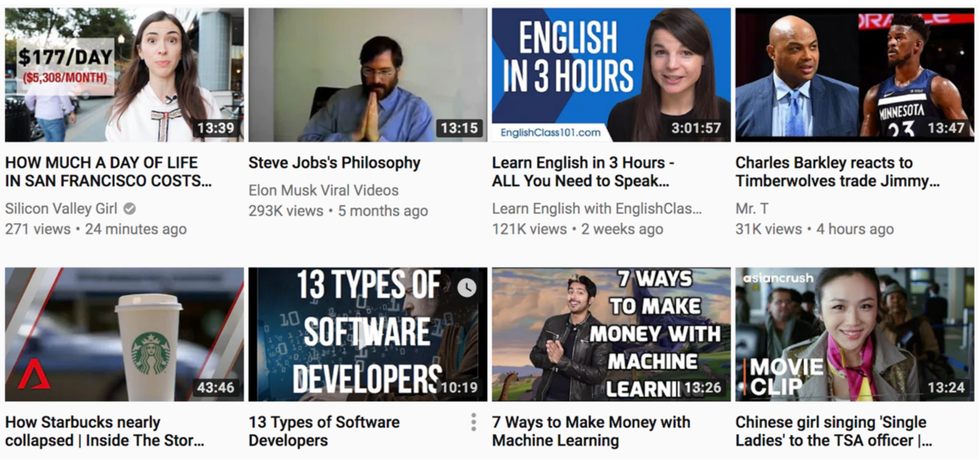
Learning Compact Model for Large-scale Multi-label Data

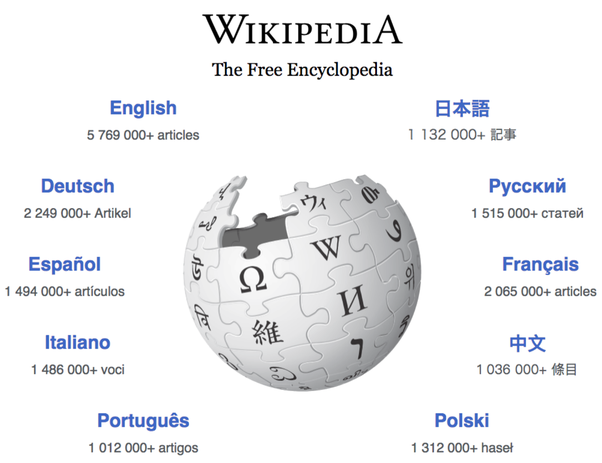
Tong Wei, Yu-Feng Li {weit, liyf}@lamda.nju.edu.cn
Department of Computer Science & Technology, Nanjing University, China

LAMDA
Learning And Mining from DatA
http://lamda.nju.edu.cn

1. Multi-Label Learning



Recommender Systems



Document Categorization

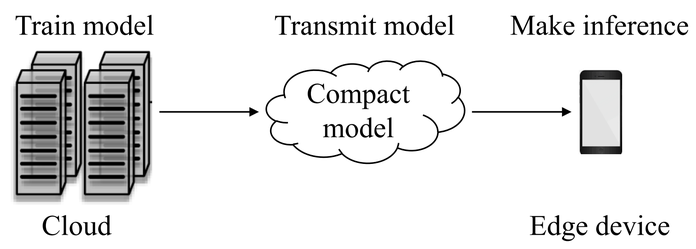
Image Tagging

2. Motivation

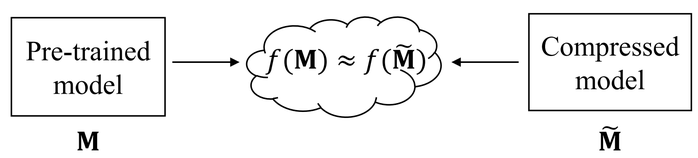
In real-world applications, data sets with **tens of thousands** of labels and features are already becoming benchmark in many domains.

Consequently, many effective approaches *[Tsoumakas et al., DMKD 2009; Zhang and Zhou, TKDE 2014]* are hard to deal with LMLL (Large-scale Multi-Label Learning) data due to large storage overhead.

A popular walk around



Challenges (retain competitive performance while doing compression)



Related work

- *[Babbar and Schölkopf, WSDM'17]* filters out spurious features parameters to reduce model size.
- *[Boutsidis et al., SODA'09; Bi and Kwok, ICML'13]* selects only a small subset of labels that can approximately span the original label space and subsequently model size is reduced.

3. Methodology

Problem Formulation

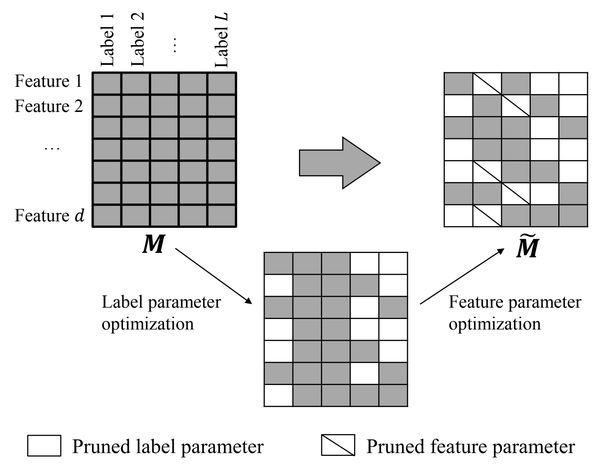
Given a pre-trained model $\mathbf{M}$, the goal is to find a compact model $\tilde{\mathbf{M}}$ with comparable performance. Such objective can be formulated as:

$$\min_{\tilde{\mathbf{M}}} \ size(\tilde{\mathbf{M}}) \quad \text{s.t.} \ f(\tilde{\mathbf{M}}, \mathcal{D}) \geq q^* - \epsilon$$

In this work, we focus on linear models, and the optimization problem can be rewritten as:

$$\min_{\tilde{\mathbf{M}}} \ ||\tilde{\mathbf{M}}||_0 \quad \text{s.t.} \ perf(\mathbf{X}\tilde{\mathbf{M}}, \mathbf{Y}) \geq q^* - \epsilon$$

Joint Label and Feature Parameter Optimization (POP)



Since the resultant optimization problem is difficult, we propose to solve it from label and feature parameter optimization aspects jointly.

Parameter Optimization w.r.t Label

$$\min_{\tilde{\mathbf{M}}} \ \sum_{j=1}^{L} \Big\| ||\tilde{\mathbf{M}}_{:,j}||_0 - \delta \Big\|_0 \quad \text{s.t.} \ perf(\mathbf{X}\tilde{\mathbf{M}}, \mathbf{Y}) \geq q^* - \epsilon, \ ||\tilde{\mathbf{M}}_{:,j}||_0 \geq \delta, \ j = 1, \ldots, L$$

We compute the impact of labels for commonly used LMLL metrics (PSP@k and PSnDCG@k). Since missing labels commonly occur in LMLL *[Bi and Kwok 2013; Xu, et. al., 2016]*, we capture the impact of labels under this scenario.

Theorem 1. *Suppose that relevant labels are randomly missing with probability π, the impact of the j-th label in terms of PSP@k and PSnDCG@k is upper bounded by $(1-\pi)w_j u_j$.*

where u_j is # of relevant instances for label j and w_j is the weight label j.

Parameter Optimization w.r.t Feature

We further locate the most important feature parameters for the influential labels.

$$\min_{\tilde{\mathbf{M}}} \ ||\tilde{\mathbf{Y}} - \mathbf{Y}^*||_F^2 + \lambda ||\tilde{\mathbf{M}}||_0 \quad \text{s.t.} \ \tilde{\mathbf{Y}} = \mathbf{X}\tilde{\mathbf{M}}; \mathbf{Y}^* = \mathbf{X}\mathbf{M}$$

Inspired by *[Zhao and Yu, JMLR'06]*, an approximate solution can be obtained by setting feature parameters that lie in range $[-\lambda, \lambda]$ to 0.

4. Empirical Studies

Comparison with baseline (BR)

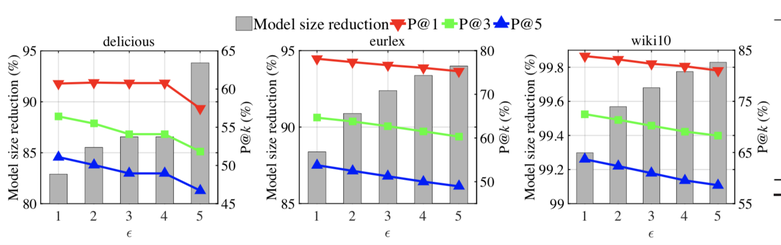
- Avg. model size reduction > 50%.
- Avg. performance loss < 0.5%.

Comparison with State-of-the-arts

- POP achieves top 2 results in 23/28 cases.
- Vs. DiSMEC: 10x size reduction on wiki10 and 4x reduction on eurlex.
- Vs. SLEEC: avg. 9x size reduction.
- Vs. FastXML: avg: 10x size reduction.
- Vs. LEML/CoH/PD-Sparse: POP consistently outperformances.

Parameter study

- POP filters out more than 80% model parameters when $\epsilon = 1$.
- Predictive accuracy goes down very slowly as ϵ becomes bigger.



Data set		PSP@1	PSP@3	PSP@5	PSnDCG@1	PSnDCG@3	PSnDCG@5	Model size
bibtex	BR	50.70	53.66	59.34	50.70	52.71	55.80	1.15 MB
	POP	50.71	53.30	58.86	50.71	52.39	55.41	**0.59 MB**
delicious	BR	32.14	33.59	33.43	32.14	33.32	33.28	7.18 MB
	POP	32.08	33.59	33.47	32.08	33.30	33.29	**1.26 MB**
eurlex	BR	39.93	45.86	49.74	39.93	44.24	46.83	156.38 MB
	POP	40.06	46.02	49.91	40.06	44.42	47.01	**20.18 MB**
wiki10	BR	13.57	13.10	13.96	13.60	13.82	13.97	23.50 GB
	POP	13.53	13.10	13.46	13.53	13.65	13.67	**67.50 MB**

Data set		FastXML	LEML	SLEEC	CoH	DiSMEC	PD-Sparse	POP (ours)
bibtex	Model size	18.72 MB	0.76 MB	2.46 MB	4.70 MB	0.71 MB	**20.00 KB**	**0.59 MB**
	PSP@1	48.54	47.97	**51.12**	36.53	50.20	48.34	**50.71**
	PSP@3	52.30	51.42	**53.95**	28.20	52.20	48.77	**53.30**
	PSP@5	58.28	57.53	**59.56**	25.59	58.60	52.93	**58.86**
	PSnDCG@1	48.54	47.97	**51.12**	36.58	50.20	48.34	**50.71**
	PSnDCG@3	51.11	50.25	**52.99**	30.62	52.00	48.49	**52.39**
	PSnDCG@5	54.38	53.59	**56.04**	29.06	**55.70**	50.72	55.41
delicious	Model size	71.29 MB	2.26 MB	7.34 MB	10.92 MB	-	**0.25 MB**	**1.26 MB**
	PSP@1	**32.35**	30.73	**32.11**	20.43	-	25.22	32.08
	PSP@3	**34.51**	32.43	33.21	22.76	-	24.63	**33.59**
	PSP@5	**35.43**	33.26	**33.83**	24.11	-	23.85	33.47
	PSnDCG@1	**32.35**	30.73	**32.11**	20.43	-	25.22	32.08
	PSnDCG@3	**34.00**	32.01	32.93	22.16	-	24.80	**33.30**
	PSnDCG@5	**34.73**	32.66	**33.41**	23.20	-	24.25	33.29
eurlex	Model size	194.40 MB	34.31 MB	245.49 MB	**15.95 MB**	79.86 MB	25.00 MB	**20.18 MB**
	PSP@1	26.62	24.10	34.25	20.78	**41.20**	38.28	**40.06**
	PSP@3	34.16	27.20	39.83	22.98	**45.40**	42.00	**46.02**
	PSP@5	38.96	29.09	42.76	21.89	**49.30**	44.89	**49.91**
	PSnDCG@1	26.62	24.10	34.25	20.78	**41.20**	38.28	**40.06**
	PSnDCG@3	32.07	26.37	38.35	22.49	**44.30**	40.96	**43.55**
	PSnDCG@5	35.23	27.62	40.30	21.92	**46.90**	42.84	**47.01**
wiki10	Model size	**501.47 MB**	506.88 MB	924.60 MB	-	880.00 MB	-	**67.50 MB**
	PSP@1	9.80	9.41	11.14	-	**13.60**	-	**13.53**
	PSP@3	10.17	10.07	11.86	-	**13.10**	-	**13.10**
	PSP@5	10.54	10.55	12.40	-	**13.80**	-	**13.46**
	PSnDCG@1	9.80	9.41	11.14	-	**13.60**	-	**13.53**
	PSnDCG@3	10.08	9.90	11.68	-	**13.20**	-	**13.65**
	PSnDCG@5	10.33	10.24	12.06	-	**13.60**	-	**13.67**

The best and the second best results are in bold.

5. Take-home Messages

Main contribution

- The impact of labels on PSP@k and PSnDCG@k is related to the label weights and label frequencies.
- We propose the POP method to compress the model size by jointly performing label and feature parameter optimization.

Feel free to check out the code