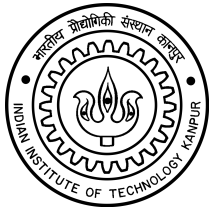
Distributional Semantics meets Multi Label Learning

Vivek Gupta[^#], Rahul Wadbude*, Nagarajan Natarajan[#]
Harish Karnick*, Prateek Jain[#] and Piyush Rai*



[^]School of Computing, University of Utah
[#] Microsoft Research Lab, India
*Indian Institute of Technology, Kanpur

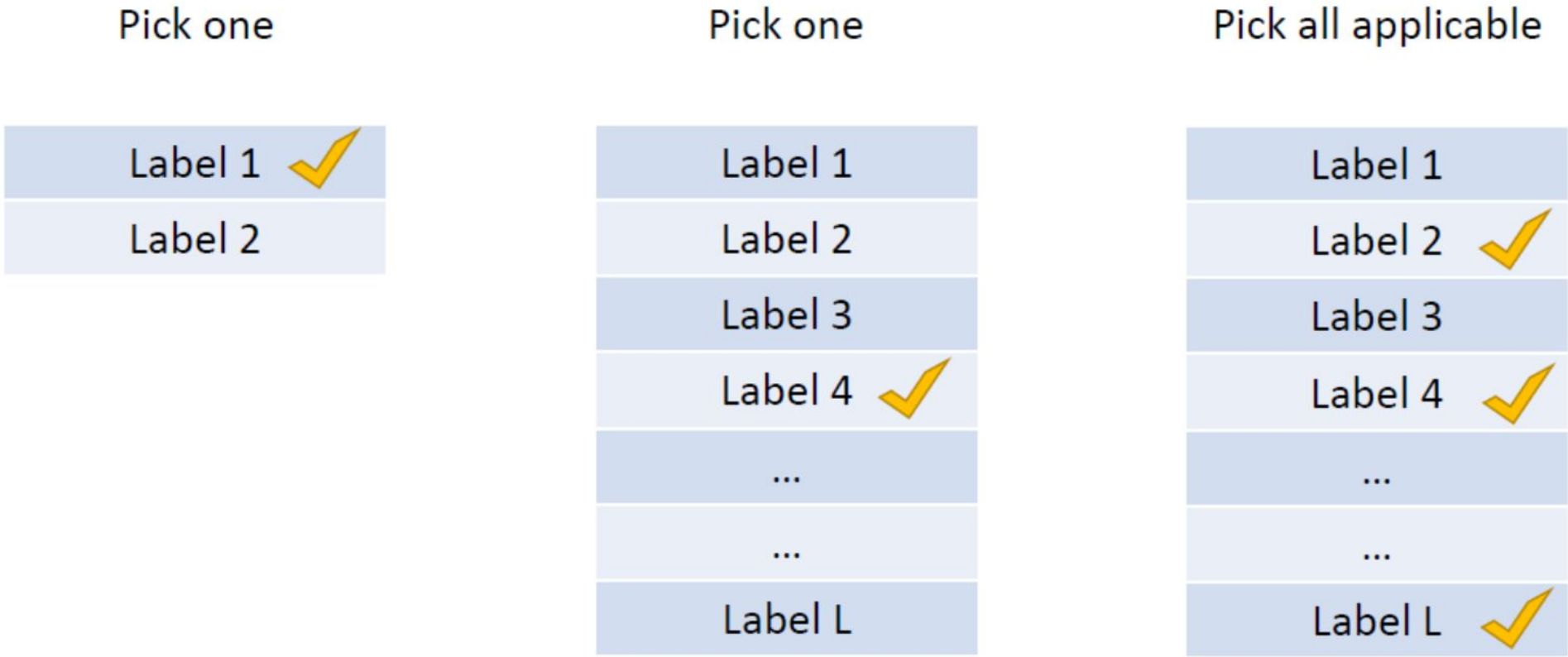
Microsoft Research

The Thirty-Third AAAI Conference on Artificial Intelligence,
AAAI 2019

*** Rahul is now @ (AlphaGrep, Bangalore)**

work done when Vivek at Microsoft

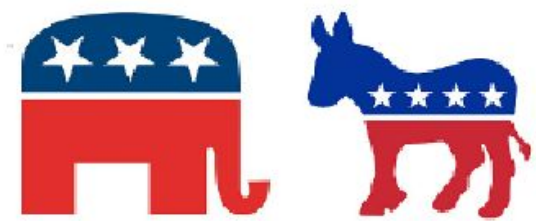
Classification Paradigms



Binary **Multi-class** **Multi-label**

Classification Paradigms

Pick one



Binary

Pick one

Multi-class

Pick all applicable

Multi-label

Extreme Multi-Label Learning

What all items would this user buy?

$f($  $)$

$f : \mathcal{X} \longrightarrow 2^{\mathcal{Y}}$

$\mathcal{X}$: Users

$\mathcal{Y}$: Items

Extreme Multi-Label Learning

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here

Create account Log in

Article Talk Read Edit View history Search

Bharata Natyam

From Wikipedia, the free encyclopedia

Bharathanatyam (Tamil: பரதநாட்டியம்) is a form of Indian classical dance that originated in the temples of Tamil Nadu.[1][2][3][4][5] It was described in the treatise *Natya Shastra* by Bharata around the beginning of the common era. Bharata Natyam is known for its grace, purity, tenderness, expression and sculpturesque poses. Lord Shiva is considered the God of this dance form. Today, it is one of the most popular and widely performed dance styles and is practiced by male and female dancers all over the world, although it is more commonly danced by women.[6]

Contents [hide]

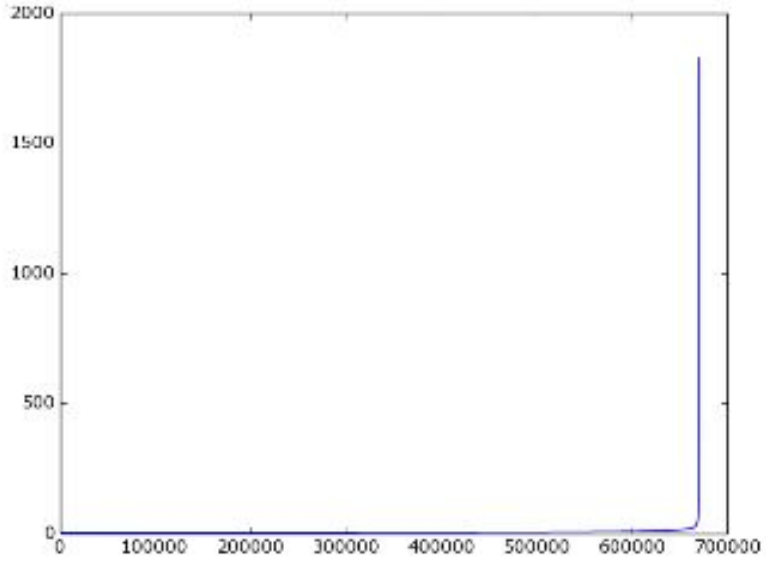
1 Etymology
2 Dance tradition
3 Essential ideas
  3.1 Spiritual symbolism
4 Medieval decline
5 Modern rebirth

Bharathanatyam

Dances by name, Indian culture, Performing arts in India, South India, Tamil culture

Challenges & Opportunity

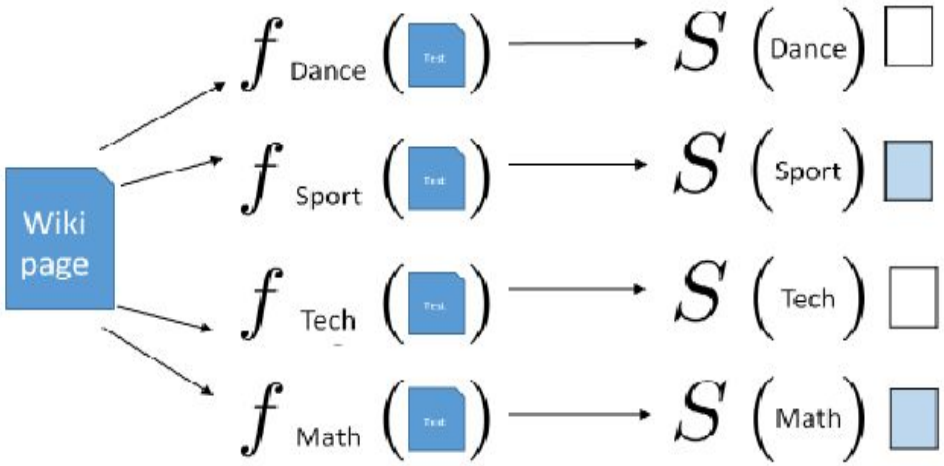
- Large scale setting
  - N (#examples), L(#Labels), D(#Feature Dimensions) in millions
- Exploiting label correlation
  - Challenging due to heavy tail distribution of Labels
- Missing Label in training & prediction set
  - Appropriate training & evaluation
- Fairness and Diversity
  - Useful for Recommendation



Primary Approaches

- 1 vs All or Binary Relevance Method
- Embedding or Dimensionality Reduction Method
- Tree or Ensemble Method
- (new) Combinations

One vs ALL Methods



Benefits

- Extremely flexible model
- In-depth theoretical analysis possible

Questions

- Are the L classifiers trained separately or jointly
- If jointly then what “joins” the classifier

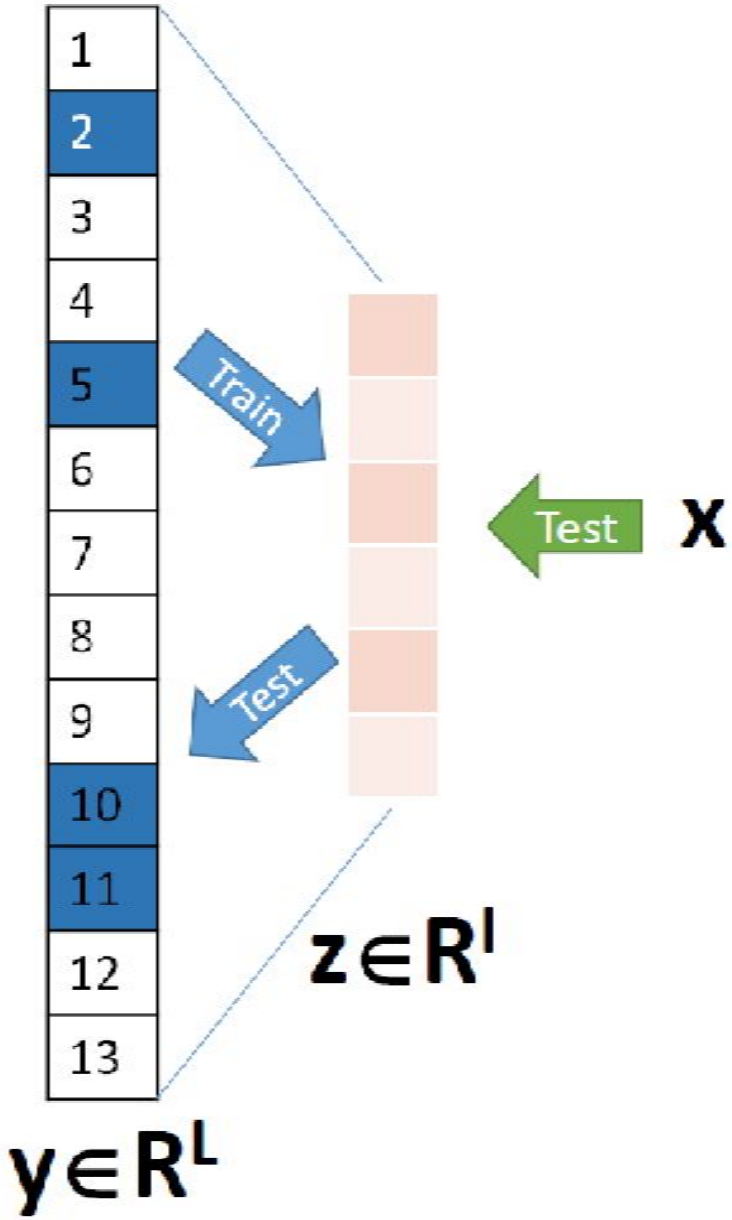
Consideration

- Training time ☹
- Test time ☹
- Model Size ☹

Embedding Methods

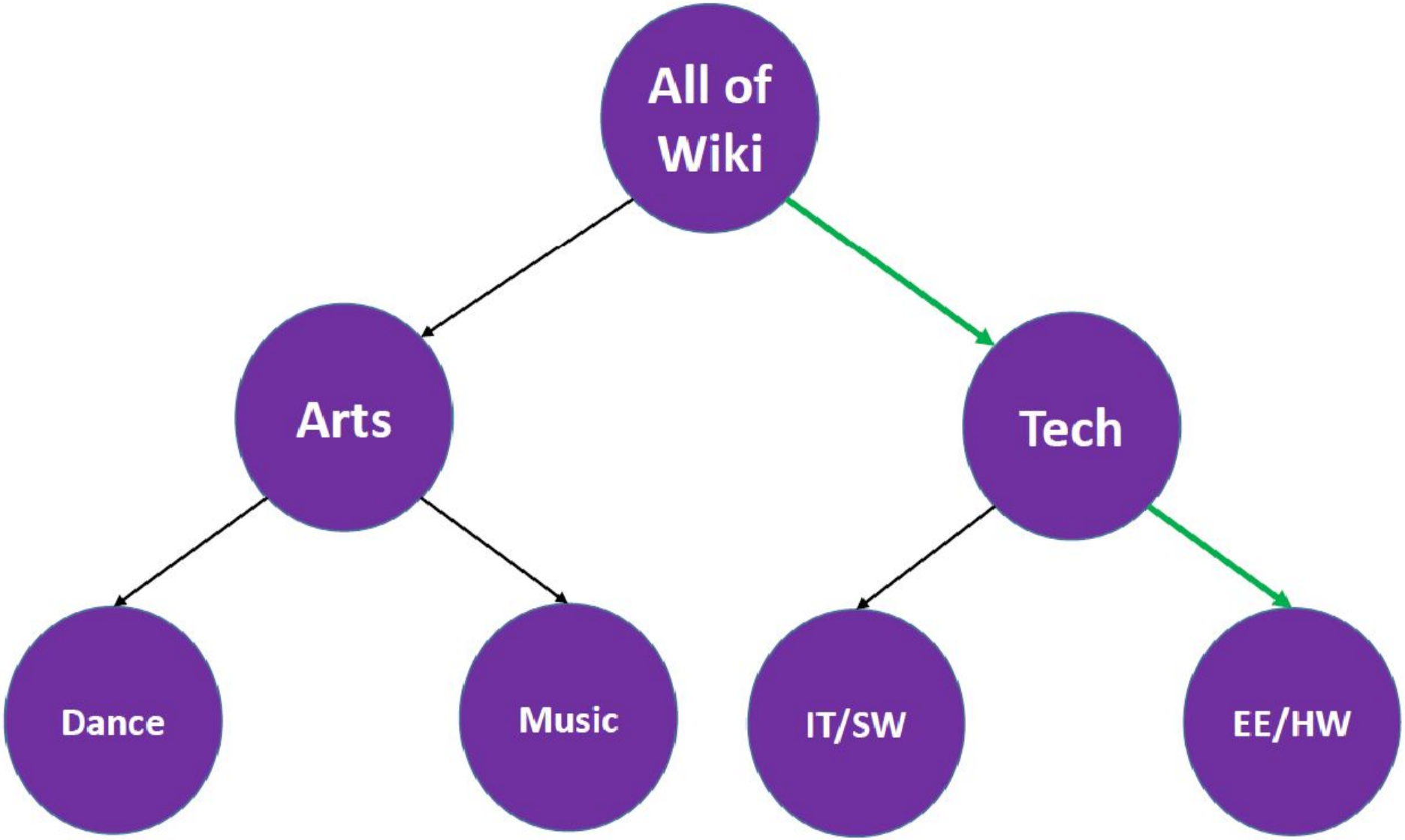
- *L >> 1* & has redundancies, project *L* in lower dimension
- Dimensionality Reduction
- Good theory and results, but expensive in prediction and training
- Questions
  - How to embed labels (locally/globally, linear/non-linear)
  - How to predict in the embedding space
  - How to “pull back” to the original label space
- CS, BCS, PLST,CPLST, LEML, SLEEC

Embedding Methods



- How to embed labels
  - RP(CS),CCA, PCA, Low local distortion proj., Learnt projections
- How to pull back
  - Sparse recovery, Nearest neighbor, Learnt projections
- Consideration
  - Training time ☺
  - Test time ☹
  - Model size ☺

Tree Methods



Tree Methods

- Partition the space of documents into several bins
    - Generally hierarchical partitioning as a tree
- At each leaf perform some classification task to predict
    - To increase efficiency and robustness use several trees(forest)
- Questions
    - Partitioning criteria (clustering, ranking, classification)
    - Leaf action (constant labelling, use of another multi labeller)
    - Ensemble size and aggregation method (single, multiple)
    - LPSR, MLRF, FAST-XML
- Consideration
    - Good accuracy, fast prediction, huge models

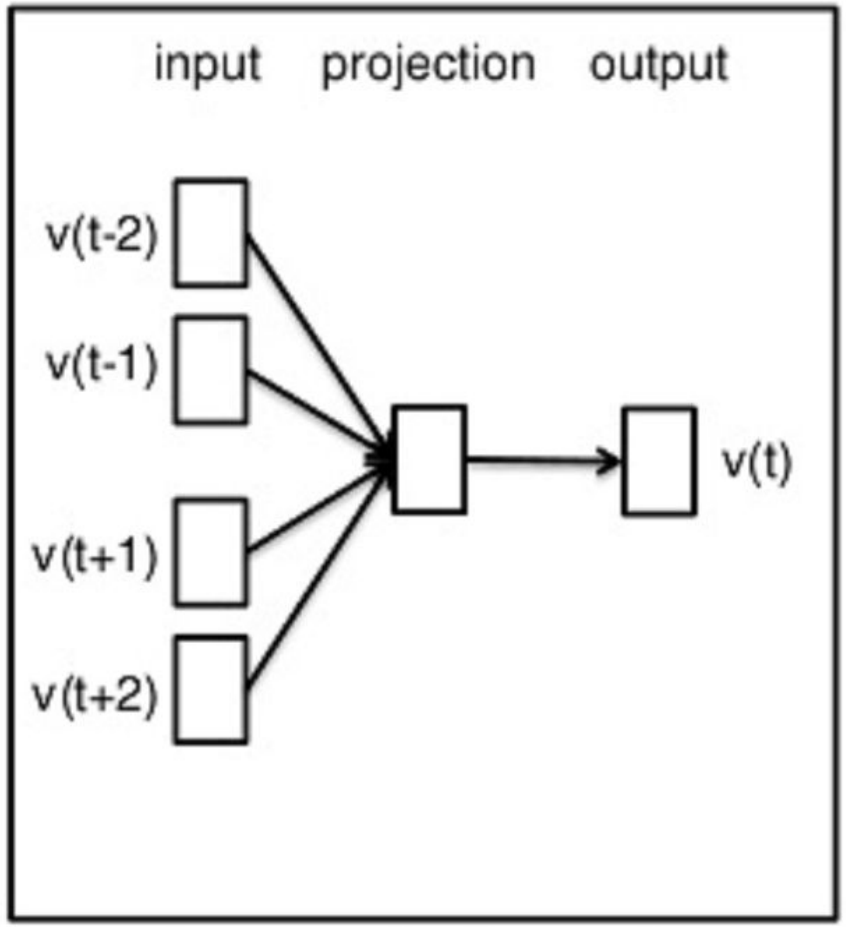
Comparison of Methods

Name	“Accuracy”	Scalability	Prediction Cost	Model Size	Well Understood
1-vs-All	Meh!	Yikes!	Are you kidding me!	Did I not make myself clear?	Now we are talking! Excellent
Embedding	Good/Best	Good/Best	Good	Good	Good
Tree	Good/Best	Good/Best	Best	Large	Meh!

Distributional Semantics

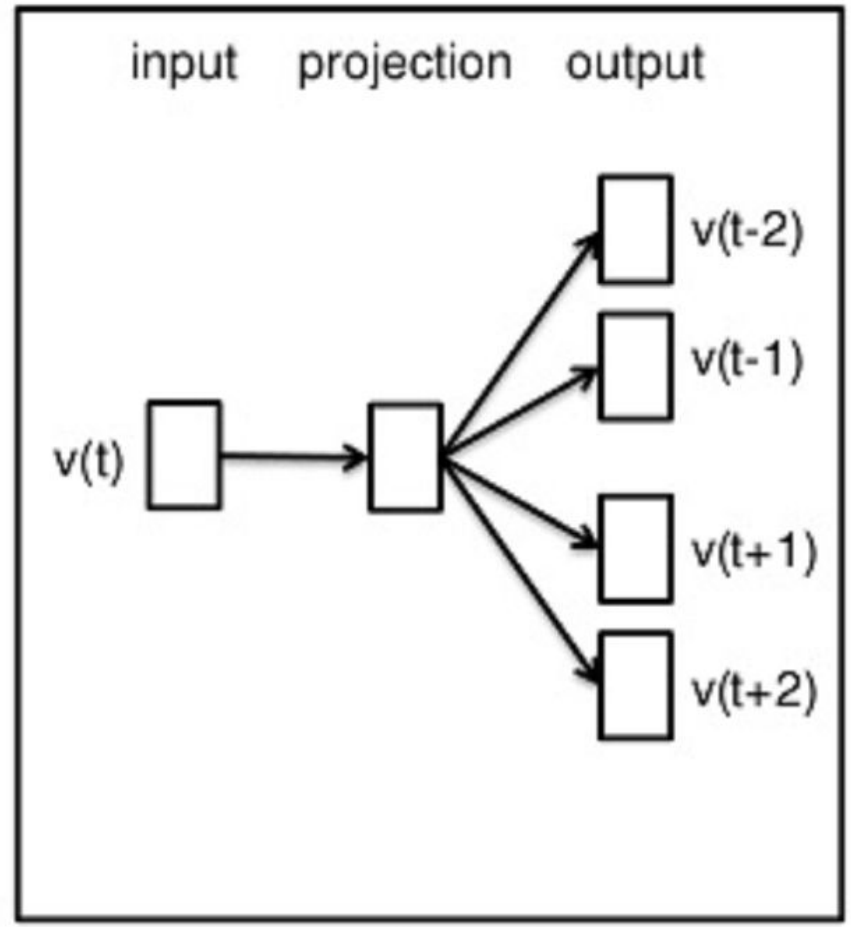
- **Hypothesis:** Similar words occurring in similar context are closer in embedded vector space.
- Each word $w_i \in V$ is represented using a vector v_{wi} . Semantically similar words occur closer in vector space.
- Good semantic representation of words exists i.e. **Word2vec (SGNS, CBOW)** created by Mikolov et al., **Glove** (Socher et al.) and many more.

CBoW



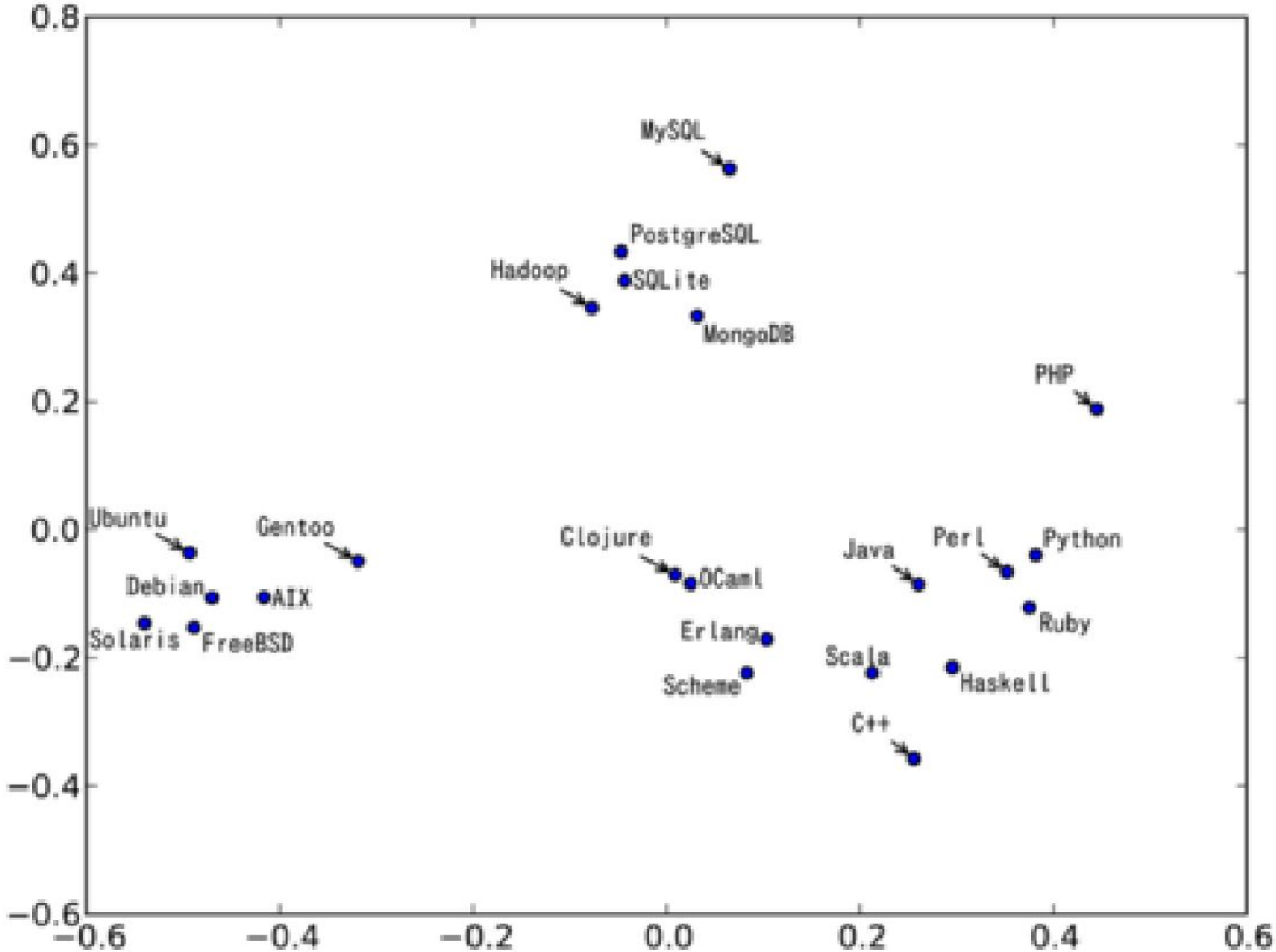
- given context words
- predict a probability of a target word

Skip-gram



- given a target word
- predict a probability of context words

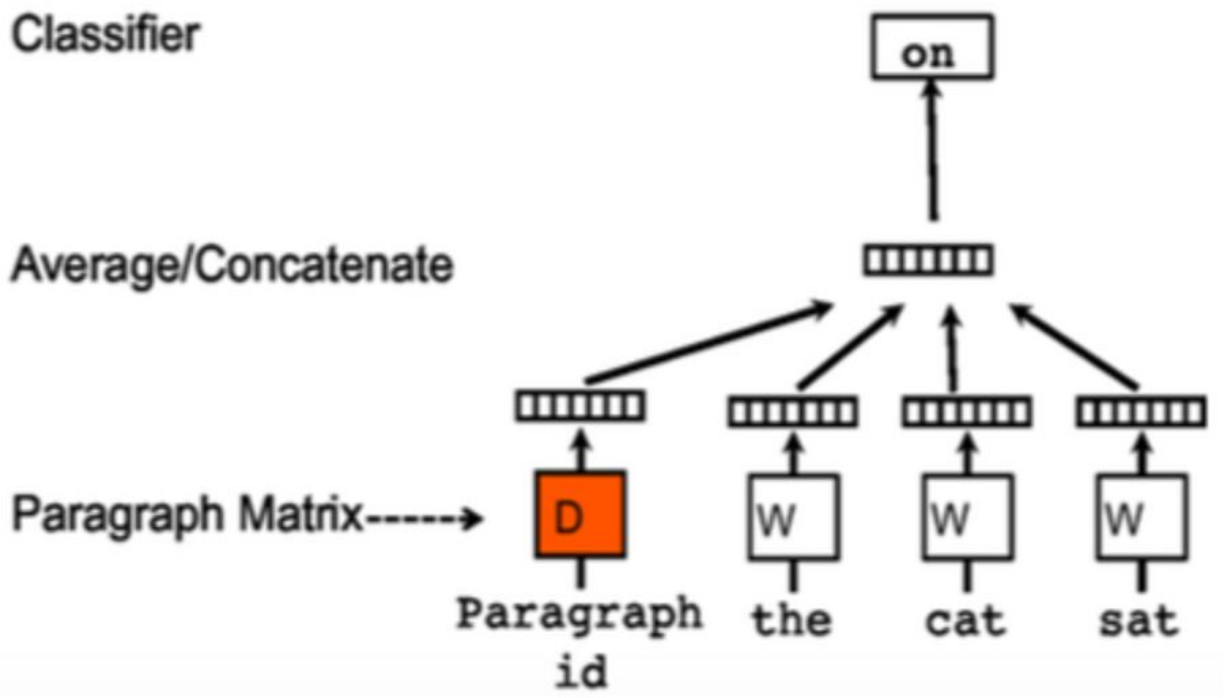
Distributional Semantics

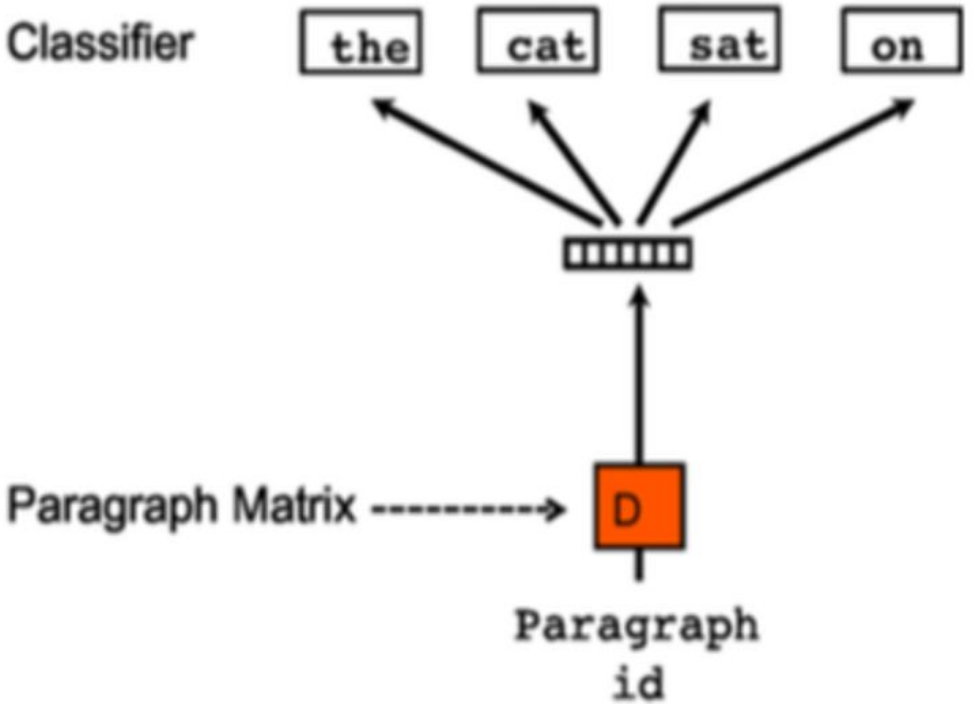


Sentence Vectors

- Each word d_i is represented using a vector v_{di} . Vector encode the semantic meaning of the sentence.
- Idea to treat sentence as a pseudo words occurring in context with all words in the sentence
- Good semantic representation of sentence exists i.e. **Doc2vec (PV-DM, PV-DBOW)** created by Mikolov et al., **Skip Thought** (Kiros et al.) and many more.
- Document vector are embedded in same space as word vectors (can use cosine similarity)

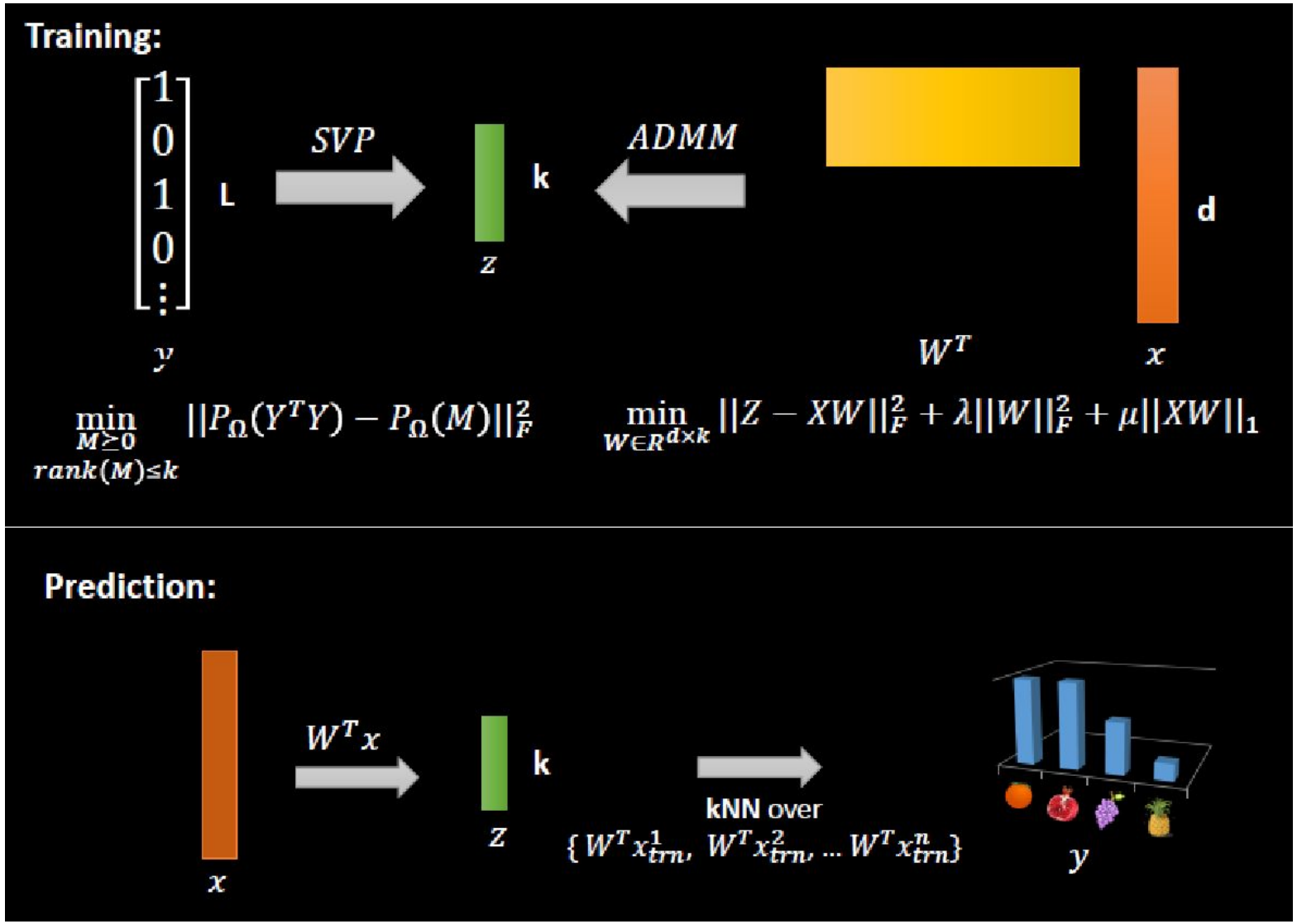
PV-DBOW (L) & PV-DM (R)





SLEEC - Embedding Based Algorithm

- Non linear neighborhood preserving low rank embedding of label vectors



SLEEC Training - Embedding Algorithm

1. Step I : Label vector embedding
   - $Y \in R^{N \times L}$ embed to $Z \in R^{N \times l}$, here $l < L$.
   - K-nearest neighbor distance preservation, dst($\mathbf{z}_i, \mathbf{z}_j$) $\approx$ dst($\mathbf{y}_i, \mathbf{y}_j$) if i $\in$ kNN(j) here, dst denote distance metric. Here y_i and z_i are i^th rows of Y and Z.
   - Singular value projection (SVP[9])
2. Step II : Fast Regressor Learning
   - regressor $\mathbf{V}$ are learned to predict $\mathbf{z}_i = \mathbf{V}\mathbf{x}_i$.
   - stores set S of $\mathbf{V}\mathbf{x}_i$ $\forall$ training examples.
   - Alternating direction method of multipliers (ADMM][3]).
3. Step-I and Step-II are learn independently

SLEEC Prediction - Embedding Algorithm

1. given x_j get $\mathbf{z}_j = \mathbf{V}\mathbf{x}_j$.
2. get k nearest neighbor of $\mathbf{z}_j$ in S. predict y_j union of kNN y_i i.e. $\mathbf{y} = \sum_{i:\mathbf{V}\mathbf{x}_i \in KNN(\mathbf{V}\mathbf{x})} \mathbf{y}_i$

Remark

Extra speedup by clustering on x and processing them independently.

SLEEC Training - Embedding Details

- **Goal**: Given **Y**, learn embeddings **Z** of smaller dimension.
- SLEEC presents following formulation:

$$\min_{Z \in \mathbb{R}^{d \times n}} \| P_\Omega(Z'Z) - P_\Omega(Y'Y) \|_F^2 + \lambda \| Z \|_1$$

where

$$(P_\Omega(Y'Y))_{ij} = \left\{ \begin{array}{ll} < \mathbf{y}_i, \mathbf{y}_j >, & \text{if } (i,j) \in \Omega \\ 0, & \text{otherwise} \end{array} \right\}$$

- where Ω denotes the set of knn.
- Use SVP for training

SLEEC Training - Regressor Details

- **Goal**: Given **X** and **Z**, learn regression matrix **V**.
- SLEEC presents following formulation:

$$\min_{V \in \mathbb{R}^{L \times d}} \| Z - VX \|_F^2 + \lambda \| V \|_F^2 + \mu \| VX \|_1$$

- Use ADMM for training

Remark

Use VX instead of Z for getting sparse embedding.

Proposed ExMLDS1: SGNS meets Label Embedding

- **word2vec** embedding using **Skip Gram Negative Sampling** objective

$$P(\text{Observing } (w, w')) = \sigma(\langle \mathbf{z}_w, \mathbf{z}_{w'} \rangle) = \frac{1}{1 + \exp(\langle -\mathbf{z}_w, \mathbf{z}_{w'} \rangle)}$$

$$\max_{\mathbf{z}} \sum_{w} \left(\sum_{w':(w',w)} \log\left(\sigma(\langle \mathbf{z}_w, \mathbf{z}_{w'} \rangle)\right) + \frac{n_-}{\#w} \sum_{w''} \log\left(\sigma(-\langle \mathbf{z}_w, \mathbf{z}_{w''} \rangle)\right) \right)$$

- **replacing words** with the **example label vectors** in the training sets

$$\max_{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n} \sum_{i=1}^{n} \left(\sum_{j:\mathcal{N}_k(\mathbf{y}_i)} \log\left(\sigma(\langle \mathbf{z}_i, \mathbf{z}_j \rangle)\right) + \frac{n_-}{n} \sum_{j'} \log\left(\sigma(-\langle \mathbf{z}_i, \mathbf{z}_{j'} \rangle)\right) \right)$$

Proposed ExMLDS1 Objective

Representation

- $\mathbb{S}^i = \{j; j \in NN_i\}_{j=1}^{K}$, here NN_i denote nearest neighbour of y_i.
- $K_{ij} = cos(z_i, z_j) = \frac{z_i.z_j}{\|z_i\|\|z_j\|}$, where z_i,z_j label embedding of y_i, y_j.
- $z_i = Vx_i$, where $V \in \mathbb{R}^{l \times D}$.

Optimization Objective

$$P_i(j \in S^i) = \sigma(\gamma K_{ij})$$

$$P_i(j \notin S^i) = 1 - \sigma(\gamma K_{ij}) = \sigma(-\gamma K_{ij})$$

$$\mathbb{J}_i = \sum_{j \in S_i} \log(P_i(j \in S^i)) + \sum_{k \notin S_i}^{K} \log(P_i(k \notin S^i))$$

$$\mathbb{J}_i = \sum_{j \in S_i} \log(\sigma(\gamma K_{ij})) + \sum_{k \notin S_i}^{K} \log(\sigma(-\gamma K_{ik}))$$

SGNS as Matrix Factorization

Theorem

SGNSs objective is equivalent to weighted matrix factorization of shifted positive point wise mutual information (SPPMI) matrix [8]

Shifted PPMI:

$$PMI_{ij}(M) = log\left(\frac{M_{ij} * |M|}{\sum_k M_{(i,k)} * \sum_k M_{(k,j)}}\right)$$

$$SPPMI_{ij}(M) = \max(PMI_{ij}(M) - log(k), 0)$$

Here, PMI is point wise mutual information matrix of M and $|M|$ represent sum of all element in M.

Proposed ExMLDS2 Algorithm

Input. Training data $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n$.

1. Compute $\widehat{M} :=$ SPPMI(M) in (4), where $M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$.
2. Let $U, S, V = \text{svd}(\widehat{M})$, and preserve top d' singular values and singular vectors.
3. Compute the embedding matrix $Z = US^{0.5}$, where $Z \in \mathbb{R}^{n \times d'}$, where ith row gives $\mathbf{z}_i$
4. Learn V s.t. $XV^T = Z$ using ADMM [3], where X is the matrix with $\mathbf{x}_i$ as rows.

return V, Z

- Overall, multi iter *SVP* algorithm replaced with single step *SPPMI* matrix factorization using *SVD*
- Regression and Prediction algorithm remain exactly similar to the *SLEEC* algorithm
- We observe the *ExMLDS2* is faster than both *SLEEC* and *EXMLDS1 (SGNS)* training

Incorporating Label-Label Correlation

- *PV-DBoW* objective while learning embeddings is to maximize similarity between embedded sentence and words that compose the sentence
- Negative sampling is included in the objective to minimize the similarity between the sentence embeddings and embeddings of frequent words
- In multi-label learning, we want to learn the embeddings of labels as well as instances jointly
- Overall Idea: think of labels as individual words, whereas label vectors (or instances with the corresponding label vectors) as sentence.
- Many real world problems also have auxiliary additional label correlation information, such as label-label co-occurrence

Incorporating Label-Label Correlation

$$\max_{z,\bar{z}} \mathbb{O}_{z,\bar{z}} = \mu_1 \mathbb{O}^1_{\bar{z}} + \mu_2 \mathbb{O}^2_{z} + \mu_3 \mathbb{O}^3{}_{\{z,\bar{z}\}}$$

$$\mathbb{O}^1_{\bar{z}} = \sum_{i=1}^{L} \left(\sum_{j:\mathcal{N}_k(C(i,:))} \log\left(\sigma(\langle \bar{\mathbf{z}}_i, \bar{\mathbf{z}}_j \rangle)\right) + \frac{n^1_-}{L} \sum_{j'} \log\left(\sigma(-\langle \bar{\mathbf{z}}_i, \bar{\mathbf{z}}_{j'} \rangle)\right) \right)$$

$$\mathbb{O}^2_{z} = \sum_{i=1}^{n} \left(\sum_{j:\mathcal{N}_k(M(i,:))} \log\left(\sigma(\langle \mathbf{z}_i, \mathbf{z}_j \rangle)\right) + \frac{n^2_-}{n} \sum_{j'} \log\left(\sigma(-\langle \mathbf{z}_i, \mathbf{z}_{j'} \rangle)\right) \right)$$

$$\mathbb{O}^3_{\{z,\bar{z}\}} = \sum_{i=1}^{L} \left(\sum_{j:y_{ij}=1} \log\left(\sigma(\langle \mathbf{z}_i, \bar{\mathbf{z}}_j \rangle)\right) + \frac{n^3_-}{L} \sum_{j'} \log\left(\sigma(-\langle \mathbf{z}_i, \bar{\mathbf{z}}_{j'} \rangle)\right) \right)$$

Proposed ExMLDS3 Training

Input. Training data $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n$ and C (label-label correlation matrix) and objective weighting μ_1,μ_2 and μ_3.

1. Compute $\widehat{A} := \mathsf{SPPMI}(A)$ in (4); write

$$A = \begin{pmatrix} \mu_2 M & \mu_3 Y \\ \mu_3 Y^T & \mu_1 C \end{pmatrix},$$

$M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$, Y is label matrix with $\mathbf{y}_i$ as rows.

2. Let $U, S, V = \mathrm{svd}(\widehat{A})$, and preserve top d' singular values and singular vectors.

3. Compute the embedding matrix $Z = US^{0.5}$; write

$$Z = \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix},$$

where rows of $Z_1 \in \mathbb{R}^{n \times d'}$ give instance embedding and rows of $Z_2 \in \mathbb{R}^{L \times d'}$ give label embedding.

4. Learn V s.t. $XV^T = Z_1$ using ADMM [3], where X is the matrix with $\mathbf{x}_i$ as rows.

return V, Z

Proposed ExMLDS3 Prediction

Input: Test point: $\mathbf{x}$, no. of nearest neighbors k, no. of desired labels p, V, embeddings Z_1 and Z_2.
1. Use Algorithm 1 (Step 3) with input Z_1, k, p to get score s_1.
3. Get score $s_2 = Z_2 V x$
4. Get final score $s = \frac{s_1}{\|s_1\|} + \frac{s_2}{\|s_2\|}$.
return top p scoring labels according to $\mathbf{s}$.

- At prediction take weighted union of labels from instance embedding and label embedding
- We explore equal weighting, but more sophisticated techniques could also be employed
- Label embedding from label-label correlation helps in handling missing label efficiently

Jointly Learning of Embedding and Regressor

- Joint learning of embedding and regressor objective for the SLEEC algorithm

$$\min_{V \in \mathbb{R}^{\hat{L} \times d}} \|P_\Omega(Y^T Y) - P_\Omega(X^T V^T V X)\|_F^2 + \lambda \|V\|_F^2 + \mu \|VX\|_1.$$

- Optimizing above objective is a significant challenge as it's highly non-convex as well as non-differentiable

- However, we can do joint learning of embedding and regression as it's totally tractable for our new objective

Proposed ExMLDS4 Algorithm

- With our proposed objective?

$$\mathbb{O}_i = \sum_{j:\mathcal{N}_k(\mathbf{y}_i)} \log\left(\sigma(K_{ij})\right) + \frac{n_-}{n} \sum_{j'} \log\left(\sigma(-K_{ij'})\right),$$

- Joint learning possible, although non-convex nature

$$\nabla_V \mathbb{O}_i = \sum_{j:\mathcal{N}_k(\mathbf{y}_i)} \sigma(-K_{ij}) \nabla_V K_{ij} - \frac{n_-}{n} \sum_{j'} \sigma(K_{ij'}) \nabla_V K_{ij'}$$

$$\nabla_V K_{ij} = -ab^3 c \mathbf{z}_i (\mathbf{x}_i)^T - abc^3 \mathbf{z}_j (\mathbf{x}_j)^T + bc(\mathbf{z}_i \mathbf{x}_j^T + \mathbf{z}_j \mathbf{x}_i^T)$$

$$a = \mathbf{z}_i^T \mathbf{z}_j, b = \frac{1}{\|\mathbf{z}_i\|}, c = \frac{1}{\|\mathbf{z}_j\|}$$

Proposed ExMLDS4 Algorithm

Input. Training data $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n$. no. of nearest neighbors k, Gaussian initialize regression matrix V
while $t = i \neq n$ **do**
1. Compute $\mathbb{O}_i$ in (12) and it's gradient $\nabla_V \mathbb{O}_i$ in (13);
2. Perform stochastic gradient descent update for V,

$$\mathbb{V} \leftarrow \mathbb{V} + \eta \nabla_V \mathbb{O}_i$$

3. Update η using Adam [10].

end while
return V

- Joint learning is a major advancement over *SLEEC* algorithms as joint learning improve performance
- Similar algorithm (*AnnexML, KDD'17*) using *DSSM* instead of *SGNS* objective proposed by Yukihiro Tagami, Yahoo! Research Japan

Dataset Statistics

- We experimented on multiples small and large multi label learning datasets taken (same splits) from Manik Varma extreme classification repository

Dataset	Feature	Label	Train	Test
Bibtex (Katakis et al., 2008; Prabhu and Varma, 2014)	1836	159	4880	2515
Delicious (Tsoumakas et al., 2008; Prabhu and Varma, 2014)	500	983	12920	3185
EURLex-4K (Loza Mencía and Fürnkranz, 2008; Prabhu and Varma, 2014)	5000	3993	15539	3809
rcv1v2 (Lewis et al., 2004; Prabhu and Varma, 2014)	47236	101	3000	3000
Delicious-200K (Tsoumakas et al., 2008; Bhatia et al., 2015)	782585	205443	196606	100095
MediaMill (Snoek et al., 2006; Bhatia et al., 2015)	120	101	30993	12914
Wiki10-31K (Bhatia et al., 2015; Zubiaga, 2012)	101938	30938	14146	6616
AmazonCat-13K (McAuley and Leskovec, 2013)	203882	13330	1186239	306782
WikiLSHTC-325 (Prabhu and Varma, 2014; Bhatia et al., 2015)	1617899	325056	1778351	587084
Wikipedia-500K	2381304	501070	1813391	783743
Amazon-670K (McAuley and Leskovec, 2013; Bhatia et al., 2015)	135909	670091	490449	153025

Evaluation Metric

- Two popular metric P@k and nDCG@k are commonly used .
- $P@k =$

$$\frac{1}{k} \sum_{l \in rank_k(\hat{\mathbf{y}})} \mathbf{y}_l$$

- $nDCG@k =$

$$\frac{\sum_{l \in \text{rank}_k(\hat{\mathbf{y}})} \frac{\mathbf{y}_l}{\log(l+1)}}{\sum_{l=1}^{min(k, \|\mathbf{y}\|_0)} \frac{1}{log(l+1)}}$$

- Here $\hat{\mathbf{y}} \in \mathbb{R}^L$ is predicted score vector and $\mathbf{y} \in \{0,1\}^L$ is ground truth vector.

Baselines

- We compared our models with multiples baseline *(embedding, tree based, one vs all)*, all results taken again from *Manik Varma extreme classification* repository

Embedding Based	Sparsity Based	Tree Based	Others
SLEEC	PD-Sparse	FastXML	One vs All
AnnexML	PPD-Sparse	PfastreXML	DiSMEC
DXML		Parabel(2019)	
XML-CNN		SwiftXML(2019)	
LEML		Slice (2019)	
LEML-IMC			

- We focused on comparing our methods on state of art embedding based extreme classification algorithm

ExMLDS 1/2 Performance

Method	Bibtex	Delicious	Eurlex	Mediamill	Delicious-200K
ExMLDS1	**23**	**259**	**580.9**	**1200**	**1937**
ExMLDS2	143.19	781.94	880.64	12000	13000
SLEEC	313	1351	4660	8912	10000

Dataset	Prec@k	Proposed	Embedding Based			Sparsity Based	Tree Based		Others	
		ExMLDS1	DXML	SLEEC	LEML	PD-SPARSE	PFASTREXML	FASTXML	ONE-VS-ALL	DISMEC
Bibtex	P@1	63.38	63.69	**65.29**	62.54	61.29	63.46	63.42	62.62	-
	P@3	38.00	37.63	**39.60**	38.41	35.82	39.22	39.23	39.09	-
	P@5	27.64	27.71	**28.63**	28.21	25.74	29.14	28.86	28.79	-
Delicious	P@1	67.94	67.57	**68.10**	65.67	51.82	67.13	69.61	65.01	-
	P@3	61.35	61.15	61.78	60.55	44.18	62.33	**64.12**	58.88	-
	P@5	56.3	56.7	57.34	56.08	38.95	58.62	**59.27**	53.28	-
Eurlex	P@1	77.55	77.13	79.52	63.40	76.43	75.45	71.36	79.89	**82.40**
	P@3	64.18	64.21	64.27	50.35	60.37	62.70	59.90	66.01	**68.50**
	P@5	52.51	52.31	52.32	41.28	49.72	52.51	50.39	53.80	**57.70**
Mediamill	P@1	87.49	**88.71**	87.37	84.01	81.86	83.98	84.22	83.57	-
	P@3	**72.62**	71.65	72.6	67.20	62.52	67.37	67.33	65.60	-
	P@5	**58.46**	56.81	58.39	52.80	45.11	53.02	53.04	48.57	-
Delicious-200K	P@1	46.07	44.13	**47.50**	40.73	34.37	41.72	43.07	-	45.50
	P@3	41.15	39.88	**42.00**	37.71	29.48	37.83	38.66	-	38.70
	P@5	38.57	37.20	**39.20**	35.84	27.04	35.58	36.19	-	35.50

ExMLDS3 Performance with Missing Labels

Dataset	Prec@k	ExMLDS3	Sleec	Leml	Leml-Imc
Bibtex	P@1	**48.51**	30.5	35.98	41.23
	P@3	**28.43**	14.9	21.02	25.25
	P@5	**20.7**	9.81	15.50	18.56
Eurlex	P@1	**60.28**	51.4	26.22	39.24
	P@3	**44.87**	37.64	22.94	32.66
	P@5	**35.31**	29.62	19.02	26.54
rcv1v2	P@1	**81.67**	41.8	64.83	73.68
	P@3	**52.82**	17.48	42.56	48.56
	P@5	**37.74**	10.63	31.68	34.82

We hide randomly **80%** of the labels from training labels. We provide extra **YY' (original)** complete label-label correlation matrix along with masked **Y** to both *LEML-IMC* and *ExMLDS3*.

ExMLDS4 Performance (Joint Training Algorithm)

Dataset	Prec@k	Proposed	Embedding Based			Sparsity Based	
		ExMLDS4	AnnexML	Sleec	XML-CNN	Pd-Sparse	PPDsparse
AmazonCat-13K	P@1	93.05	93.55	90.53	**95.06**	89.31	92.72
	P@3	79.18	78.38	76.33	**79.86**	74.03	78.14
	P@5	64.54	63.32	61.52	**63.91**	60.11	63.41
Wiki10K-31K	P@1	**86.82**	86.50	85.88	84.06	77.71	-
	P@3	**74.30**	74.28	72.98	73.96	65.73	-
	P@5	63.68	**64.19**	62.70	64.11	55.39	-
Delicious-200K	P@1	47.70	46.66	**47.85**	-	34.37	45.05
	P@3	41.22	40.79	**42.21**	-	29.48	38.34
	P@5	37.98	37.64	**39.43**	-	27.04	34.90
WikiLSHTC-325K	P@1	62.15	63.36	54.83	-	61.26	**64.13**
	P@3	39.58	40.66	33.42	-	39.48	**42.10**
	P@5	29.10	29.79	23.85	-	28.79	**31.14**
Wikipedia-500K	P@1	62.27	**63.86**	58.39	59.85	-	-
	P@3	41.43	**42.69**	37.88	39.28	-	-
	P@5	31.42	**32.37**	28.21	29.31	-	-
Amazon-670K	P@1	41.47	42.08	35.05	-	**44.70**	43.04
	P@3	36.35	36.65	31.25	-	**39.70**	38.24
	P@5	32.43	32.76	28.56	-	**36.10**	34.94

Conclusions & Future Work

- Novel objective for learning *instance-label embeddings* for *multi-label classification*, that leverages the *word2vec* embedding method
- Proposed formulation can be optimized efficiently by *SPPMI* matrix factorization, making it's 10 times faster than *SLEEC*
- Proposed method is competitive compared to state-of-the-art multi-label learning methods in terms of prediction accuracy
- Proposed *SGNS* objective can jointly learn the *embeddings Z* and *regressor V* and obtain near state of art results on multiple datasets
- Proposed novel objective can incorporates side information, that is particularly useful for handling missing labels

Acknowledgement

- Prof. Purushottam Kar for allowing me to reuse some of his slides for extreme learning introduction

- Anonymous reviewers whose reviews help in improving the quality of the paper

- Microsoft Research Lab, Bangalore, School of Computing, University of Utah and Indian Institute of Technology, Kanpur for needed support and guidance

Main References

- **ExMLDS** : Rahul Wadbude and Vivek Gupta. *"Leveraging Distributional Semantics for Multi Label Leaning"*. AAAI 2019
- ***Word2vec :*** Tomas Mikolov and Ilya Sutskever. *"Distributed representations of words and phrases and their compositionality"* In: NIPS 2013
- **Doc2vec** : Quoc V Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents" In ICML 2014
- **SLEEC** : Kush Bhatiya and Himanshu Jain. *"Sparse Local Embeddings for Extreme Multi-label Classification"* , in *NIPS,* 2015.
- **FastXML** : Yashoteja Prabhu, and Manik Varma. "*A Fast, Accurate and Stable Tree-classifier for eXtreme Multi-label Learning"*, in *KDD,* 2014
- **PfastreXML** : Himanshu Jian and Yashoteja Prabhu. "*Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications*", in *KDD*, 2016.
- **PDSparse** : I. E. H. Yen and X. Huang. "*PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification"* in *ICML*, 2016.
- **DiSMEC** : R. Babbar, and B. Schölkopf. "*DiSMEC - Distributed Sparse Machines for Extreme Multi-label Classification"*, in *WSDM* 2017.
- **PPDSparse** : I. E. H. Yen and X. Huang. "PPDSparse: A Parallel Primal-Dual Sparse Method for Extreme Classification" in *KDD*, 2017.
- **AnnexML** : Yukihiro Tagami. *"Approximate Nearest Neighbor Search for Extreme Multi-label Classification", in KDD 2017*
- **Slice:** H. Jain and V. Balasubramanian, **"***Scalable linear extreme classifiers trained on 100 million labels for related searches", in WSDM 2019*
- **Parabel:** Y. Prabhu and A. Kag**, "***Partitioned label trees for extreme classification with application to dynamic search advertising", in WWW 2018*
- **SwiftML:** Y. Prabhu and A. Ka**g.** *"Extreme multi-label learning with label features for warm-start tagging, ranking and recommendation", in WSDM 2018*


Ask Your Query