PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks

Authors

Jian Tang
Microsoft Research Asia
jiatang@microsoft.com

Meng Qu
Peking University
mnqu@pku.edu.cn

Qiaozhu Mei
University of Michigan
qmei@umich.edu

Presenter: ***Feven Tsegaye Aga***

Overview

- Semi-supervised representation learning method for text data has proposed
- The method called ***predictive text embedding (PTE)***
- PTE utilizes both labeled and unlabeled data to learn the embedding of text.
  1. The labeled and unlabeled information are represented as a large-scale heterogeneous text network.
  2. The information embedded into a low dimensional space

Introduction

- Unsupervised text embeddings are generalizable for different tasks
- Comparing with deep learning approaches, the performance of text embeddings usually falls short on specific tasks
- PTE fills this gap
- Experiments have conducted on both long and short documents.

➢ PTE outperforms on long and comparable results on short documents.

Related Work

- Unsupervised text embedding
  - CBOW (Mikolov et al, 2013)
  - Skip-gram (Mikolov et al,2013)
  - Paragraph vector (Le et al,2014)

Cons

  - Not tuned for specific tasks

Pros

  - Simple model scalable
  - Leverage a large amount of unlabeled data, embeddings are general for different tasks
  - Insensitive parameters

- Supervised text embedding
  - Recurrent neural network (Mikolov et al, 2013)
  - Recursive neural network (Socher et al,2012)
  - Convolution neural network (Kim et al,2014)

Cons

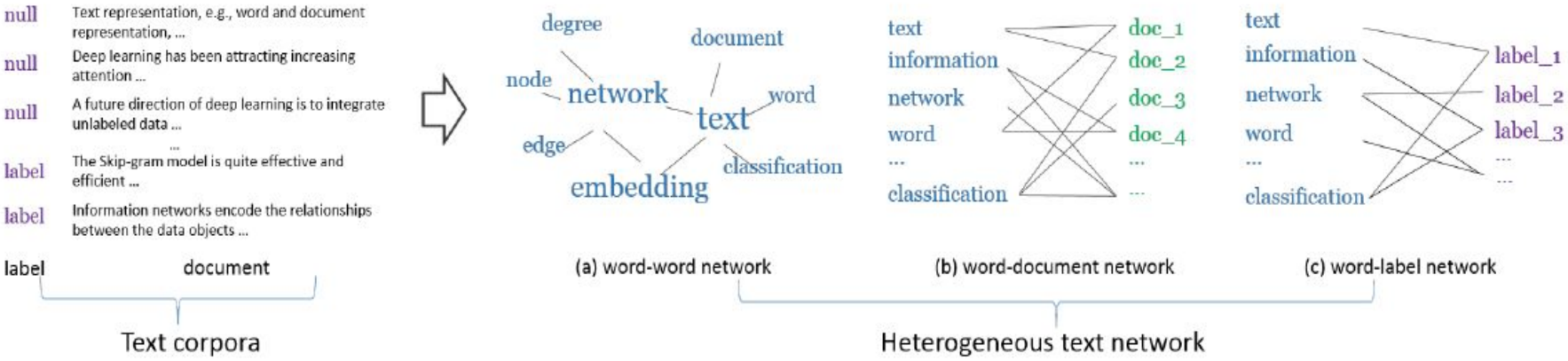
  - Computational expensive
  - Require a large number of labeled data, hard to leverage unlabeled data
  - Very sensitive parameters, difficult to tune

Pros

  - State of the art performance on specific task

Predictive Text Embedding

- adapts the advantages of unsupervised text embeddings but naturally utilizes the labeled data for specific task
- text network uniformly represent both unsupervised and supervised data
- different levels of word co-occurrences are: local context-level, label-level and document-level



Bipartite Network Embedding

First adapt the LINE model for embedding bipartite networks. The essential idea is to make use of the second-order proximity

For each edge (vi,vj), define a conditional probability

$$p(v_i|v_j) = \frac{\exp(\vec{u}_i^T \cdot \vec{u}_j)}{\sum_{i' \in A} \exp(\vec{u}_{i'}^T \cdot \vec{u}_j)}$$

Objective,

$$O = -\sum_{(i,j) \in E} w_{ij} \log p(v_j|v_i)$$

Heterogeneous Text Network Embedding

The heterogeneous text network is composed of three bipartite networks:

- word-word, word-document and word-label networks

Objective,

$$O_{pte} = O_{ww} + O_{wd} + O_{wl},$$

where

$$O_{ww} = -\sum_{(i,j)\in E_{ww}} w_{ij} \log p(v_i|v_j) \quad \text{(Word-word network)}$$

$$O_{wd} = -\sum_{(i,j)\in E_{wd}} w_{ij} \log p(v_i|d_j) \quad \text{(Word-document network)}$$

$$O_{wl} = -\sum_{(i,j)\in E_{wl}} w_{ij} \log p(v_i|l_j) \quad \text{(Word-label network)}$$

Optimization

Two different ways of optimization

- Joint training: jointly train the three networks
- Pre-training and fine-tuning: jointly train the word-word and word-document networks
  : fine tuning the word embedding with the word-label network

Algorithm 1: Joint training.

Data: G_{ww}, G_{wd}, G_{wl}, number of samples T, number of negative samples K.
Result: word embeddings $\vec{w}$.
while $iter \leq T$ **do**
- sample an edge from E_{ww} and draw K negative edges, and update the word embeddings;
- sample an edge from E_{wd} and draw K negative edges, and update the word and document embeddings;
- sample an edge from E_{wl} and draw K negative edges, and update the word and label embeddings;

end

Algorithm 2: Pre-training + Fine-tuning.

Data: G_{ww}, G_{wd}, G_{wl}, number of samples T, number of negative samples K.
Result: word embeddings $\vec{w}$.
while $iter \leq T$ **do**
- sample an edge from E_{ww} and draw K negative edges, and update the word embeddings;
- sample an edge from E_{wd} and draw K negative edges, and update the word and document embeddings;

end
while $iter \leq T$ **do**
- sample an edge from E_{wl} and draw K negative edges, and update the word and label embeddings;

end

Text Embedding

- The heterogeneous text network encodes word co-occurrences at different levels, extracted from both unlabeled data and labeled information for a specific classification task.
- Therefore, the word representations are robust and optimized for the specified task.
- The representation of an arbitrary piece of text is obtained by averaging the vectors of the words in the piece of text.
- That is. the vector representation of a piece of text ***d = w1,w2,….,wn*** can be computed as

$$\vec{d} = \frac{1}{n}\sum_{i=1}^{n}\vec{u}_i, \quad \text{where } \vec{u}_i \text{ is the embedding of word } w_i.$$

- The average of the word embeddings is the solution to minimizing the following objective function:

$$O = \sum_{i=1}^{n} l(\vec{u}_i, \vec{d}),$$

Experiment Setup

Compared algorithms

- BOW: classical “bag-of-words”representation
- Unsupervised text embedding
  - Skip-gram: local content level word co-occurrence
  - Paragraph vector: document level word co-occurrences
  - LINE: to learn unsupervised text embedding by combining word-word network and word-document network
- Supervised text embedding
  - CNN: proposed for modeling sentences

Text classification

- Embeddings as features
- Classifier: logistic regression

Quantitative Results

- Performance on Long Documents

Table 2: Results of text classification on **long** documents.

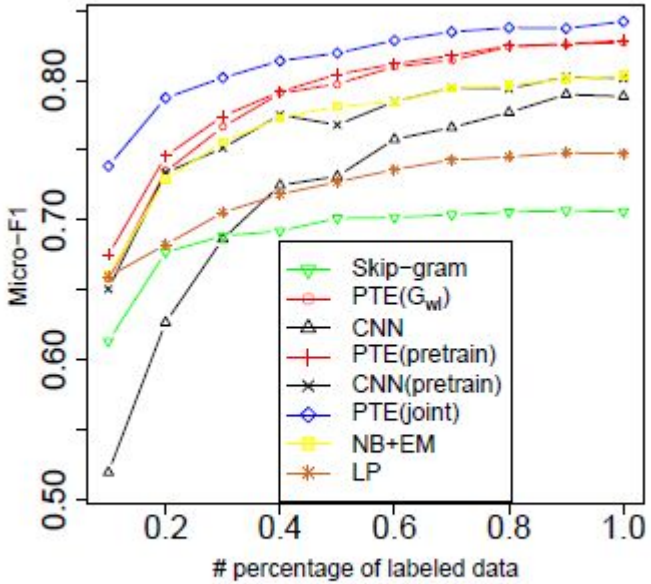
		20NG		Wikipedia		IMDB	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Word	BOW	80.88	79.30	79.95	80.03	86.54	86.54
Unsupervised Embedding	Skip-gram	70.62	68.99	75.80	75.77	85.34	85.34
	PVDBOW	75.13	73.48	76.68	76.75	86.76	86.76
	PVDM	61.03	56.46	72.96	72.76	82.33	82.33
	LINE(G_{ww})	72.78	70.95	77.72	77.72	86.16	86.16
	LINE(G_{wd})	79.73	78.40	80.14	80.13	89.14	89.14
	LINE($G_{ww} + G_{wd}$)	78.74	77.39	79.91	79.94	89.07	89.07
Predictive Embedding	CNN	78.85	78.29	79.72	79.77	86.15	86.15
	CNN(pretrain)	80.15	79.43	79.25	79.32	89.00	89.00
	PTE(G_{wl})	82.70	81.97	79.00	79.02	85.98	85.98
	PTE($G_{ww} + G_{wl}$)	83.90	83.11	81.65	81.62	89.14	89.14
	PTE($G_{wd} + G_{wl}$)	**84.39**	**83.64**	82.29	82.27	89.76	89.76
	PTE(pretrain)	82.86	82.12	79.18	79.21	86.28	86.28
	PTE(joint)	84.20	83.39	**82.51**	**82.49**	**89.80**	**89.80**

- Performance on short documents

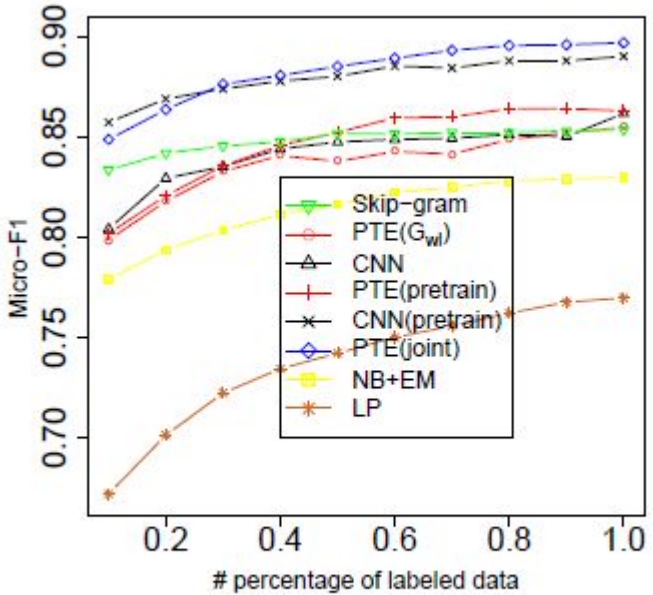
Table 4: Results of text classification on **short** documents.

		DBLP		MR		Twitter	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Word	BOW	75.28	71.59	71.90	71.90	75.27	75.27
Unsupervised Embedding	Skip-gram	73.08	68.92	67.05	67.05	73.02	73.00
	PVDBOW	67.19	62.46	67.78	67.78	71.29	71.18
	PVDM	37.11	34.38	58.22	58.17	70.75	70.73
	LINE(G_{ww})	73.98	69.92	71.07	71.06	73.19	73.18
	LINE(G_{wd})	71.50	67.23	69.25	69.24	73.19	73.19
	LINE($G_{ww} + G_{wd}$)	74.22	70.12	71.13	71.12	73.84	73.84
Predictive Embedding	CNN	76.16	73.08	72.71	72.69	**75.97**	**75.96**
	CNN(pretrain)	75.39	72.28	68.96	68.87	75.92	75.92
	PTE(G_{wl})	76.45	72.74	73.44	73.42	73.92	73.91
	PTE($G_{ww} + G_{wl}$)	76.80	73.28	72.93	72.92	74.93	74.92
	PTE($G_{wd} + G_{wl}$)	**77.46**	**74.03**	73.13	73.11	75.61	75.61
	PTE(pretrain)	76.53	72.94	73.27	73.24	73.79	73.79
	PTE(joint)	77.15	73.61	**73.58**	**73.57**	75.21	75.21

Performance on w.r.t labeled data



(a) 20NG



(b) IMDB

Conclusion and Future works

- Introduced Predictive Text Embedding
- Adapts the advantages of unsupervised text embeddings but naturally utilizes labeled information in representation learning.
- Encode unsupervised and supervised information through *Word-word, word-document, word-label* large scale heterogeneous text networks
- PTE outperforms on long documents and generates comparable results to CNN on short documents.
- PTE is much faster and much easier to configure with few parameters

Future works

- A way to fix is to adjust the sampling probability from the word-label and word-word/word-document when the labeled data is scarce.
- To improve the predictive text embedding method by utilizing word orders.

References

- M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In NIPS, volume 14, pages 585{591, 2001.
- Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(8):1798{1828, 2013.
- S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In Social network data analytics, pages 115{148. Springer, 2011.
- D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. JMLR, 3:993{1022, 2003.
- P. Blunsom, E. Grefenstette, N. Kalchbrenner, et al. A convolutional neural network for modelling sentences. In ACL, 2014.
- R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. JMLR, 12:2493{2537, 2011.
- J. R. Firth. A synopsis of linguistic theory, 1930{1955. In J.R. Firth (Ed.), Studies in linguistic analysis, pages 1-32.
- Y. Kim. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.
- J. B. Kruskal and M. Wish. Multidimensional scaling, volume 11. Sage, 1978.
- Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053, 2014.
- Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361:310, 1995.
- D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. JMLR, 5:361{397, 2004.
- D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. Journal of the American society for information science and technology,58(7):1019{1031, 2007.
- Y. Liu, Z. Liu, T.-S. Chua, and M. Sun. Topical word embeddings. 2015.
- A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In ACL-HLT, pages 142{150, 2011.

University of Hildesheim

Data analytics 1

FastXML: A Fast, Accurate and Stable tree-classifier for eXtreme Multi-label Learning

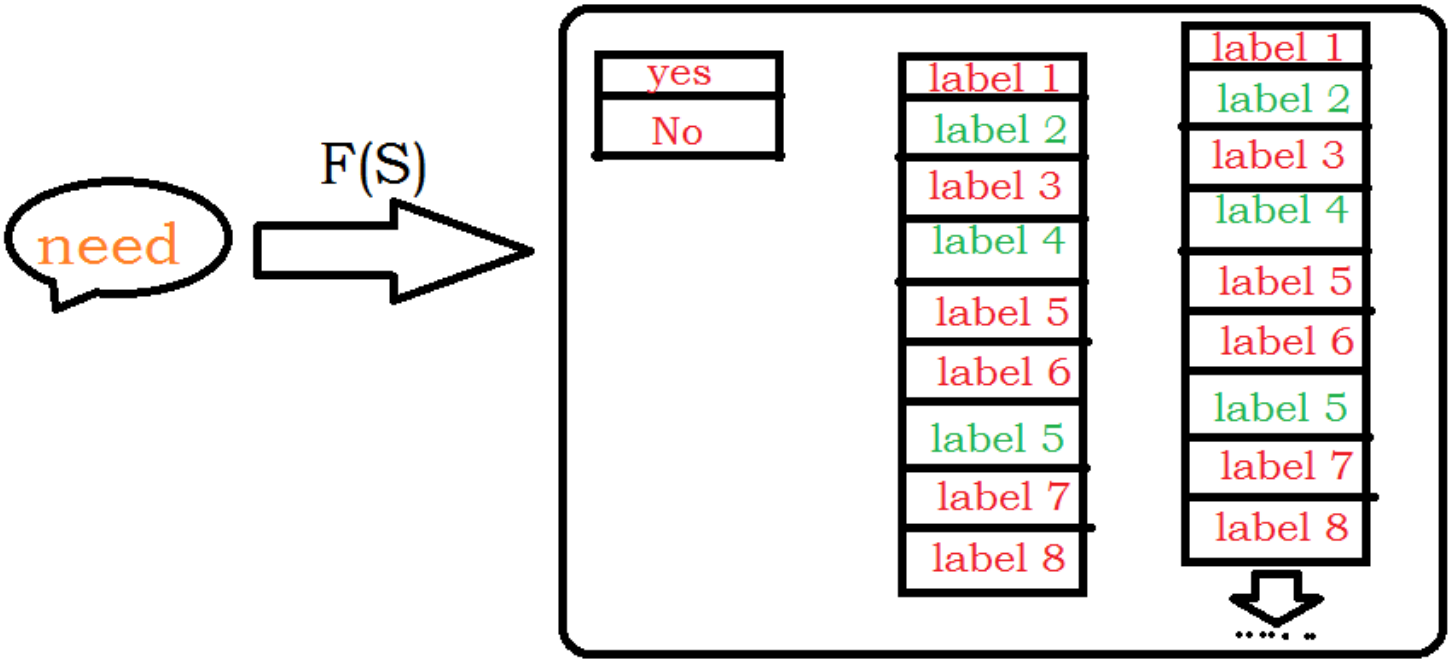
Presented by :
Mohamed Ikbal NACER

Work-plan

1. Introduction
2. Bedrock work
3. FastXML
4. Ensemble of balanced Tree classifier
5. Modes partitioned using nDCG
6. Alternating minimization based optimization
7. Comparison and performance
8. Conclusion
9. References

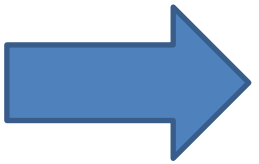
Introduction

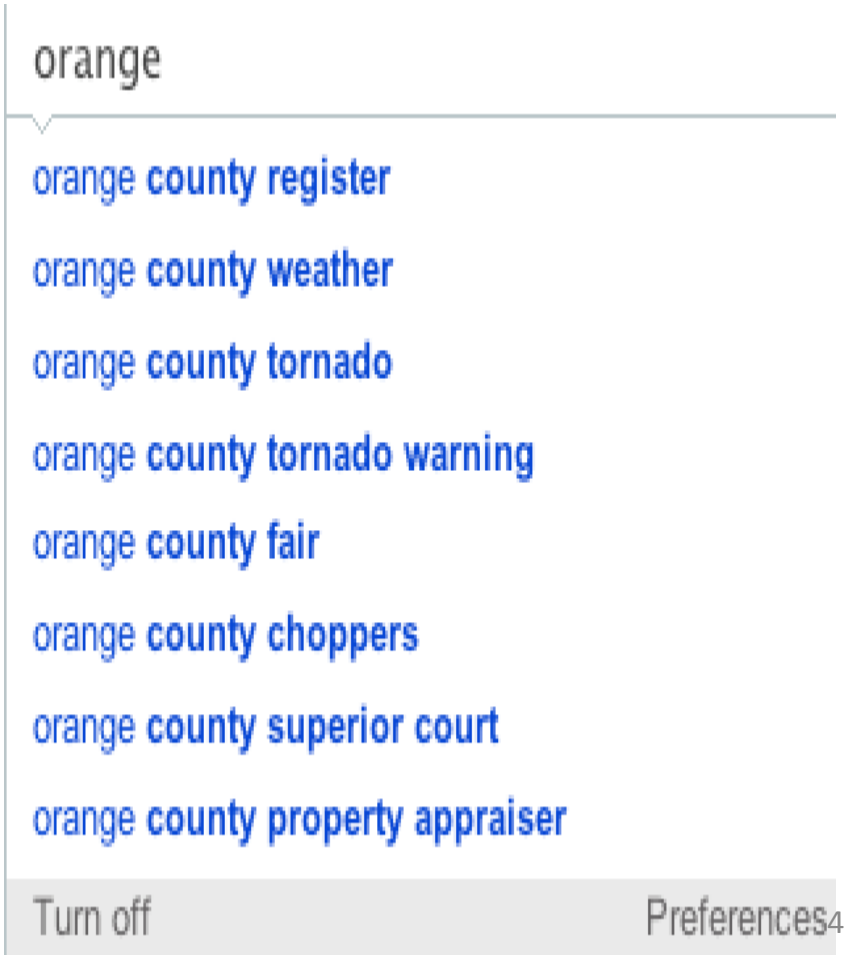
- The problem grows from a choice between two labels (binary) to a choice between a lot of labels (n_labels).
- Today due to business and academic needs we must be able to tag our data point to a large number of labels



Introduction

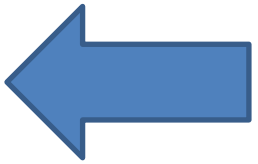
This is what we are aiming to do ：





Introduction

Why we do not consider it this way ?
And we end up simply in a binary solution ?



Introduction

- The paper is written by Manik Varma from Microsoft research Indian collaborating with his PhD student Yashoteja Prabhu at the university of Delhi
- It is an optimization of previous work by Varma him self
- This work has been optimized later with collaboration with other PhD students

Introduction

- The objective in this paper is to learn classifier that can tag data point with the most relevant subset
- Extreme multi label classifier is new paradigm of thinking in machine learning
- It will be used for possible ranking and recommendation

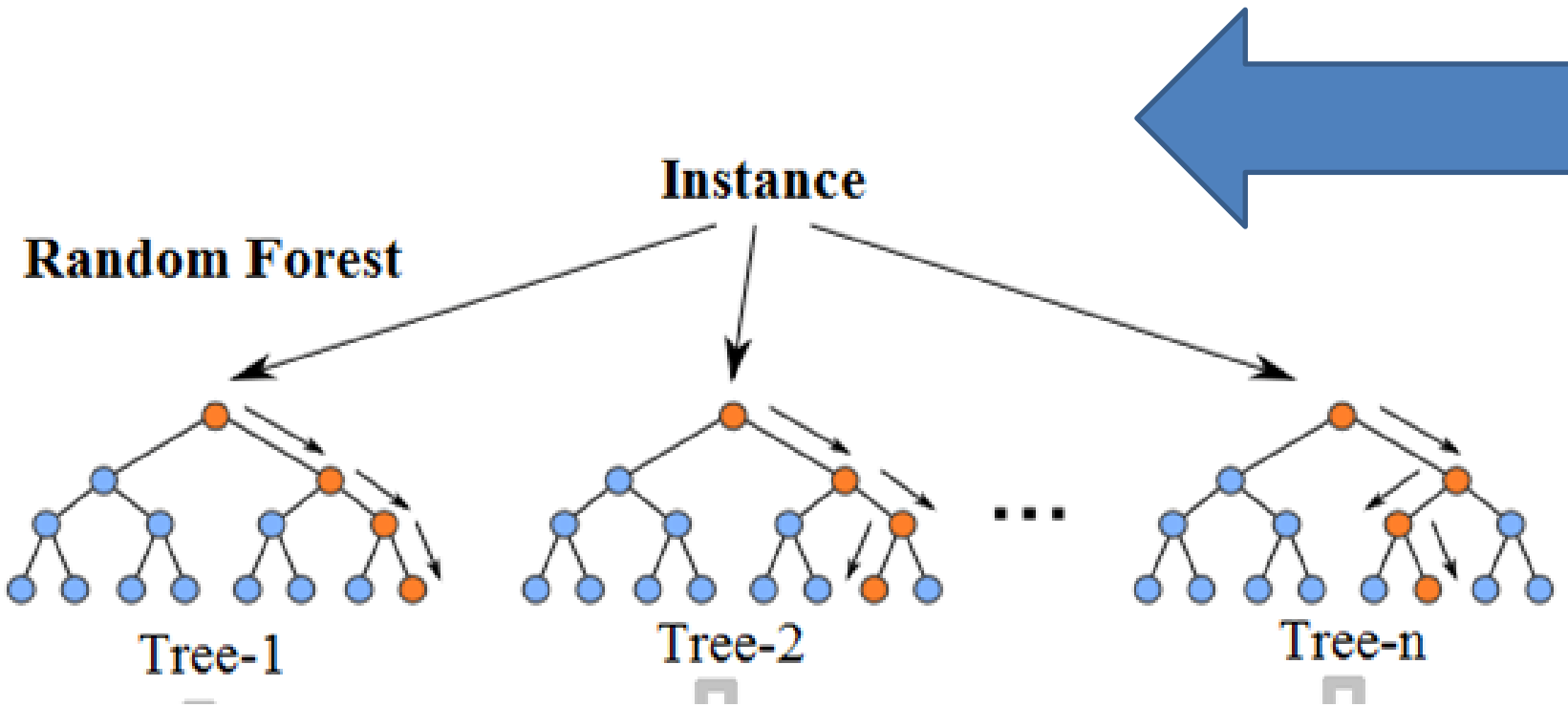
Bedrock work

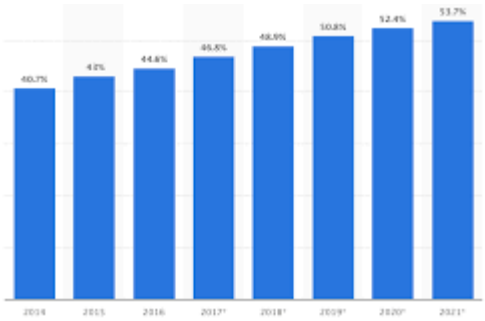
- M.L.R.F. (Multi Label Random Forest) was introduced 2 years before FastXML
- It has been trained on 10 million top queries of bing
- Was build for the aim of building a product that can predict from million of subset of bing queries, what and which may lead to click on an ad.

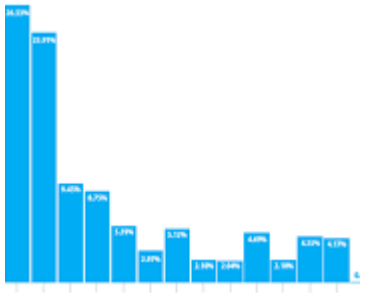
FastXML

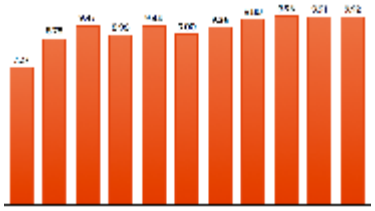
- Logarithm time prediction in milliseconds
- *ensemble of balanced Tree classifier.
- Accuracy gain up to 25% over competing methods
- *modes partitioned using nDCG.
- Up to 1000* faster training over the state of the art.
- *alternating minimization based optimization.

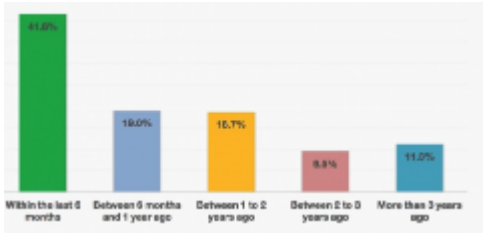
Ensemble of balanced Tree classifier











Ensemble of balanced Tree classifier

- Each Tree will have a probability and they will be voting for the over all one that will show the user based on user similarity
- It is the same architecture of MLRF
- The difference reside in the way we will be building the graph or exactly how we will be learning the difference between each subset

Modes partitioned using nDCG

- The problem here is how we will be Spliting those nodes (to left and right node)
- This paper used nDCG (normalized Discounted Cumulative Gain)
- the basic thing is that we should emphasis the thing that will be liked then care about what can be disliked

Modes partitioned using nDCG

why he did not go with traditionally loss function
such as the Geni index ,entropy or the hamming loss ?

Modes partitioned using nDCG

why he did not go with traditionally loss function
such as the Geni index ,entropy or the hamming loss ?

Because they does not have a concept of ranking related to them
,Moreover they place an equal emphasis on like and dislike items

Modes partitioned using nDCG

$$\mathcal{L}_{\text{nDCG@}k}(\mathbf{r},\mathbf{y}) = I_k(\mathbf{y}) \sum_{l=1}^{\kappa} \frac{y_{r_l}}{\log(1+l)} \quad (3)$$

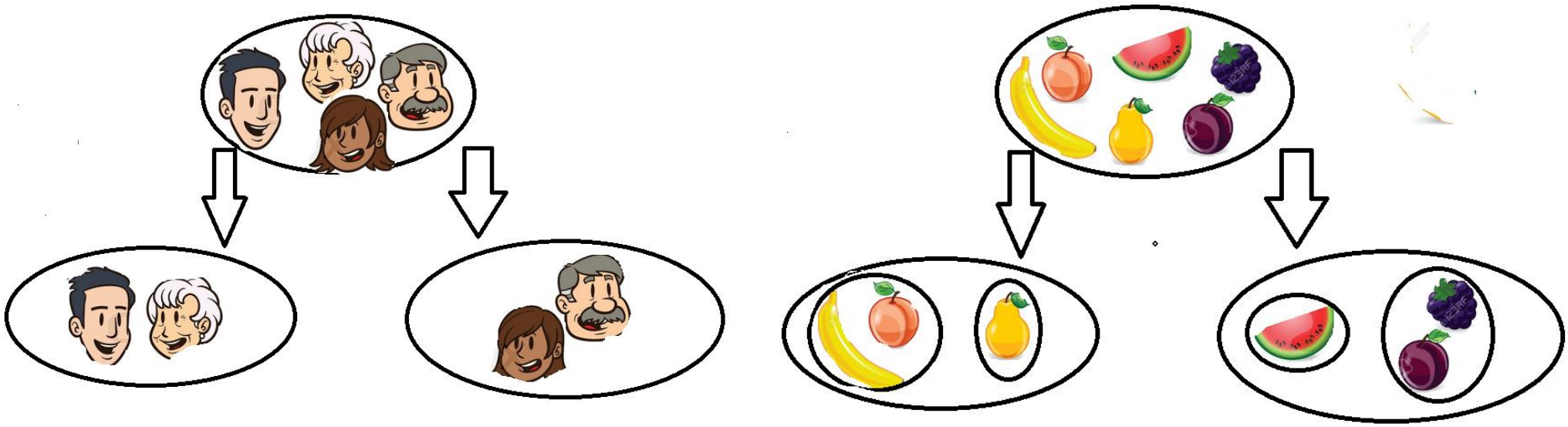
$$\text{where } I_k(\mathbf{y}) = \frac{1}{\sum_{l=1}^{\min(k,\mathbf{1}^\top \mathbf{y})} \frac{1}{\log(1+l)}} \quad (4)$$

$$\begin{aligned} \min \quad & \|\mathbf{w}\|_1 + \sum_i C_\delta(\delta_i) \log(1 + e^{-\delta_i \mathbf{w}^\top \mathbf{x}_i}) \\ & - C_r \sum_i \tfrac{1}{2}(1+\delta_i) \mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^+, \mathbf{y}_i) \\ & - C_r \sum_i \tfrac{1}{2}(1-\delta_i) \mathcal{L}_{\text{nDCG@}L}(\mathbf{r}^-, \mathbf{y}_i) \\ \text{w.r.t.} \quad & \mathbf{w} \in \mathcal{R}^D, \boldsymbol{\delta} \in \{-1,+1\}^L, \mathbf{r}^+, \mathbf{r}^- \in \Pi(1,L) \end{aligned} \quad (5)$$

- nDCG is a number between 0 and 1.
- Measure the quality of a ranking.
- FastXML partitions the current node's feature space by learning a linear separator w
- We will be learning W in which we will be assigning less then >0 and <0 to left and right respectively.

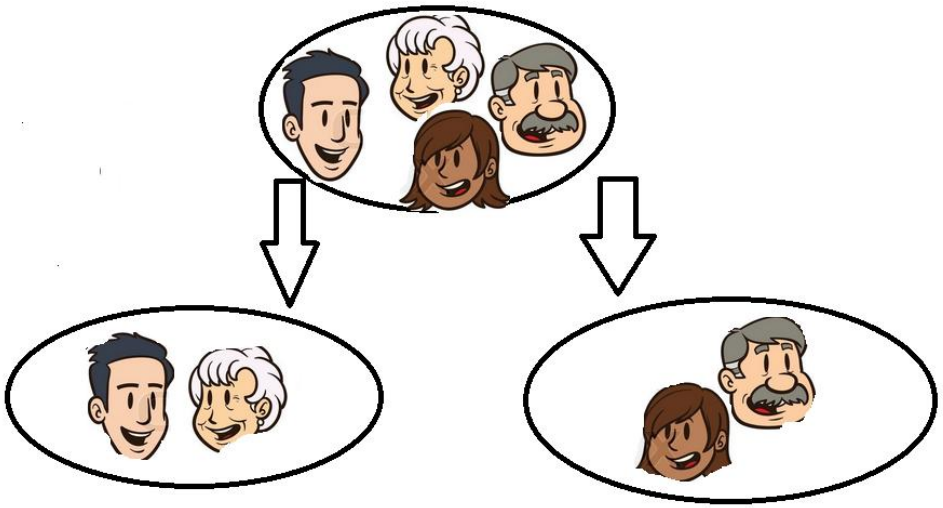
Alternating minimization based optimization

- nDCG is a hard function to optimize
- So they did something special to optimize it
- They will make the problem more complex first
- They have introduced the idea of shifting variable then optimizing it by the usage of nDCG
- This will not just introduce clustering over the items but rank over them also.

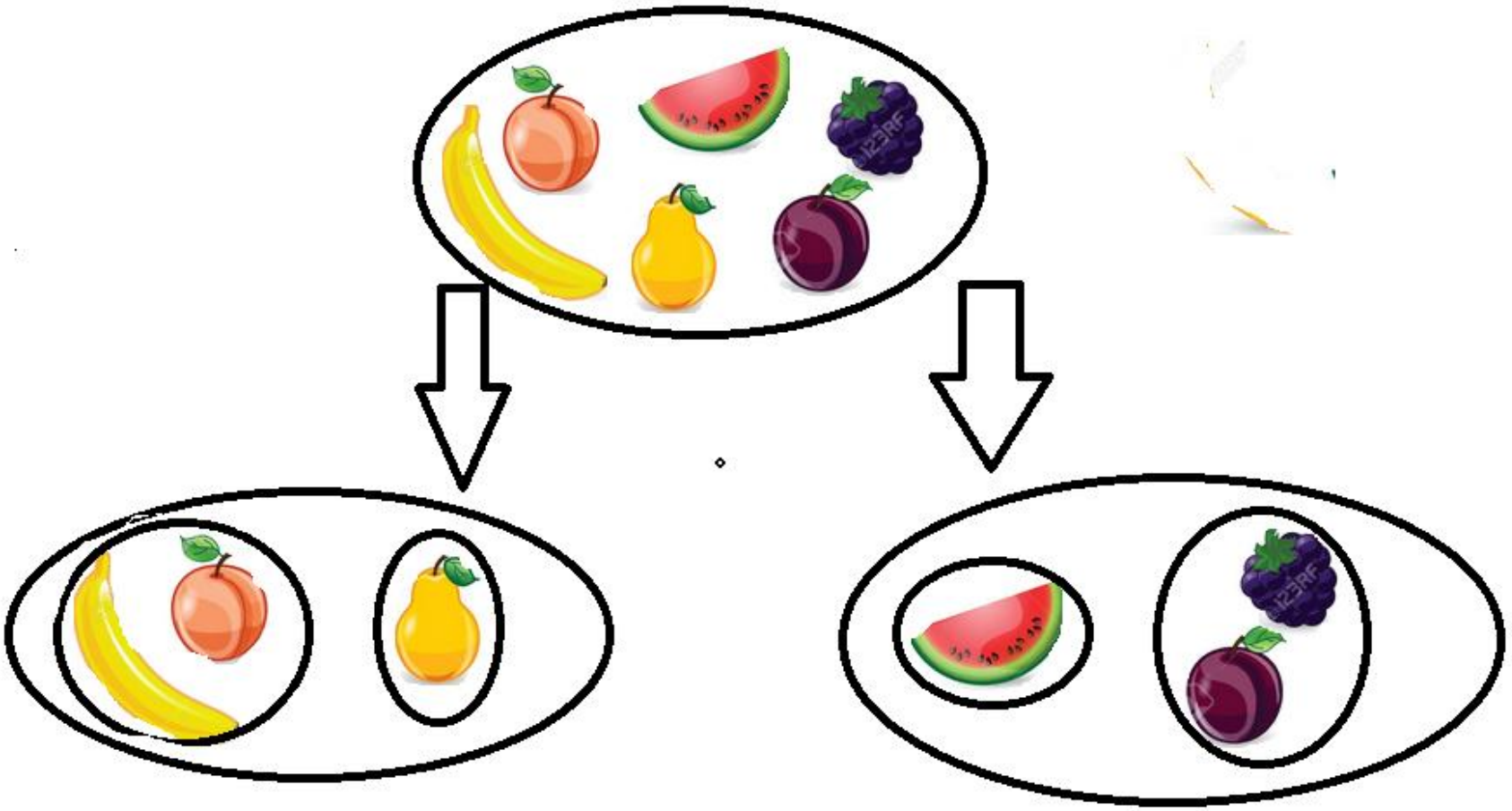


Alternating minimization based optimization

1. Randomly assign the person to right and left
2. we get a list and variable of what those people like.
3. based on the variable we can Generate a rank(nDCG)
4. we calculate nDCG and if it is low we shift the element from the left to the right.
5. we keep doing till we converge. NDCG will big
6. convergence occurs when there is no more shifting



Alternating minimization based optimization



Alternating minimization based optimization

- When no group of variables shift anymore, this is point of convergence
- Now we shall go up to the user and separate them from each other based on their variable that has already assigned left or right.
- We will keep doing this last description (function) recursively till we reach convergence.

Comparison and performance

Data set	Train N	Features D	Labels L	Test M	Avg. labels per pt.	Avg. pts per label
BibTeX	4880	1836	159	2515	2.40	111.71
Delicious	12920	500	983	3185	19.02	311.61
MediaMill	30993	120	101	12914	4.38	1902.16
RCV1-X	781265	47236	2456	23149	4.61	1510.13
WikiLSHTC	1892600	1617899	325056	472835	3.26	23.74
Ads-430K	1118084	87890	434594	502926	2.10	7.86
Ads-1M	3917928	164592	1082898	1563137	1.96	7.07

Comparison and performance

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML-T	$\mathbf{64.53 \pm 0.72}$	$\mathbf{40.17 \pm 0.63}$	$\mathbf{29.27 \pm 0.53}$
FastXML	63.26 ± 0.84	39.19 ± 0.66	28.72 ± 0.48
MLRF	62.81 ± 0.84	38.74 ± 0.69	28.45 ± 0.43
LPSR	62.95 ± 0.70	39.16 ± 0.64	28.75 ± 0.45
1-vs-All	63.39 ± 0.64	39.55 ± 0.65	29.13 ± 0.45

(b) Delicious $N = 13K, D = 500K, L = 983$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{69.65 \pm 0.82}$	$\mathbf{63.93 \pm 0.50}$	$\mathbf{59.36 \pm 0.57}$
MLRF	67.86 ± 0.70	62.02 ± 0.55	57.59 ± 0.43
LPSR	65.55 ± 0.99	59.39 ± 0.48	53.99 ± 0.31
1-vs-All	65.42 ± 1.05	59.34 ± 0.56	53.72 ± 0.50

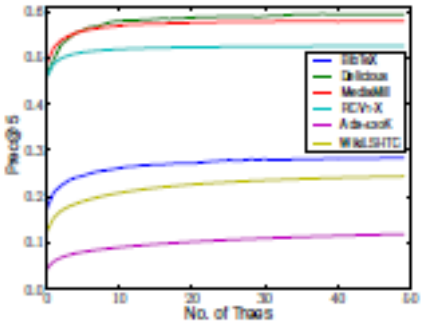
(c) MediaMill $N = 30K, D = 120, L = 101$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{87.35 \pm 0.27}$	$\mathbf{72.14 \pm 0.20}$	$\mathbf{58.15 \pm 0.15}$
MLRF	86.83 ± 0.18	71.18 ± 0.19	57.09 ± 0.16
LPSR	82.33 ± 2.15	66.37 ± 0.35	50.00 ± 0.20
1-vs-All	82.31 ± 2.19	66.17 ± 0.43	50.32 ± 0.56

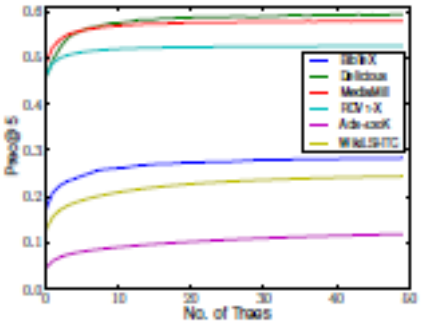
(d) RCV1-X $N = 781K, D = 47K, L = 2.5K$

Algorithm	P1 (%)	P3 (%)	P5 (%)
FastXML	$\mathbf{91.23 \pm 0.22}$	$\mathbf{73.51 \pm 0.25}$	$\mathbf{53.31 \pm 0.65}$
MLRF	87.66 ± 0.46	69.89 ± 0.43	50.36 ± 0.74
LPSR	90.04 ± 0.19	72.27 ± 0.20	52.34 ± 0.61
1-vs-All	90.18 ± 0.18	72.55 ± 0.16	52.68 ± 0.57

Data set	Tree Balance: $\frac{H}{log(N/MaxLeaf)}$			Avg. labels per leaf for FastXML
	FastXML	MLRF	LPSR	
BibTeX	$\mathbf{1.02 \pm 0.01}$	3.45 ± 0.01	1.56 ± 0.11	6.15
Delicious	$\mathbf{1.03 \pm 0.01}$	4.95 ± 0.01	3.14 ± 0.71	69.12
MediaMill	$\mathbf{1.01 \pm 0.01}$	1.59 ± 0.01	1.06 ± 0.01	10.26
RCV1-X	$\mathbf{1.02 \pm 0.01}$	5.62 ± 0.15	1.32 ± 0.02	18.73
WikiLSHTC	$\mathbf{1.01 \pm 0.01}$	-	3.69 ± 0.01	13.36
Ads-430K	$\mathbf{1.00 \pm 0.01}$	-	94.71 ± 0.01	10.97
Ads-1M	$\mathbf{1.00 \pm 0.01}$	-	105.83 ± 0.01	11.03



(a) Random order



(b) Forward sequential

Conclusion

- Extreme classifier tackle problems of tagging a lot of labels
- FastXML outperforms the state of the art and made a jump from 8 hours with a 1,000 node cluster to 20 minutes in a standard PC
- FastXML is based on MLRF but uses a different optimization technique (nDGC)
- FastXML is a new paradigm in multi label classification oriented toward recommendation
- Further work has been done toward minimize the time of execution

References

- 1- R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In WWW, pages 13–24, 2013.
- 2- M. N. Volkovs and R. S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In ICML, pages 1089–1096, 2009.
- 3- B. Hariharan, S. V. N. Vishwnathan, and M. Varma. efficient max-margin multi-label classification with applications to zero-shot learning. ML, 2012.

THANK YOU FOR YOUR ATTENTION.

Large-scale Multi-label Learning with Missing Labels

Hsiang-Fu Yu
Department of Computer Science, University of Texas at Austin
Prateek Jain
Purushottam Kar
Microsoft Research India, Bangalore
Inderjit S. Dhillon
Department of Computer Science, University of Texas at Austin

Table of Content

Abstract

- The multi-label classification problem has generated significant interest in recent years.
- However, existing approaches do not adequately address two key challenges: (a) scaling up to problems with a large number (say millions) of labels, and (b) handling data with missing labels.
- This paper directly address both these problems by studying the multi-label problem in a generic empirical risk minimization (ERM) framework.
- Its framework, despite being simple, is surprisingly able to encompass several recent label compression based methods which can be derived as special cases of this method.

....Abstract

- Extensive empirical results on a variety of benchmark datasets is presented and show that these methods perform significantly better than existing label compression based methods and can scale up to very large datasets such as a Wikipedia dataset that has more than 200,000 labels.

Introduction

- Large scale multi-label classification is an important learning problem with several applications to real-world problems such as image/video annotation and query/keyword suggestions.
- The goal in multi-label classification is to predict a label vector $y \in \{0, 1\}^L$, for a given data point $x \in R^d$.
- Recent research on multi-label classification has largely shifted its focus to the other end of the spectrum where the number of labels is assumed to be extremely large, with the key challenge being the design of scalable algorithms that offer real-time predictions and have a small memory footprint.

- This paper takes a more direct approach by formulating the problem as that of learning a low-rank linear model $Z \in R^{d\ X\ L}$
- This learning problem is cast in the standard ERM framework that allows us to use a variety of loss functions and regularizations for Z.
- Moreover, we can extend our formulation to handle missing labels. The ability to learn in the presence of missing labels is crucial as for most real-world applications, one cannot expect to accurately obtain all the labels for a given data point.
- In order to solve for the low-rank linear model that results from our formulation, we use the popular alternating minimization algorithm.
- Finally, we provide an extensive empirical evaluation of our method on a variety of benchmark datasets.

Related Works

- Binary Relevance (BR) which treats each label as an independent binary classification task, is quite accurate for multi-label learning. However, for a large number of labels, this method becomes infeasible due to increased model size and prediction time.
- Recently, techniques have been developed that either reduce the dimension of the labels (CPLST, BCS), or reduce the feature dimension, or both, such as WSABIE.
- Most of these techniques are tied to a specific loss function and/or cannot handle missing labels.

Algorithm Formulation

- $x_i \in R^d$
- $y_i \in \{0, 1\}^L$
- $l(y; f(x;Z)) \in R$
- $(X; Y)$ where $X = [x_1,.....,x_n]^T$ and $Y = [y_1\ y_2\ .....\ y_n]^T$

$$\hat{Z} = \arg\min_{Z} J(Z) = \sum_{i=1}^{n}\sum_{j=1}^{L} \ell(Y_{ij}, f^j(\boldsymbol{x}_i; Z)) + \lambda \cdot r(Z)$$

If there are missing labels, the loss is computed over known labels.

$$\hat{Z} = \arg\min_{Z} J_\Omega(Z) = \sum_{(i,j)\in\Omega} \ell(Y_{ij}, f^j(\boldsymbol{x}_i; Z)) + \lambda \cdot r(Z)$$

- Where *d* and *L* are large, we consider a low rank decomposition of the form of Z
- $Z = WH^T$, where $W \in R^{d \times k}$ and $H \in R^{L \times k}$

$$r(Z) = \|Z\|_{\mathrm{tr}}$$

$$\|Z\|_{\mathrm{tr}} = \frac{1}{2}\left(\|W\|_F^2 + \|H\|_F^2\right)$$

$$J_\Omega(W,H) = \sum_{(i,j)\in\Omega} \ell(Y_{ij}, \boldsymbol{x}_i^T W \boldsymbol{h}_j) + \frac{\lambda}{2}\left(\|W\|_F^2 + \|H\|_F^2\right)$$

- When either of W or H is fixed, $J_\Omega(W, H)$ becomes a convex function.
- This allows us to apply alternating minimization, a standard technique for optimizing functions with such a property.

Experimental Results

- Four standard datasets (bibtex, delicious, eurlex, and nus-wide), two datasets with *d* is much greater than *L* (autofood and compphys), and a very large scale Wikipedia based dataset, which contains about 1M wikipages and 200K labels.
- Competing methods: 1. LEML (Low rank Empirical risk minimization for Multi-Label Learning), 2. CPLST (Conditional Principal Label Space Transformation), 3. BCS (Bayesian Compressed Sensing), 4. BR (Binary Relevance), 5. WSABIE (Web Scale Annotation by Image Embedding)
- Evaluation criteria: top K accuracy, Hamming loss, and average AUC

Four standard datasets (bibtex, delicious, eurlex, and nus-wide), two datasets with *d* is much greater than *L* (autofood and compphys), and a very large scaleWikipedia based dataset, which contains about 1M wikipages and 200K labels.

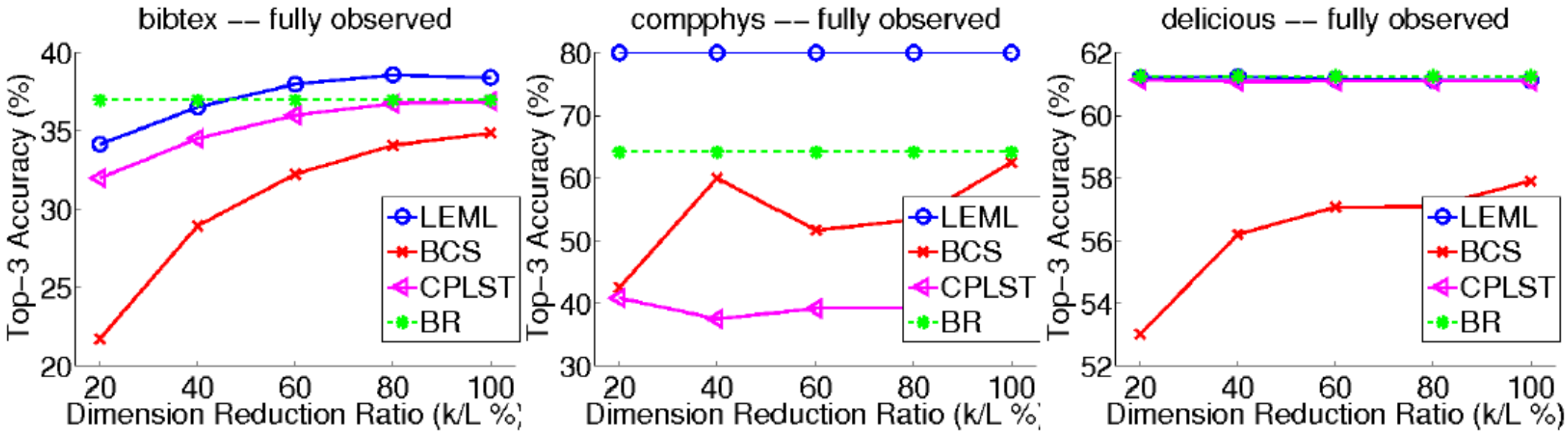
			Training set			Test set		
Dataset	d	L	n	$\bar{d}$	$\bar{L}$	n	$\bar{d}$	$\bar{L}$
bibtex	1,836	159	4,880	68.74	2.40	2,515	68.50	2.40
autofood	9,382	162	155	143.92	15.80	38	143.71	13.71
compphys	33,284	208	161	792.78	9.80	40	899.02	11.83
delicious	500	983	12,920	18.17	19.03	3,185	18.80	19.00
eurlex	5,000	3,993	17,413	236.69	5.30	1,935	240.96	5.32
nus-wide	1,134	1,000	161,789	862.70	5.78	107,859	862.94	5.79
wiki	366,932	213,707	881,805	146.78	7.06	10,000	147.78	7.08

Results with full labels

- Datasets are divided into two groups: *small datasets* (bibtex, autofood, compphys, and delicious) to which all methods are able to scale and *large datasets* (eurlex, nus-wide, and wiki) to which only LEML and WSABIE are able to scale.
- Dimension reduction based approaches are first compared to assess their performance with varying dimensionality reduction ratios.
- The next figure presents these results for LEML, CPLST and BCS on the squared L_2 loss with BR included for reference.

Clearly LEML consistently outperforms other methods for all ratios.

Fig. 1



(a) bibtex (b) compphys (c) delicious

Next we compare LEML to WSABIE with three surrogates (squared, logistic, and L_2-hinge)

Table 1

		Top-3 Accuracy				Average AUC			
		LEML			WSABIE	LEML			WSABIE
	k/L	SQ	LR	SH		SQ	LR	SH	
bibtex	20%	**34.16**	25.65	27.37	28.77	0.8910	0.8677	0.8541	**0.9055**
	40%	**36.53**	28.20	24.81	30.05	0.9015	0.8809	0.8467	**0.9092**
	60%	**38.00**	28.68	23.26	31.11	0.9040	0.8861	0.8505	**0.9089**
autofood	20%	**81.58**	80.70	**81.58**	66.67	0.9565	**0.9598**	0.9424	0.8779
	40%	76.32	**80.70**	78.95	70.18	0.9277	**0.9590**	0.9485	0.8806
	60%	70.18	80.70	**81.58**	60.53	0.8815	**0.9582**	0.9513	0.8518
compphys	20%	**80.00**	**80.00**	**80.00**	49.17	0.9163	0.9223	**0.9274**	0.8212
	40%	**80.00**	78.33	79.17	39.17	**0.9199**	0.9157	0.9191	0.8066
	60%	**80.00**	**80.00**	**80.00**	49.17	**0.9179**	0.9143	0.9098	0.8040
delicious	20%	**61.20**	53.68	57.27	42.87	0.8854	0.8588	**0.8894**	0.8561
	40%	**61.23**	49.13	52.95	42.05	0.8827	0.8534	**0.8868**	0.8553
	60%	**61.15**	46.76	49.58	42.22	0.8814	0.8517	**0.8852**	0.8523

Observations

- Table 1 shows that although the best loss function for each dataset varies, LEML is always superior to or competitive with WSABIE.
- Based on Figure 1 and the table:

    1) LEML can deliver accuracies competitive with BR even with a severe reduction in dimensionality

    2) On bibtex and compphys, LEML is even shown to outperform BR

    3) On autofood and compphys, CPLST seems to suffer from overfitting and demonstrates a significant dip in performance. In contrast, LEML,which brings regularization into the formulation performs well consistently on all datasets.

Larger Datasets

Table 2 shows results for LEML and WSABIE on the three larger datasets.

		LEML				WSABIE			
dataset	k	time (s)	top-1	top-3	AUC	time (s)	top-1	top-3	AUC
eurlex	250	**175**	**51.99**	**39.79**	**0.9425**	373	33.13	25.01	0.8648
	500	**487**	**56.90**	**44.20**	**0.9456**	777	31.58	24.00	0.8651
nus-wide	50	**574**	**20.71**	**15.96**	**0.7741**	4,705	14.58	11.37	0.7658
	100	**1,097**	**20.76**	**16.00**	**0.7718**	6,880	12.46	10.21	0.7597
wiki	250	**9,932**	**19.56**	14.43	**0.9086**	79,086	18.91	**14.65**	0.9020
	500	**18,072**	**22.83**	**17.30**	**0.9374**	139,290	19.20	15.66	0.9058

Observations

- LEML is clearly superior than WSABIE on all evaluation criteria.
- On wiki, although both methods share a similar performance for k = 250, on increasing k to 500, LEML again outperforms WSABIE.
- Also clearly noticeable is the stark difference in the running times of the two methods. The time phenomenon becomes more serious in WSABIE when L increases.
- All in all, the results clearly demonstrate the scalability and efficiency of LEML.

Results with missing labels

- We compare LEML, BCS, and BR.

	Top-3 Accuracy			Hamming loss			Average AUC		
	LEML	BCS	BR	LEML	BCS	BR	LEML	BCS	BR
bibtex	**28.50**	23.84	25.78	**0.0136**	0.2496	0.0193	**0.8332**	0.7871	0.8087
autofood	**67.54**	35.09	62.28	**0.0671**	0.2445	0.0760	**0.8634**	0.6322	0.8178
compphys	**65.00**	35.83	31.67	**0.0518**	0.2569	0.0566	**0.7964**	0.6442	0.7459

- Table above shows the results when 20% entries were revealed (i.e. 80% missing rate).
- The results clearly show that LEML outperforms BCS and LEML with respect to all three evaluation criteria.

Conclusion

- We studied the multi-label learning problem with missing labels in the standard ERM framework.
- We modeled our framework with rank constraints and regularizers to increase scalability and efficiency.
- To solve the obtained non-convex problem, we proposed an alternating minimization based method that critically exploits structure in the loss function to make our method scalable.
- This learning framework admits excess risk bounds that indicate better generalization performance than the existing methods like BR, something which the experiments also confirmed.
- Experiments additionally demonstrated that these techniques are much more efficient than other large scale multi-label classifiers and give superior performance than the existing label compression based approaches.

Future Works

- For future work, we would like to extend LEML to other (non decomposable) loss functions such as ranking losses and study conditions under which alternating minimization for our problem is guaranteed to converge to the global optimum.

References

- Agrawal, Rahul, Gupta, Archit, Prabhu, Yashoteja, and Varma, Manik. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the International World Wide Web Conference*, 2013.
- Bucak, Serhat Selcuk, Mallapragada, Pavan Kumar, Jin, Rong, and Jain, Anil K. Efficient multi-label ranking for multi-class learning: Application to object recognition. In *Proceedings of IEEE International Conference on Computer Vision*, 2009.
- Shamir, Ohad and Shalev-Shwartz, Shai. Collaborative Filtering with the Trace Norm: Learning, Bounding, and Transducing. In *24th Annual Conference on Learning Theory*, 2011.
- Kapoor, Ashish, Viswanathan, Raajay, and Jain, Prateek. Multilabel classification using bayesian compressed sensing.