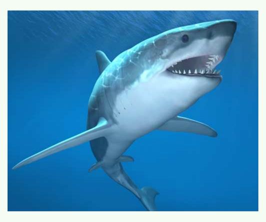
Cost-Sensitive Label Embedding for Multi-Label Classification

Kuan-Hao Huang and Hsuan-Tien Lin

CSIE Department, National Taiwan University

Multi-Label Classification

image $\mathbf{x}$				
tag	{dog, cat}	{dog}	{dog, cat, rabbit}	{shark}
label $\mathbf{y}$	$(1, 1, 0, 0)$	$(1, 0, 0, 0)$	$(1, 1, 1, 0)$	$(0, 0, 0, 1)$

- given training instances $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$
- learn a predictor h from $\mathcal{D}$
- let prediction $\tilde{\mathbf{y}} = h(\mathbf{x})$ be close to ground truth $\mathbf{y}$

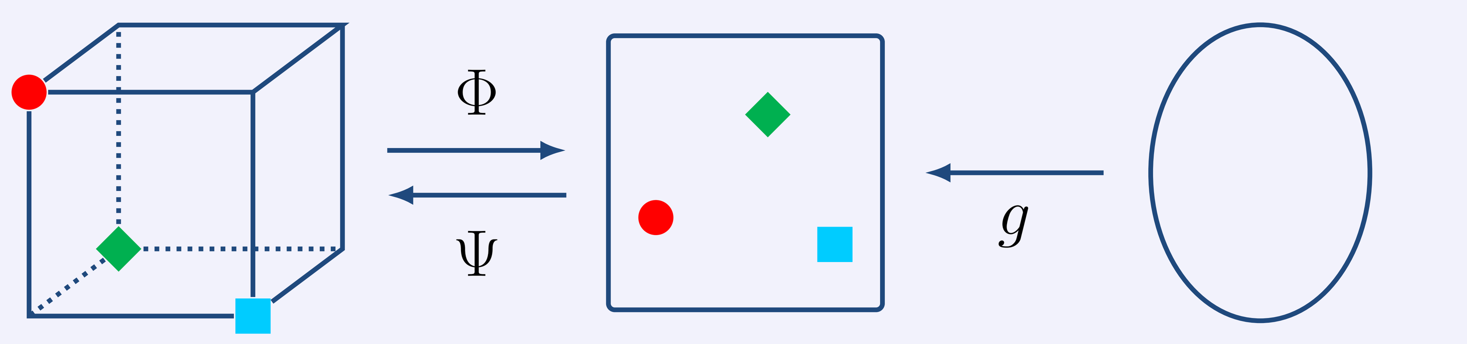
Cost-Sensitivity

- different applications use different evaluation criteria
- Hamming loss, Ranking loss, F1 score, Accuracy score, etc.
- take the evaluation criterion as the additional input
- make predictions toward the criterion

Cost-Sensitive Multi-Label Classification

- given $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ and cost function $c(\cdot, \cdot)$
- learn a predictor h from both $\mathcal{D}$ and $c(\cdot, \cdot)$
- make prediction $\tilde{\mathbf{y}} = h(\mathbf{x})$ such that $c(\mathbf{y}, \tilde{\mathbf{y}})$ is small

Label Embedding Approach

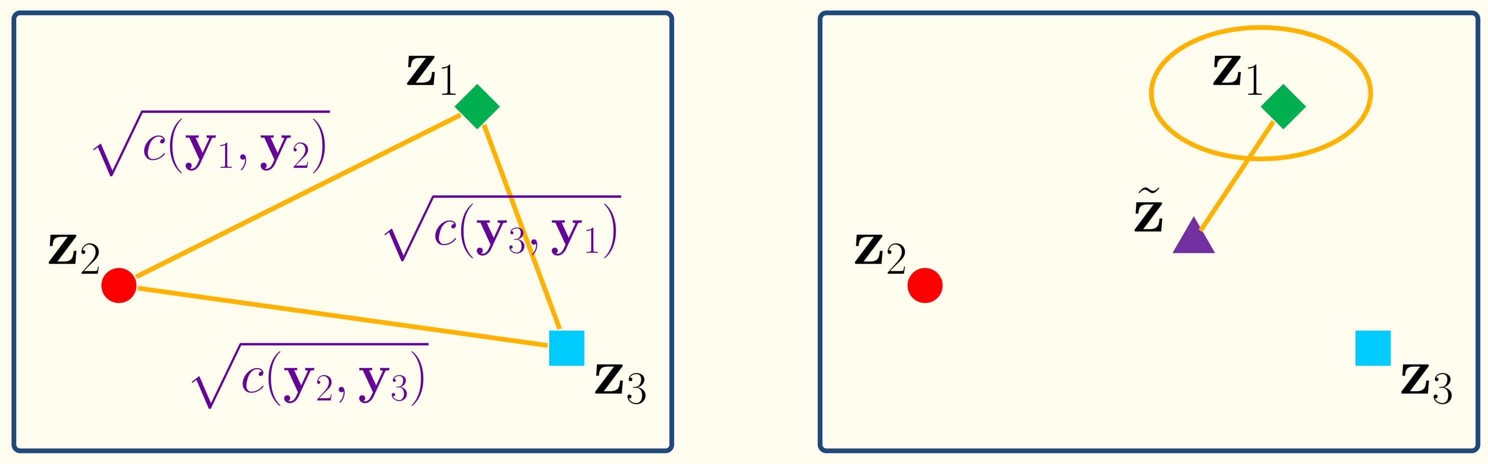


- embedding function Φ: label vector $\mathbf{y} \to$ embedded vector $\mathbf{z}$
- learn a regressor g from $\{(\mathbf{x}^{(n)}, \mathbf{z}^{(n)})\}_{n=1}^{N}$
- for testing instance $\mathbf{x}$, predicted embedded vector $\tilde{\mathbf{z}} = g(\mathbf{x})$
- decoding function Ψ: predicted embedded vector $\tilde{\mathbf{z}} \to$ predicted label vector $\tilde{\mathbf{y}}$

Related Works

- cost-sensitivity: PCC and CFT [Li et al., 2014; Dembczynski et al., 2010]
- label embedding: PLST, FaIE, RAkEL, etc. [Tai et al., 2012; Lin et al., 2014; Tsoumakas et al., 2011]
- cost-sensitivity + label embedding: no existing works

Proposed Cost-Sensitive Label Embedding (CLEMS)



Embedding

- cost information $\Leftrightarrow$ distances between embedded vectors
- higher (lower) cost $c(\mathbf{y}_i, \mathbf{y}_j) \Leftrightarrow$ larger (smaller) distance $d(\mathbf{z}_i, \mathbf{z}_j)$
- let $d(\mathbf{z}_i, \mathbf{z}_j) \approx \sqrt{c(\mathbf{y}_i, \mathbf{y}_j)}$ by multidimensional scaling (manifold learning)
- embedding function Φ: mapping $\mathbf{y}_i \to \mathbf{z}_i$

Decoding

- for testing instance $\mathbf{x}$, predicted embedded vector $\tilde{\mathbf{z}} = g(\mathbf{x})$
- find nearest embedded vector $\mathbf{z}_q$ of $\tilde{\mathbf{z}}$ from a candidate set
- cost-sensitive prediction $\tilde{\mathbf{y}} = \mathbf{y}_q$
- decoding function Ψ: nearest neighbor + inverse mapping Φ^{-1}

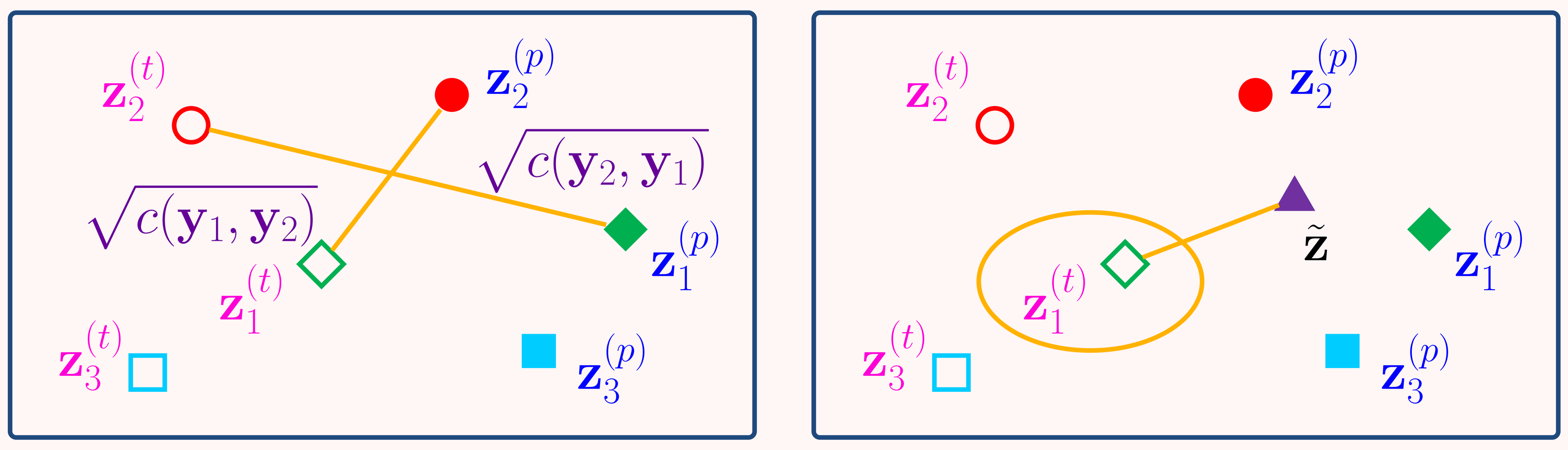
Theoretical Explanation

- for ground truth $\mathbf{y}$, prediction $\tilde{\mathbf{y}}$, the cost $c(\mathbf{y}, \tilde{\mathbf{y}})$ has upper bound

$$c(\mathbf{y}, \tilde{\mathbf{y}}) \leq 5\Bigg(\underbrace{\left(d(\mathbf{z}, \mathbf{z}_q) - \sqrt{c(\mathbf{y}, \tilde{\mathbf{y}})}\right)^2}_{\text{embedding error}} + \underbrace{d(\mathbf{z}, \tilde{\mathbf{z}})^2}_{\text{regression error}}\Bigg)$$

- embedding error $\to$ multidimensional scaling
- regression error $\to$ regressor g

Mirroring Trick for Asymmetric Cost Function



- asymmetric cost vs. symmetric distance
- $c(\mathbf{y}_i, \mathbf{y}_j) \neq c(\mathbf{y}_j, \mathbf{y}_i)$ vs. $d(\mathbf{z}_i, \mathbf{z}_j)$
- two roles of $\mathbf{z}$: ground truth role $\mathbf{z}_i^{(t)}$ and prediction role $\mathbf{z}_i^{(p)}$
- predict $\mathbf{y}_i$ as $\mathbf{y}_j \Rightarrow \sqrt{c(\mathbf{y}_i, \mathbf{y}_j)}$ for $\mathbf{z}_i^{(t)}$ and $\mathbf{z}_j^{(p)}$
- predict $\mathbf{y}_j$ as $\mathbf{y}_i \Rightarrow \sqrt{c(\mathbf{y}_j, \mathbf{y}_i)}$ for $\mathbf{z}_i^{(p)}$ and $\mathbf{z}_j^{(t)}$
- learn regressor g from $\mathbf{z}_i^{(p)}, \mathbf{z}_2^{(p)}, ..., \mathbf{z}_L^{(p)}$
- find nearest embedded vector of $\tilde{\mathbf{z}}$ from $\mathbf{z}_1^{(t)}, \mathbf{z}_2^{(t)}, ..., \mathbf{z}_L^{(t)}$

Experimental Results

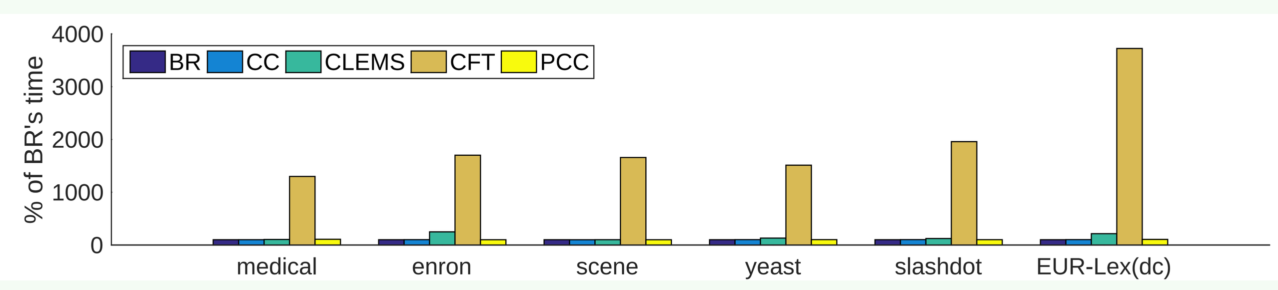


Figure: Average training time

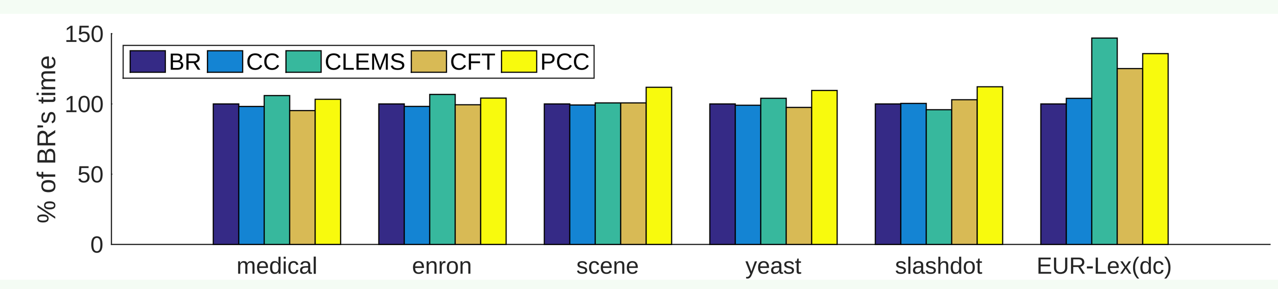


Figure: Average predicting time

Table: Performance across different evaluation criteria

data	F1 score (↑)				Accuracy score (↑)				Rank loss (↓)				Composition loss (↓)			
	BR	CLEMS	CFT	PCC	BR	CLEMS	CFT	PCC	BR	CLEMS	CFT	PCC	BR	CLEMS	CFT	PCC
emot.	0.596	**0.676**	0.640	0.643	0.523	**0.589**	0.557	–	1.764	1.484	1.563	**1.467**	0.352	**0.271**	0.324	–
scene	0.577	**0.770**	0.703	0.745	0.568	**0.760**	0.656	–	1.169	0.672	0.723	**0.645**	0.866	**0.578**	0.776	–
yeast	0.611	**0.671**	0.649	0.614	0.503	**0.568**	0.543	–	9.673	**8.302**	8.566	8.469	1.345	**1.308**	1.335	–
birds	0.569	**0.677**	0.601	0.636	0.551	**0.642**	0.586	–	6.845	4.886	4.908	**3.660**	0.656	**0.563**	0.607	–
med.	0.517	**0.814**	0.635	0.573	0.496	**0.786**	0.613	–	13.78	5.170	5.811	**4.234**	0.562	**0.289**	0.438	–
enron	0.543	**0.606**	0.557	0.542	0.433	**0.491**	0.448	–	44.83	29.40	26.64	**25.11**	0.688	**0.659**	0.677	–
lang.	0.160	**0.375**	0.168	0.247	0.159	**0.327**	0.164	–	43.46	31.03	34.16	**19.11**	0.919	**0.734**	0.910	–
arts	0.167	**0.492**	0.334	0.349	0.156	**0.451**	0.281	–	17.22	9.865	10.07	**8.467**	1.117	**0.815**	1.001	–
EUR.	0.417	**0.670**	0.456	0.483	0.411	**0.650**	0.450	–	168.3	89.52	129.5	**43.28**	0.593	**0.344**	0.552	–

- **cost-sensitivity**: CLEMS $=$ CFT $>$ PCC
- **performance**: CLEMS $\approx$ PCC $>$ CFT
- **speed**: CLEMS $\approx$ PCC $>$ CFT

CLEMS is a promising algorithm!