Supplementary Material for Conditional Bernoulli Mixtures for Multi-label Classification

A Several Implementation and Experiment Details

A.1 Dealing with Empty Predictions

Predicting empty label subsets could be undesirable when it is known a priori that each instance matches at least one label. The dynamic programming prediction algorithm in the paper can be easily modified to output the most probably non-empty subsets. In our experiments, we allow CBM to predict empty sets only when the training set contains empty sets. This strategy is shown to improve the test performance slightly. Occasionally, this could make CBM with 1 component perform slightly differently from BinRel (see Figure 3 in the paper).

A.2 Hyper Parameter Tuning

In our experiments, we do cross-validation on the training set to tune the following hyper parameters:

- LR Gaussian prior variance V_{LR}, on grids $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}\}$;
- CRF Gaussian prior variance V_{CRF}, on grids $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}\}$;
- CBM+LR number of components K, on grids $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60\}$;
- CBM+GB number of components K, on grids $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60\}$.

Tuning results are shown in Table 1.

Table 1: Tuned Hyper Parameters

dataset	V_{LR}	V_{CRF}	K (CBM+LR)	K (CBM+GB)
SCENE	1.0	1.0	20	25
RCV1	10^{6}	1.0	45	45
TMC2007	10^{-1}	10^{-1}	40	20
MEDIAMILL	10^{3}	1	50	5
NUS-WIDE	1.0	1.0	50	10

Our gradient boosting implementation uses regression trees of 5 leaves and shrinkage rate 0.1 as default values. For PCC, the beam search width is set to be 15, as suggested in [3].

B Results for Jaccard Index and Hamming Loss

Let $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ be a multi-label dataset with ground truth labels, and $\{\hat{\mathbf{y}}_n\}_{n=1}^{N}$ be the predictions made by a classifier. The Jaccard index is defined as

$$\frac{1}{N}\sum_{n=1}^{N}\frac{|\mathbf{y}_n \cap \hat{\mathbf{y}}_n|}{|\mathbf{y}_n \cup \hat{\mathbf{y}}_n|}, \tag{1}$$

where $\mathbf{y}_n$ and $\hat{\mathbf{y}}_n$ are interpreted as sets. Jaccard index is a well-behaved evaluation measure, often more practical than subset accuracy, because it assigns partial credits to "almost correct" answers and handles label imbalance well. To the best of our knowledge, the optimal classifier form for Jaccard index as defined in (1) is unknown. The authors

in [2] make a distinction between EUM and DTA multilabel population utility. The utility defined in (1) is refereed to as the DTA utility. For Jaccard index defined as the EUM utility, the optimal classifier can be decomposed into binary classifiers $h^*_\ell(\mathbf{x}) = \mathbb{1}[p(y_\ell = 1|\mathbf{x}) > \eta]$, where η is a common threshold shared by all labels $\ell \in \mathcal{Y}$. The interested readers can refer to [2] for more details.

Hamming loss measures bit-wise errors and is defined as

$$\frac{1}{NL}\sum_{n=1}^{N}\sum_{\ell=1}^{L}\mathbb{1}[y_{n\ell} \neq \hat{y}_{n\ell}]. \tag{2}$$

Hamming loss is less discriminative than subset accuracy and Jaccard index because it ignores label imbalance. In practice, each instance usually matches only a few labels out of many candidates. A trivial classifier predicting empty set could also get a decent Hamming loss. According to [1], the optimal classifier for Hamming loss simply predicts each label independently based on its marginal probability

$$h^*_\ell(\mathbf{x}) = \arg\max_{y_\ell} p(y_\ell|\mathbf{x}). \tag{3}$$

The theoretical analysis in [1] also shows that a multi-label classifier designed for optimizing one measure may be suboptimal under other measures. Testing performance based on Jaccard index and Hamming loss is shown in Table 2. BinRel achieves the best Hamming loss on all datasets, as expected. CBM achieves the highest Jaccard index on three out of five datasets, which is reasonable given the fact that predicting the joint mode as currently done in CBM is only optimal for subset accuracy and multi-label optimality is measure related. It should be interesting to consider how to design prediction methods based on the joint distribution estimated by CBM in order to optimize for other measures. We leave this for future work.

Table 2: The testing performance of different methods on five datasets. All numbers are in percentages. Best performances are bolded.

dataset		SCENE (image)		RCV1 (text)		TMC2007 (text)		MEDIAMILL (video)		NUS-WIDE (image)	
#labels / #label subsets		6 /	15	103 /	799	22 /	1341	101 /	6555	81 /	18K
#features / #data points		294 /	2407	47K /	6000	49K /	29K	120 /	44K	128 /	270K
Method	Learner	Jaccard	Hamming	Jaccard	Hamming	Jaccard	Hamming	Jaccard	Hamming	Jaccard	Hamming
BinRel	LR	58.4	10.8	64.9	**1.4**	52.2	**6.5**	42.3	3.1	32.3	**2.1**
PowSet	LR	71.9	9.5	67.2	1.9	51.9	6.8	35.2	3.6	32.3	**2.1**
CC	LR	67.5	10.9	63.4	1.6	52.4	**6.5**	40.5	3.2	32.2	**2.1**
PCC	LR	69.4	10.3	63.5	1.6	52.8	**6.5**	38.0	3.5	32.1	2.2
ECC-label	LR	65.6	10.1	64.8	**1.4**	52.1	**6.5**	41.1	3.1	32.3	**2.1**
ECC-subset	LR	68.0	10.3	67.4	**1.4**	52.2	**6.5**	41.1	3.2	32.3	**2.1**
CDN	LR	66.4	10.9	50.9	2.8	44.0	8.5	38.6	4.2	32.8	2.7
pairCRF	linear	72.8	9.2	64.6	1.7	**53.4**	**6.5**	35.9	3.5	32.7	2.2
CBM	LR	73.6	8.9	**69.5**	**1.4**	53.1	6.7	41.2	3.4	**34.2**	**2.1**
BinRel	GB	63.9	**8.3**	55.8	1.7	52.2	6.7	**44.2**	**3.0**	30.6	**2.1**
PowSet	GB	74.9	8.6	51.3	2.9	42.7	9.0	38.5	3.9	24.3	2.4
CBM	GB	**75.2**	8.5	62.5	1.8	52.8	6.8	43.0	3.2	31.6	2.2

C Running Time

To facilitate wider comparison of different methods, we also report their training time and prediction time measured in our experiments. We use linear learners for all algorithms to make results comparable. Our Java implementations of BinRel, PowSet, CC, PCC, ECC, pairCRF and CBM are all multi-threaded.[1] The CDN implementation is taken from the MEKA package[2] without further modification. Each experiment is conducted on a computer with Intel Xeon CPU E5-2690 v3 2.6GHz, 48 logical cores and 128GB of RAM. The timing results are in Table 3. All numbers are in seconds and the predict time measures the time required to make predictions on the entire test set. We can see in

[1] Our implementations of CBM and several baselines (PowSet, PCC, CRF, etc.) are available at `https://github.com/cheng-li/pyramid`.

[2] `http://meka.sourceforge.net`

terms of speed, BinRel is the clear winner. However, its poor accuracy makes its advantage in speed less attractive. On datasets with a large number of possible subsets, CBM takes significantly less training time compared with PowSet, while still achieving better accuracy. Currently, there are two main factors that affect the training time of CBM: the number of clusters and the number of EM iterations. Our current prototype implementation of CBM mainly serves to demonstrate the concept. Designing better regularization and training techniques could possibly further reduce those two number and speed up CBM training, and we leave it for future work.

Table 3: The training time and prediction time of different methods on five datasets. All numbers are in seconds.

dataset		SCENE (image)		RCV1 (text)		TMC2007 (text)		MEDIAMILL (video)		NUS-WIDE (image)	
#labels / #label subsets		6 /	15	103 /	799	22 /	1341	101 /	6555	81 /	18K
#features / #data points		294 /	2407	47K /	6000	49K /	29K	120 /	44K	128 /	270K
Method	Learner	Train	Predict	Train	Predict	Train	Predict	Train	Predict	Train	Predict
BinRel	LR	2	<1	19	<1	26	<1	136	<1	128	1
PowSet	LR	35	<1	3147	<1	38037	1	85794	1	521760	34
CC	LR	3	<1	509	<1	332	<1	1949	1	2520	2
PCC	LR	3	<1	509	3	332	1	1949	4	2520	27
ECC-label	LR	22	<1	4915	27	3404	15	19642	38	25791	246
ECC-subset	LR	22	<1	4915	26	3404	18	19642	39	25791	287
CDN	LR	4	45	18417	213433	54253	596228	3126	6572	17941	41789
pairCRF	linear	11	<1	2136	<1	215	<1	2990	<1	48404	7
CBM	LR	70	<1	4412	4	1495	1	17608	13	35363	48

References

[1] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

[2] Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent multilabel classification. In *Advances in Neural Information Processing Systems*, pages 3303–3311, 2015.

[3] Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. Beam search algorithms for multilabel learning. *Machine learning*, 92(1):65–89, 2013.