip})^T$: p-vector of cluster-specific underlying model parameters.
- b_i: the cluster specific random effects.

3.2 Estimate Parameters

To solve the frailty model, serveral methods have been proposed these years. Xue and Ding (1999) used a Gibbs Sampling approach. Ripatti and Palmgren (2000) considered a penalized partial likelihood approach. Vaida and Xu (2000) proposed a nonparametric maximum likelihood estimator, obtained using a Monte Carlo EM algorithm. Cortinas-Abrahantes et al. A comprehensive comparison of these methods can be found in Gamst et al.(2009). We will follow one of these methods mentioned above or other related methods, to solve this problem.

References

[1] Jose Cortinas Abrahantes and Tomasz Burzykowski. A version of the em algorithm for proportional hazard model with random effects. 2005.

[2] Cox R David. Regression models and life tables (with discussion). *Journal of the Royal Statistical Society*, 34:187–220, 1972.

[3] Anthony Gamst, Michael Donohue, and Ronghui Xu. Asymptotic properties and empirical evaluation of the npmle in the proportional hazards mixed-effects model. *Statistica Sinica*, pages 997–1011, 2009.

[4] Kyeong Eun Lee, Yongku Kim, and Ronghui Xu. Bayesian variable selection under the proportional hazards mixed-effects model. *Computational statistics & data analysis*, 75:53–65, 2014.

[5] Tao-Wen Liu. A regularized limited memory bfgs method for nonconvex unconstrained minimization. *Numerical Algorithms*, 65(2):305–323, 2014.

[6] Yun Lu and Scott L Zeger. On the equivalence of case-crossover and time series methods in environmental epidemiology. *Biostatistics*, 8(2):337–344, 2007.

[7] Sushil Mittal, David Madigan, Randall S Burd, and Marc A Suchard. High-dimensional, massive sample-size cox proportional hazards regression for survival analysis. *Biostatistics*, page kxt043, 2013.

[8] Samuli Ripatti and Juni Palmgren. Estimation of multivariate frailty models using penalized partial likelihood. *Biometrics*, 56(4):1016–1022, 2000.

[9] Florin Vaida and Ronghui Xu. Proportional hazards model with random effects. *Statistics in medicine*, 19(24):3309–3324, 2000.

Appendix B

Supplemental for Cox Proportional Hazard Model

Supplementary Online Content

Table of Contents

Appendix S1. Java Implementation of Cox Proportional Hazards Models

To perform our analyses of 12-month $PM_{2.5}$ exposures and cause-specific mortality on our large-scale data, we implemented both linear and non-linear Cox PH methods in Java. Our implementation overcame memory and processing limitations of conventional software packages, such as R and SAS, using data grouping and linkage methods, optimization techniques, and multi-threading, as detailed below. Our Java implementation will be hosted on GitHub once our paper is accepted. These methods were able to run our models based on data for the whole country in approximately ten minutes. Below we briefly present our data grouping methods, followed by our Cox PH re-implementation for the linear model, then the restricted cubic spline, and finally their validation.

Data Processing. We created joint datasets that minimize redundant entries by aggregating rows with the same attributes by ZIP code and month and by adding to these joint datasets two counting attributes: the number of deaths and the total population for the ZIP code and month.

Re-Implementation of Linear Cox PH model. The original Cox PH estimates air pollution-associated mortality risks using stratum-specific baselines[1]:

$$h(t_i|X_i, s_i) = h_{0_s}\exp(\beta X_i), \qquad (1)$$

where $i \in [1, n]$ represents each individual i, with n the total number of individuals. h_{0_s} is a stratum-specific baseline hazard function, $y_i = min(t_i, c_i)$, where t_i is event time and c_i the right-censoring time for each individual i. In addition, let $\boldsymbol{X_i} = \left(x_{i1}, x_{i2}, \dots, x_{ip}\right)^T$ be a p-vector of covariates for the individual i, and $\boldsymbol{\beta} = \left(\beta_1, \beta_2, \dots, \beta_p\right)^T$ be the p-vector of estimating model parameters. The n observed data $D = \{(y_i, \delta_i, \boldsymbol{x_i}): i \in [1, n]\}$, where $\delta_i = I(t_i \leq c_i)$ is an indicator variable such that $\delta_i = 1$ if the observation is not censored and 0 otherwise.

To simplify the problem, a partial likelihood function of Cox PH was proposed by Cox [1]:

$$L_p(\boldsymbol{\beta}|D) = \prod_{i=1}^{n}\left(\frac{\exp(\boldsymbol{\beta^T x_i})}{\sum_{t\in R(y_i)}\exp(\boldsymbol{\beta^T x_t})}\right)^{\delta_i} \qquad (2)$$

where $R(y_i)$ is the risk set of the given individual i, and $R(y_i) = \{t: y_t \geq y_i\}$, representing any individual t, who has survived at least longer than individual i. In general, we will assume that there are no tied survival times. Therefore, in order to break the ties, Mittal et al.[2] proposed to add very small random number (uniform from $[-10^{-5}, 10^{-5}]$) to the event time. Since the survival time order, but not actual event time, is used when updating the model, we simply shuffle the risk set and end up with an ordered event times to break the ties in our implementation. Furthermore, optimizing the above partial likelihood is equivalent to estimate the partial log-likelihood function:

$$l_p(\beta|D) = \sum_{i=1}^{n}\delta_i\left\{\boldsymbol{\beta^T x_i} - \log\left[\sum_{t\in R(y_i)}\exp(\boldsymbol{\beta^T x_t})\right]\right\} \qquad (3)$$

To simplify the model explanation, we omit the penalty term of $\boldsymbol{\beta}$ here; however, the L2 penalty term has been adopted in our Cox PH model. Fortunately, Eq (3), even with the L2 penalty term, is a convex function, and a wide range of optimization algorithms can be utilized. We chose to employ the Limited-memory BFGS(L-BFGS) algorithm[3] to minimize the negative log-partial likelihoods:

$$\boldsymbol{\beta}^* = argmin_{\boldsymbol{\beta}} - l_p(\boldsymbol{\beta}|D)$$

L-BFGS is a limited memory quasi-Newton methods for large scale optimization, which has been employed as one of the most popular optimization algorithms in machine learning[4, 5]. As a quasi-Newton method, L-BFGS is not required to compute the Hessian matrix of variables, but to estimate an approximation of Hessian with only a few vectors. Thus, the L-BFGS method is particularly well suited for optimization problems with a large number of variables. Even though, the optimization is still expensive, since the mathematical operations on millions of subjects require massive computation. Since the number of variables and the corresponding number of categories for each variable are very few, it is not hard to notice that lots of the rows share the exactly the same covariates. As a result of this, lots of productions of estimated parameter $\boldsymbol{\beta}$ and covariates $\boldsymbol{x}$ are the same. If subjects share the same variables (x_g), we simply group them together, with two counts for each group g: number of deaths (δ_g) and total number of people (y_g), defined as:

$$\delta_g = \sum_{i \in g} \delta_i$$

$$y_g = \sum_{i \in g} 1 = |g|$$

By introducing groups, we could redefine Eq (3) as:

$$l_p(\beta|D,G) = \sum_{g \in G} \delta_g \left\{ \boldsymbol{\beta^T x_g} - \log\left[\sum_{t \in R(y_g)} \mathrm{y_g} \exp(\boldsymbol{\beta^T x_t}) \right] \right\} \quad (4)$$

where G is the total distinct groups from the whole dataset. We could also roughly estimate the upper bound number of distinct groups within G. Assuming variables only contain gender, race, age and ZIP code, wherein ZIP code has distinct $PM_{2.5}$, leading to that the variables from these ZIP areas are all different, we have 40,000 (number of ZIP Codes) distinct variables at least, and 2 types of gender, 2 groups of race and 26 age categories. In total, we have $40{,}000 * 2 * 2 * 26 \cong 4\ million$ distinct groups at a time point. Comparing to computing productions, exponentiations and logarithms on 64 million enrollees, we only need at most $\frac{1}{16}$ computation power, which saving almost 95% time.

Suppose there are S strata in total, and then individuals are split into S strata, the partial log-likelihood Eq (4) can be considered as one component from a specific strata or group G. Overall, the partial log-likelihood for Strata Cox PH model can be written as:

$$l_p(\beta|D,S) = \sum_{G \in S} l_p(\beta|D,G) \quad (5)$$

which is still a convex optimization problem.

Restricted Cubic Splines. To Study the non-linear relationship between the cause of death and exposures, we implemented Restricted Cubic Splines (RCS)[6] in the Strata Cox PH model. The mechanism of RCS is to split up the continuous range of predictor variables with a few number (l) of pre-defined "knots", which are written as $k_1, k_2, \ldots, k_l$ in an ascending order. With l knots, a size of $l-2$ new variables will be generated for each original variable (x) by the following formula:

$$x^i = (x-k_i)_+^3 - (x-k_{l-1})_+^3 \frac{k_l - k_i}{k_l - k_{l-1}} + (x-k_l)_+^3 \frac{k_{l-1} - k_i}{k_l - k_{l-1}} \quad (6)$$

For $i = 1, 2, \ldots, l-2$. And u_+ is defined as:

$$u_+ = \begin{cases} u & if\ u > 0 \\ 0 & otherwise \end{cases}$$

Exposure-Response Curves. We examined the shape of the association of $PM_{2.5}$ exposure and cause-specific mortality using restricted cubic splines in Cox PH models with 3 knots[6], with the locations of the knots as [0.1, 0.5, 0.9] in quantiles of each continuous variable in our study. We note that restricted cubic splines are designed differently in some existing software packages, such as **rcspline.eval**[7] in R, wherein normalization is introduced to the new variables to reduce ill-conditioning problems. We applied the $norm = 2$ settings[7], which normalizes each new variable by the two-thirds of the spacing between the first and last knots. Therefore, the new generated variables can be written as:

$$x^i_{norm} = \frac{x^i}{(k_l - k_1)^{\frac{2}{3}}} \quad (7)$$

wherein x^i is calculated from Eq (6). The normalization takes the advantage of making all new generated non-linear terms be on the original variable scale[7].

Model Validation. To verify of our Java implementation of the stratified Cox PH model with and without restricted cubic splines, we created a sample dataset of 10,000 subjects, which is sufficiently small to run our full analysis in R. For each subject, we included data on age (65-90), gender (male or female), race (White, Black, Hispanic, Asian,

other), location (800 sites), date (120 total months), and death (0 or 1), totaling 1,011,945 subject-date records. We applied the Cox PH function **coxph** from the "survival" package in R[8], with strata on age, gender and race, and Restricted Cubic Splines for the exposure using **rcspline.eval** from "Hmisc" package in R[7]. Knots were specified at the 10th, 50th, and 90th percentiles. We also used the same settings in our Java implementation of these models and compared our findings to those from R. As shown in Table S1, our Java implementation provided almost identical risk estimates and standard errors to those obtained in R.

Table S1. Validation of Java implementation of Cox PH models using public package in R[a]

Estimators	Linear Cox PH		Non-Linear Cox PH	
	R	Java	R	Java
β_x **(se)**	0.00906 **(0.00895)**	0.00904 **(0.00895)**	0.00948 **(0.02367)**	0.00940 **(0.02367)**
$\beta_{x1_{norm}}$ **(se)**	-	-	0.00197 **(0.01982)**	0.00205 **(0.01982)**

[a] Models include strata for age, gender, race, and ZIP code. Non-linear models estimated using restricted cubic splines with 3 knots.

References

1. Cox DR. Regression models and life-tables. *J R Stat Soc Ser B*. 1972;34(2):187-202.
2. Mittal S, Madigan D, Burd RS, et al. High-dimensional , massive sample-size Cox proportional hazards regression for survival analysis. 2014:207-221. doi:10.1093/biostatistics/kxt043
3. Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. *Math Program*. 1989;45(1-3):503-528.
4. Malouf R. A comparison of algorithms for maximum entropy parameter estimation. In: *Proceedings of the 6th Conference on Natural Language Learning-Volume 20*. Association for Computational Linguistics; 2002:1-7.
5. Andrew G, Gao J. Scalable training of L 1-regularized log-linear models. In: *Proceedings of the 24th International Conference on Machine Learning*. ACM; 2007:33-40.
6. Croxford R. Restricted Cubic Spline Regression : A Brief Introduction. *Toronto Inst Clin Eval Sci*. 2016:1-5.
7. Harrell Jr FE, Harrell Jr MFE. Package 'Hmisc.' 2019.
8. Therneau TM, Lumley T. Package 'survival.' *Surviv Anal Publ CRAN*. 2014.