Learning Label Structure for Compressed Sensing Based Multilabel Classification

Subhojit Som
Microsoft Corporation
1 Microsoft Way, Redmond, WA 98052, USA
Email: Subhojit.Som@microsoft.com

***Abstract*—We present a compressed sensing based approach to multilabel classification that exploits the label structure present in many multilabel applications. The compressed sensing method exploits the sparsity in the label vector. The label vector is projected to a lower dimensional space by a random projection matrix. From the training data we learn how to predict the projected vector directly from the features of the samples. For a new test sample, we first predict the projected vector and then use compressed sensing recovery algorithm to estimate the sparse label vector. For many practical scenarios the label vector is not only sparse but the active labels represent a context or theme; hence have a structure. In this paper we propose to learn the label structure instead of considering the individual labels to be independent and identically distributed. We assume a Bayesian model for the labels and model the label structure as latent tree. We learn the label structure from the training data and use the learned structure during estimation of the label vector from predicted projections. Furthermore, we propose a new structure learning approach where we hash the labels into smaller number of buckets and learn the structure from these buckets. This significantly reduces the computational complexity without sacrificing accuracy. We present numerical results to demonstrate this approach and its benefit.**

***Keywords*—multilabel classification; compressed sensing; structure learning**

I. Introduction

Multilabel classification is a supervised classification problem where each sample (instance) can have multiple labels. We represent the label of the i^{th} sample by binary label vector $\mathbf{y}^i = (y_1^i, y_2^i, \ldots, y_L^i) \in \{0,1\}^L$, where L is the cardinality of the label set. Here y_l^i is the indicator function of l^{th} label and $y_l^i = 1$ implies label l is active for sample i. The task is to predict the vector $\mathbf{y}$ for a new sample.

Multilabel classification problems arise in many different domains including image, text, audio, video etc. and have received increased attention in recent times (see [1], [2] and the references therein). The most popular approach for multilabel classification problem is to train one learner per label and then aggregate those learner outputs to solve the overall prediction problem. This is known as Binary Relevance (BR) in the literature [2]. The number of labels in web-scale multilabel applications like tag suggestion, keyword recommendation, object recognition etc. have grown very large. Hence training one learner per label has become unrealistic for such applications. One approach to address the issue is to use lossy label transformation. Compressive sensing [3], principal label space transformation [4], Bloom filter [5] based methods are examples of lossy label transformation approach for multilabel classification. These algorithms transform the label vector to a smaller dimensional subspace in order to achieve computational efficiency at the cost of prediction accuracy.

For most applications the multiple labels of the samples are not independent of each other and their relationship can be modeled statistically. In multiple object detection problem, an image can have multiple objects and those objects are typically related. The statistical model of the objects captures the context, e.g., in an image of an office room, several of the objects like computer, mouse, keyboard, book, table, chair etc. are highly likely to be present. Presence of some of them increases the likelihood of presence of the others and at the same time decreases the likelihood of the presence of an unrelated object, for example a boat. Under this statistical model, the elements in the sparse label vector are not independent and identically distributed (i.i.d.) and if this relationship between objects (structure) is used properly by the CS reconstruction algorithm then the prediction accuracy can be improved.

Label structure has been used extensively in multilabel classification algorithms in various forms including other lossy label transformation based methods but it has not been used for compressed sensing based methods. [6] uses tree/DAG label structure to improve the prediction accuracy of principal component analysis (PCA) based lossy principal label space transformation method. Instead of learning like ours, this algorithm assumes that the label structure is given. There are other works where label tree structure is assumed to be given [7], [8]. Learning label dependency is also very common in multilabel prediction but this dependency, in the form of tree hierarchy, is used for building classifiers rather than jointly making probabilistic inference [9].

In this paper, we show the impact of using label structure in compressed sensing based multilabel classification algorithms. We show that by simply assuming the non-identical nature of the labels can improve the prediction accuracy. We estimate the marginal distribution of the labels from the training data but still assume them to be independent and use that as prior in a Bayesian compressed sensing algorithm known as Approximate Message Passing (AMP) [10], [11]. We further propose to learn a latent tree structure [12] for sparse label space which approximates the joint distribution of the labels. We use an iterative message passing called turbo AMP [13], [14], which can exploit structured sparsity of the label vector to further improve the prediction accuracy, especially at the high compression ratios. Although we assume a latent tree model, any other model can also be used. In addition, we also propose a two step structure learning method that further exploits

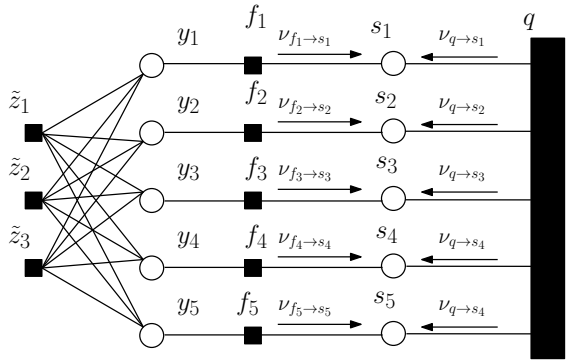

Fig. 1: Factor graph representation of the model and turbo message passing scheme. The left side of nodes s_l denote the SPE and it represents the projection matrix $\mathbf{A}$. Functionally it is the CS recovery block. The right side is the SPD block which represents the structure or joint distribution $p(\mathbf{s})$.

sparsity, uses hashing trick and imposes more constraint on the structure but reduces complexity of the structure learning algorithm with negligible loss in accuracy. Using benchmark datasets we demonstrate the performance of the proposed approach.

II. Compressed Sensing Bases Multilabel Classification

For large scale multilabel problems, although the label space is very high dimensional, in most applications, the label vectors are sparse, i.e., $||\mathbf{y}^i||_0 \ll L$, where $||\mathbf{y}^i||_0$ is the number of active labels for sample i. Compressive sensing (CS) based approach [3] exploits the label sparsity to significantly reduce the number of learners to be trained.

In compressed sensing [15], [16], a high dimensional sparse signal is reconstructed from very small number of measurements. More specifically, a K-sparse vector in $\mathbb{R}^L$ can be recovered optimally from M measurements $\mathbf{z}^i = (z_1^i, z_2^i, \ldots, z_M^i) \in \mathbb{R}^M$

$$\mathbf{z}^i = \mathbf{A}\mathbf{y}^i, \tag{1}$$

if the measurement[1] matrix $\mathbf{A} \in \mathbb{R}^{L\times M}$ satisfies certain non-coherence properties like *Restricted Isometry Property* (RIP) [16] or *Coherence* [15], where $M = O(K \log L/K) \ll L$. In this paper we present Gaussian random matric Randomly generated matrices such as Gaussian, Bernoulli and Hadamard show RIP property.

In the CS based multilabel classification problem, we fix a realization of the random projection matrix $\mathbf{A}$. The M predictors $\{h^m\}, m = 1, 2, \ldots, M$ are learned from the training data which directly predict the compressed vector $\mathbf{z} \in \mathbb{R}^M$ instead of label vector $\mathbf{y} \in \mathbb{R}^L$. Since $M \ll L$ the number of predictors to be learned is significantly reduced. Once the compressed vector is predicted for a new sample, any compressed sensing algorithm can be used to recover the label vector. Among CS reconstruction algorithms, greedy algorithms Orthogonal Matching Pursuit (OMP), CoSaMP and convex optimization Lasso were used in [3], Bayesian Compressed Sensing (BCS) was used in [17], [18].

[1]In this paper we use the terms measurement, projection and compression interchangeably.

III. Generative model

We use a Bayesian framework for solving the structured sparse CS problem. We assume a generative model for the labels

$$p(y_l \mid s_l) \quad = \quad s_l \mathcal{N}(y_l; 0, \sigma_s^2) + (1 - s_l)\delta(y_l), \tag{2}$$

where $s_l \in \{0, 1\}$ denotes a hidden binary indicator variable that denotes whether the label is active and model y_l as conditionally Gaussian. Thus we model y_l as real valued which is implicit in all prior CS based multilabel work. Moreover, modeling y_l as zero mean Gaussian distributed is a standard practice in Bayesian CS based methods for multilabel prediction [17]. Henceforth we will consider $\mathbf{y} \in \mathbb{R}^L$ instead of being binary.

Figure 1 shows the factor graph of the label and the compressive projection model. The square nodes are function nodes and the circles are the variable nodes [13]. In the graph, factor q denotes the label structure $p(\mathbf{s})$ which can be any distribution. The model captures a structure when label indicator variables (s_l) are not independent. The factor nodes f_l represents the conditional distribution $p(y_l \mid s_l)$ in (2). The dense factor graph on the left denotes the observed compressive projection corrupted by additive white Gaussian noise $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_w^2\mathbf{I})$

$$\tilde{\mathbf{z}} = \mathbf{A}\mathbf{y} + \mathbf{w}. \tag{3}$$

For the CS recovery step, the vector $\tilde{\mathbf{z}}$ can be considered to be prediction of the projection vector for a new sample and its m^{th} element $\tilde{z}_m$ is predicted by the corresponding regressor h_m that we learned from the training data.

IV. Proposed Method

In this section we outline the complete method of multilabel classification using structured sparse compressed sensing. The algorithm proceeds with the following steps.

1) **Training:**
   a) **Projection:** We first take projection of the label vectors $\mathbf{y}$ of the training data using a fixed realization of a random projection matrix $\mathbf{A}$ via (1) to obtain M dimensional projected vectors $\mathbf{z}$.

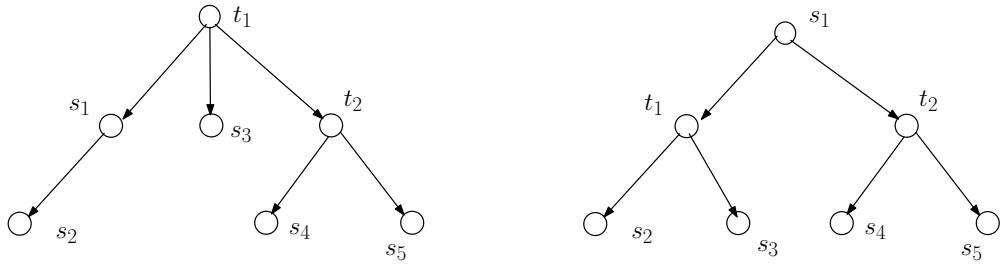

Fig. 2: Two possible latent tree structures for the label vector shown in Figure 1. The s nodes can be leaf, intermediate or even root node but t nodes cannot be a leaf node.

b) **Learning regressors:** We learn M different regressors $\{h_m\}$ from the training data which can map the feature vectors to $z^m, m \in \{1, 2, \dots M\}$, i.e., we learn one regressor per dimension of the projected space.

c) **Learning Label Structure:** We learn a joint distribution for the label structure $p(\mathbf{s})$ from the training data.

2) **Prediction:**

a) **Predict projection vector:** We first predict $\tilde{\mathbf{z}}$ from feature vector of the sample using the M regressors learned in step 1b.

b) **Structured sparse CS recovery:** Estimate label vector $\mathbf{y}$ or label activity indicator vector $\mathbf{s}$ from $\tilde{\mathbf{z}}$ using a compressed sensing reconstruction method that uses label structure $p(\mathbf{s})$ learned in step 1c to improve accuracy.

The approach of learning $p(\mathbf{s})$ in step 1c and using it in step 2b to improve accuracy of prediction is the first contribution of this paper. We assume a latent tree model for $p(\mathbf{s})$ and use the Chow Liu Recursive Grouping (CLRG) algorithm proposed in [12] for learning $p(\mathbf{s})$. For structured sparse CS recovery we use the turbo AMP algorithm [14]. We would like to emphasize here that although we use a tree structure and use CLRG for learning that structure, the algorithm above is generic and works for any distribution $p(\mathbf{s})$ and any structure learning algorithm for $p(\mathbf{s})$. We also propose a more efficient and scalable approach for learning a latent tree structure that uses label sparsity and hashing trick where we impose some constraint on the tree structure. Empirical results show that this constraint do not lead to any significant loss of accuracy. This is our second contribution. In the following sections we go over these steps in further detail.

A. Structured sparse CS inference

In this section we review the turbo AMP algorithm. When binary indicator variable $\mathbf{s}$ is structured sparse with a known distribution $p(\mathbf{s})$, turbo inference algorithm can be used to estimate posterior distribution of y_l and s_l from compressed observations $\tilde{\mathbf{z}}$. The turbo inference is a loopy belief propagation [19] based algorithm that works in an iterative fashion by passing messages between two blocks known as sparsity pattern equalizer (SPE) and sparsity pattern decoder (SPD) [13]. In Figure 1, the SPE block is the left side of the variable nodes s_l and the SPD is the right side of s_l. Messages exchanged between these two blocks are denoted by $\nu_{f_l \to s_l}$ and $\nu_{q \to s_l}$. Both of these blocks implement sum-product belief propagation individually. SPE exploits the compression structure and the graph represents the projection matrix $\mathbf{A}$. The SPD exploits the label structure and represents the joint distribution $p(\mathbf{s})$. Messages $\nu_{q \to s_l}$ that go from SPD to SPE work like effective priors on the variables s_l. Thus SPE works under non-identical prior (belief) on s_l and tries to infer y_l and s_l based on that belief, observed vector $\tilde{\mathbf{z}}$, known projection matrix $\mathbf{A}$ and the model parameters σ_s^2 and σ_w^2. After doing few iterations of loopy belief propagation, SPE sends messages $\nu_{f_l \to s_l}$ to SPD. Based on this belief and the structure $p(\mathbf{s})$ SPD decodes the s_l and generates messages $\nu_{q \to s_l}$ which are in turn sent to SPE. For a latent tree, sum-product belief propagation is exact since there is no loop in the graph and only one iteration of internal message passing is needed. The turbo scheme is motivated by turbo equalizer in digital communication where encoded bits received via analog channel are being decoded. The analog channel is analogous to the compression in CS. The bits are encoded in a way that all combinations are not equally likely or valid and this is analogous to the label structure $p(\mathbf{s})$.

For SPE, which is the CS recovery block, we use a loopy belief propagation based inference algorithm known as Approximate Message Passing (AMP) [10]. AMP has shown better estimation accuracy than greedy CS algorithms like OMP, CoSaMP etc. It has better computational complexity than convex ℓ_1 optimization algorithms like Lasso with similar estimation accuracy [10], [20]. SPE gets the belief messages $\nu_{q \to s_l}$ which work as non-uniform prior on each s_l. [11] shows that if different s_l variables have non-identical priors of being active, i.e., $p(s_l)$ are not equal for all l, and this prior is provided to AMP (by an oracle), then the prediction accuracy improves over the scenario when this prior is not known to AMP and the improvement is quantifiable under asymptotic settings. In the turbo AMP, the SPD works like an oracle by providing the belief messages $\nu_{q \to s_l}$. SPD generates $\nu_{q \to s_l}$ based on the input $\nu_{f_l \to s_l}$ from SPE which is the instance specific information and the generic information about the label structure $p(\mathbf{s})$.

In compressed sensing applications where turbo schemes have been applied, the support structure $p(\mathbf{s})$ is typically known. The edges of the graphs are known, e.g., in wavelet quad-tree structure or Markov random field. Sometimes the parameters are to be learned. In contrast, in multilabel classification problem the label structure is typically unknown. Thus we need to learn that structure (both the edges and their parameters) from the training data. In this paper we assume

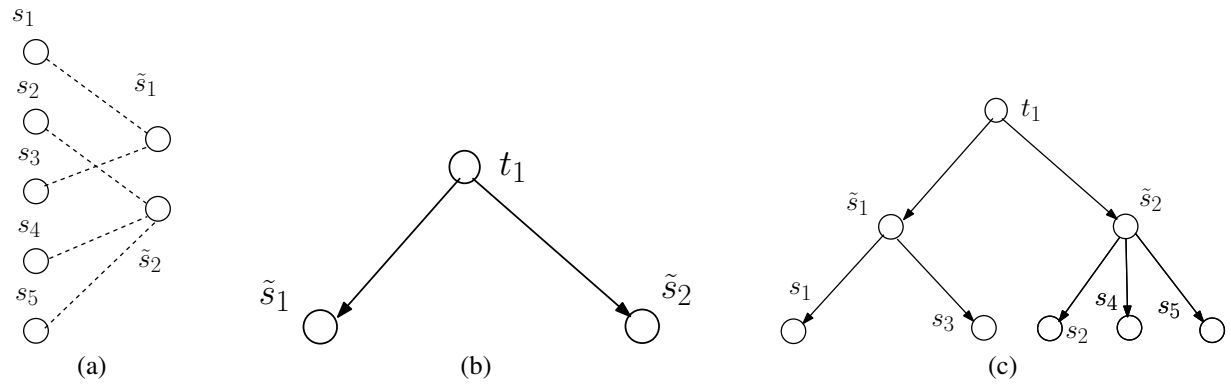

Fig. 3: Latent tree learning via hashing. (a) The nodes $s_1, \ldots, s_5$ denote the five indicator variables corresponding to the five lables. s_1 and s_3 are hashed to bucket $\tilde{s}_1$ and s_2, s_4 and s_5 are hashed to bucket $\tilde{s}_2$. (b) The structure learning algorithm learns the structure based on observed values of $\tilde{s}_1$ and $\tilde{s}_2$ from the training data. (c) Based on the hashing in (a), all nodes are added back and via EM algorithm all the edges are estimated.

latent tree structure for the label vector and discuss learning that structure in the next section.

B. Learning label structure

In this work, we assume a latent tree structure of the label space due to the following reasons. First, tree type hierarchy of labels is very intuitive representation of objects or topics. Second, learning a tree structure is tractable, and efficient algorithms exist for the same. Third, the turbo inference algorithm is a sum-product belief propagation algorithm and for the SPD exact inference can be done on a tree with just one cycle of message passing. Fourth, tree like structures of the label vector have been successfully applied to various multilabel prediction algorithms (although in different ways than our proposed method as briefly discussed in section I).

Unlike other tree structures typically used in multilabel prediction problems, where the root node includes all labels and then labels are divided into branches at every node, in this model we no not make any such assumption. We also learn latent or hidden nodes from the data (hence the name latent tree). Thus we learn the joint distribution $p(\mathbf{s}, \mathbf{t})$, where $\mathbf{t}$ denotes the vector of hidden nodes. The label distribution $p(\mathbf{s})$ can be obtained by marginalizing over $\mathbf{t}$. We assume that the hidden nodes are also binary. The hidden nodes in practice usually turn out to be context, theme or topic but we do not explicitly make any such assumption. We always learn minimal latent tree since for any given latent tree we can always add another hidden node (e.g., a new leaf latent node to the tree) keeping the marginal distribution $p(\mathbf{s})$ the same. The nodes in $\mathbf{s}$ can be both leaf nodes or intermediate or parent nodes. But the nodes in $\mathbf{t}$ can never be a leaf node, since it will not be a minimal latent tree anymore. Figure 2 depicts two possible latent tree models $p(\mathbf{s}, \mathbf{t})$ for the label vector in Figure 1 that illustrate this.

We now review the CLRG algorithm [12] for latent tree learning. The CLRG is a two step algorithm. In the first step, a global Chow-Liu tree [21] is learned based on max-weight spanning tree algorithm which can be solved by Kruskal's algorithm or Prim's algorithm in $O(L^2)$ time [22]. This tree places the nodes that should be far apart in the final latent tree at higher distance. Then a recursive grouping algorithm [12] builds the latent tree by placing hidden nodes appropriately. At every step in recursive grouping a very small subset of nodes need to be considered due to the Chow-Liu tree structure which places the candidate adjacent nodes within a small neighborhood of each other. Once the latent tree is built its parameter are learned by Expectation Maximization (EM) algorithm whose complexity is linear with L. The overall complexity of the CLRG algorithm is dominated by Chow-Liu tree construction and hence $O(L^2)$. We note here that for very large label space, the co-occurrence matrix of labels is very sparse too and that reduces the number of candidate edges in the graph. An adjacency list based implementation gives a complexity of $O(E \log L)$ [22], where E is the number of non-zero entries in label co-occurrence matrix from the entire training dataset.

V. Learning structure via label hashing

For high dimensional label vectors the structure learning can be computationally expensive because of high computational complexity of the CLRG algorithm. That motivates us to explore efficient structure learning schemes for label vector. Feature hashing is a very successful mechanism to handle very high dimensional feature vectors in machine learning [23], e.g., in large scale text classification problems. Open source algorithms like Vowpal Wabbit[2] use feature hashing to achieve scalability. Label hashing has also been successful for multi-label classification problem [5], which is a direct motivation for our approach.

The idea of label hashing is the following. We map the individual labels to some buckets and the number of buckets $B \ll L$. We design a hash function that maps each label to a bucket. Since the number of buckets is smaller than the number of labels there will be collision, i.e., the same bucket is used for multiple labels. For a well designed hash function or a random assignment, labels from one topic will not collide with labels from another topic all the time. That helps to preserve information about the labels. The results in [5] demonstrate that for multilabel classification when the label

[2]https://github.com/JohnLangford/vowpal_wabbit/wiki

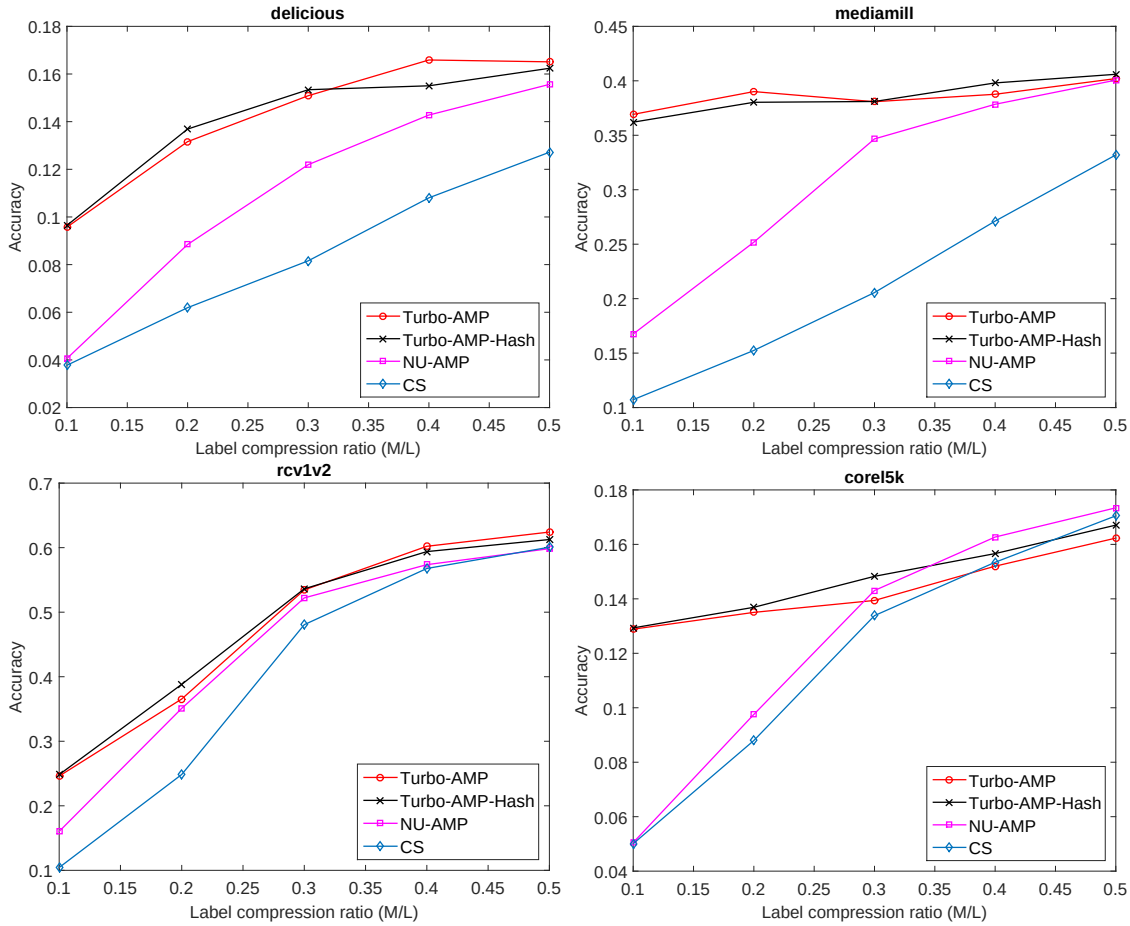

Fig. 4: Accuracy vs label compression ratio (M/L).

vector is sparse, well designed hash function can retain enough structural information of the label vector when compressed into the hashing buckets. In contrast to some popular hashing tricks like Bloom filter, we use a single hashing function instead of a set of hashing functions that give an array of buckets. In this paper we use a random assignment of labels to buckets as hashing method for simplicity. A structure-dependent hashing mechanism can be a potential future research work.

Let us explain the approach with the example label vector in Figure 1. Figure 3(a) shows the mapping via hashing function where the 5 label nodes are mapped to 2 hash buckets. The hashing buckets are represented by binary vector $\tilde{\mathbf{s}} \in \{0,1\}^B$ and is considered active if at least one of its corresponding labels is active. Else it is considered inactive. In the first step, we obtain the hashing buckets for all samples in the training dataset and then use the CLRG algorithm to learn the latent tree model $p(\tilde{\mathbf{s}}, \mathbf{t})$ as shown in Figure 3(b). In the second step, we augment the factor graph in Figure 3(b) with the label indicator nodes $\mathbf{s}$ and construct the edges connecting the $\mathbf{s}$ and $\tilde{\mathbf{s}}$ nodes according to the hash map in Figure 3(a). This augmented graph is shown in Figure 3(c). In this new graph both $\tilde{\mathbf{s}}$ and $\mathbf{t}$ become hidden nodes. We learn the parameters of this new model via EM algorithm. For initializing the node and edge potentials, we use the estimated parameter values from step 1 for nodes and edges present in graph of Figure 3(b). For the new edges connecting $\tilde{\mathbf{s}}$ and $\mathbf{s}$, we initialize with very small values for $p(s_l = 1 \,|\, \tilde{s}_b = 0)$ and moderate values of $p(s_l = 1 \,|\, \tilde{s}_b = 1)$ with the intuition that they represent conditional probability of a label being active given the corresponding bucket being inactive and active respectively. In our experiments we initialize these edge parameters with 0.005 and 0.2 respectively.

Learning the active edges of the graph is $O(L^2)$ and it is the high complexity part of the overall structure learning process. Since the number of edges in the tree is linear with the number of nodes the parameter learning via EM is $O(L)$. Step 1 involves both finding the active edges and parameter learning via EM but step 2 involves only parameter learning via EM. Thus we are pushing the high complexity part of the overall structure learning to step 1 where the number of nodes (hashing buckets) are an order or more smaller than the number of nodes (labels) in the step 2.

VI. Experimental Results

In this section we provide numerical results using our proposed algorithms on standard benchmark multilabel datasets. We demonstrate that using label structure in compressed sensing helps to improve prediction accuracy across these datasets, specially at high compression region (i.e., low M/L) which is our target operating zone. We also show that when we use

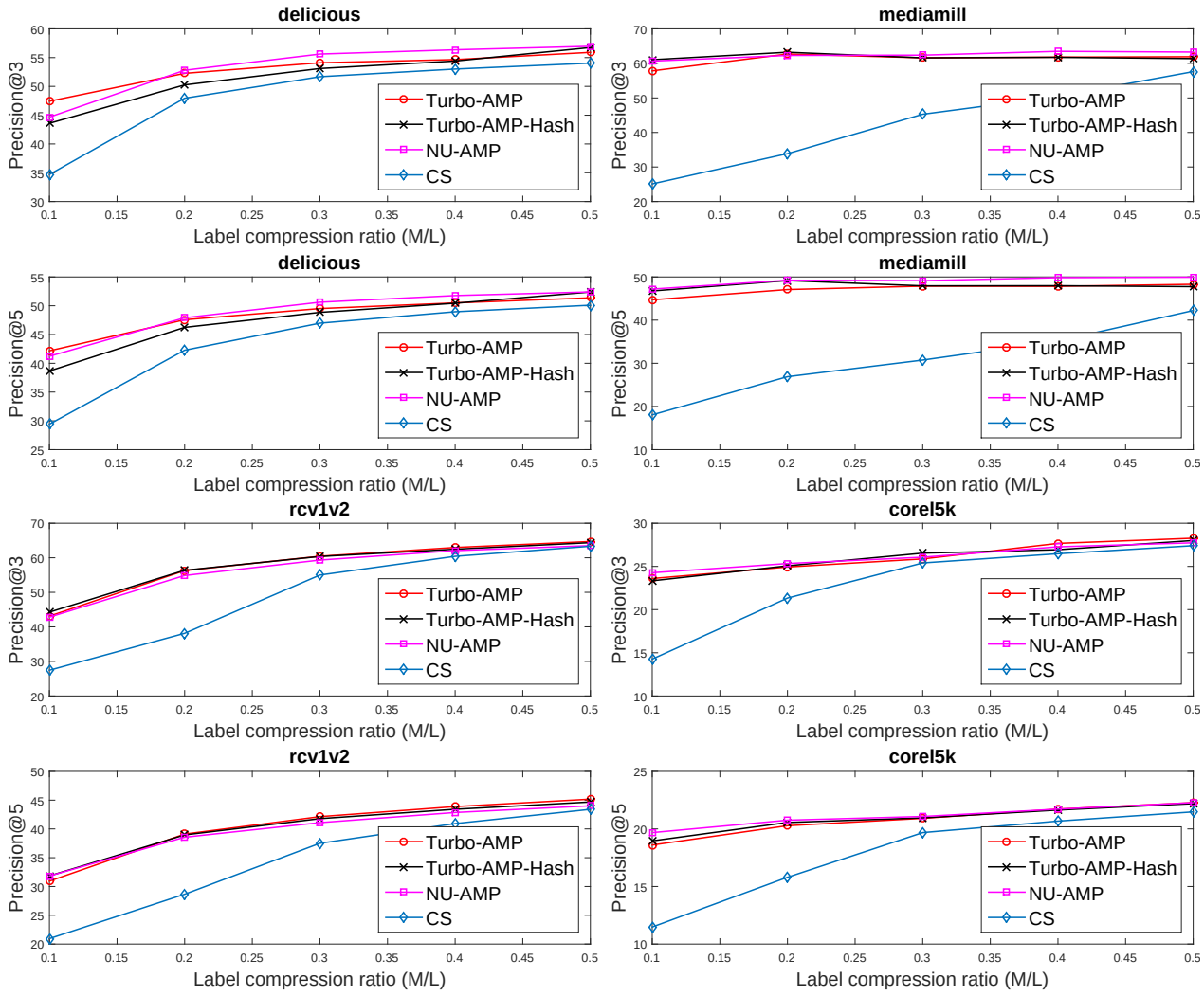

Fig. 5: Precision @3 and 5 vs label compression ratio (M/L).

TABLE I: Dataset statistics

Dataset name	Training instances	Feature dimension	Unique labels	Test instances	Label cardinality	Label density
delicious	12920	500	983	3185	19.02	0.019
mediamill	30993	120	101	12914	4.38	0.043
rcv1-v2	3000	47236	101	3000	2.88	0.029
corel5k	4500	499	374	500	3.52	0.009

our hashing based structure learning method, we get similar prediction accuracy as compared to the unconstrained structure learning.

We used the datasets[3] outlined in Table I for our experiments. Label cardinality is defined as the average number of active labels per instance. Label density, which represents the sparsity level of the label vector is defined as the average of the ratio of number of active labels and total number of labels. We report the results using two different metrics – accuracy and precision@k. Let T denote the set of actual active labels and P denote the set of predicted active labels. Accuracy [1], also known as Hamming score (please note that it is different from Hamming loss), is defined as $|T \cap P|/|T \cup P|$. Precision@k [17] is defined as the fraction of true positives among top k predicted active labels $|T \cap P_k|/k$, where P_k denotes the set of top k predicted active labels. Accuracy captures overall quality of the prediction whereas precision@k is more popular for applications like tag suggestion, keyword recommendation etc since only the top results are presented to the user in these applications [24].

We study classification performance as a function of label compression ratio (M/L). The purpose of these experiments is to demonstrate the effect of using structure in the CS recovery in multilabel prediction. We compare results from four experiments – (a) turbo AMP that uses structure learned in one step from the complete label vector (Turbo-AMP in figures), (b) turbo AMP that uses structure learned in two steps via label hashing (Turbo-AMP-Hash in figures) with B/L ratio of 1:10, (c), Non-uniform AMP that uses prior marginal distribution of the labels (NU-AMP in figures), and

[3]http://mulan.sourceforge.net/datasets-mlc.html

(d) plain CS using AMP with i.i.d. prior for $p(\mathbf{s})$ as the CS algorithm (CS in figures). The NU-AMP does not use the joint distribution $p(\mathbf{s})$ and uses the marginal distribution of s_l approximated from observed labels in the training dataset and assumes that the labels are independent. It doesn't use CLRG algorithm to learn the structure or joint distribution. In our experiments we use Gaussian projection matrix $\mathbf{A}$ and use linear regression for regressors h_m. We set $\sigma_s^2 = 1$ and estimate σ_w^2 from 5 fold cross-validation.

We report accuracy in Figure 4 and precision at $k = 3$ and $k = 5$ in Figure 5. We can see improvement of turbo AMP over unstructured plain CS recovery with AMP. The difference is more prominent at high compression (low M/L) which is helpful since our target is to achieve more compression. The improvements are more significant for the accuracy metric as compared to precision@k. The precision@k captures the quality of only the top k predicted labels. The accuracy metric captures both the precision and recall aspect of all labels (by penalizing both false detection and mis detection) and clearly structure learning helps in the overall quality of the predicted labels. From Figure 5, for precision@k, all the three methods compared to the plain CS algorithm, show improved performance but there is hardly any incentive to learn the structure (joint distribution $p(\mathbf{s})$) over using non-uniform independent prior (the marginal distribution of s_l). But the incentive is clear when we consider the more comprehensive metric of accuracy in Figure 4. We see that using the marginal distribution for s_l in NU-AMP helps to improve over plain CS algorithm but turbo-AMP does even better with structure learning. We also observe that even at $B/L = 1 : 10$, the structure learned via hashing trick achieves similar performance as compared to unconstrained label learning. We see that the higher the label cardinality (i.e., the average number of labels per sample) of a dataset the improvement from structure learning is more prominent. For example, for the delicious dataset, the average label cardinality is 19.02 and we see the benefit of structure learning across the entire compression range. The rcv1-v2 dataset has label cardinality of 2.88 and the benefit is less obvious. Higher cardinality typically leads to more structure. Thus we see the impact of structure learning more prominently.

VII. Conclusion

We demonstrated that exploiting label structure can improve the prediction accuracy for compressed sensing based approach to multilabel classification, especially at the high compression region. This opens up new research direction for efficient structure learning algorithms for multilabel scenario, which can potentially be domain and problem specific. In fact, our proposed hashing trick based structure learning algorithm is an example of such improvement. Although in this paper we didn't focus on applications where the label structure is known, these applications can also be benefited from the proposed turbo AMP algorithm in multilabel setting.

References

[1] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, pp. 333–359, 2011.

[2] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data mining and knowledge discovery handbook*, pp. 667–685, Springer, 2010.

[3] D. Hsu, S. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing.," in *Advances in Neural Information Processing Systems*, vol. 22, pp. 772–780, 2009.

[4] F. Tai and H.-T. Lin, "Multilabel classification with principal label space transformation," *Neural Computation*, vol. 24, no. 9, pp. 2508–2542, 2012.

[5] M. M. Cisse, N. Usunier, T. Artieres, and P. Gallinari, "Robust Bloom filters for large multilabel classification tasks," in *Advances in Neural Information Processing Systems*, pp. 1851–1859, 2013.

[6] W. Bi and J. T. Kwok, "Multi-label classification on tree-and dag-structured hierarchies," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 17–24, 2011.

[7] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

[8] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *The Journal of Machine Learning Research*, vol. 7, pp. 1601–1626, 2006.

[9] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 999–1008, ACM, 2010.

[10] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.

[11] S. Som, L. C. Potter, and P. Schniter, "On approximate message passing for reconstruction of non-uniformly sparse signals," in *National Aerospace and Electronics Conference (NAECON), Proceedings of the IEEE*, pp. 223–229, IEEE, 2010.

[12] M. J. Choi, V. Y. Tan, A. Anandkumar, and A. S. Willsky, "Learning latent tree graphical models," *The Journal of Machine Learning Research*, vol. 12, pp. 1771–1812, 2011.

[13] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, pp. 1–6, IEEE, 2010.

[14] S. Som and P. Schniter, "Compressive imaging using approximate message passing and a Markov-tree prior," *Signal Processing, IEEE Transactions on*, vol. 60, no. 7, pp. 3439–3448, 2012.

[15] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[16] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[17] A. Kapoor, R. Viswanathan, and P. Jain, "Multilabel classification using Bayesian compressed sensing," in *Advances in Neural Information Processing Systems*, pp. 2645–2653, 2012.

[18] D. Vasisht, A. Damianou, M. Varma, and A. Kapoor, "Active learning for sparse Bayesian multilabel classification," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 472–481, ACM, 2014.

[19] B. J. Frey and D. J. MacKay, "A revolution: Belief propagation in graphs with cycles," *Advances in neural information processing systems*, pp. 479–485, 1998.

[20] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *Information Theory, IEEE Transactions on*, vol. 57, no. 2, pp. 764–785, 2011.

[21] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *Information Theory, IEEE Transactions on*, vol. 14, no. 3, pp. 462–467, 1968.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[23] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1113–1120, ACM, 2009.

[24] 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*, pp. 13–24, 2013.