

# Cost-Sensitive Label Embedding for Multi-Label Classification

**Kuan-Hao Huang**

Advisor: Hsuan-Tien Lin

Department of Computer Science & Information Engineering  
National Taiwan University







TAAI, November 27, 2016

# Multi-Label Classification (MLC)

## Multi-label classification

- ▶ an **extension** of the **multiclass classification**
- ▶ allow instance with **multiple** associated classes

Example: image tag with (dog, cat, rabbit, shark)

image				
tag	{ dog, cat }	{ dog }	{ dog, cat, rabbit }	{ shark }
label	(1, 1, 0, 0)	(1, 0, 0, 0)	(1, 1, 1, 0)	(0, 0, 0, 1)

# Multi-Label Classification (MLC)

## Notation

- ▶ feature vector (image):  $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$
- ▶ label vector (tag):  $\mathbf{y} \in \mathcal{Y} \subseteq \{0, 1\}^K$

## Multi-label classification (MLC)

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

## Evaluation of closeness

- ▶ **cost function**  $c(\mathbf{y}, \tilde{\mathbf{y}})$ : the penalty of predicting  $\mathbf{y}$  as  $\tilde{\mathbf{y}}$
- ▶ Hamming loss, 0/1 loss, Rank loss, F1 score(loss), Accuracy score(loss)

# Cost-Sensitive Multi-Label Classification (CSMLC)

## Notation

- ▶ feature vector (image):  $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$
- ▶ label vector (tag):  $\mathbf{y} \in \mathcal{Y} \subseteq \{0, 1\}^K$

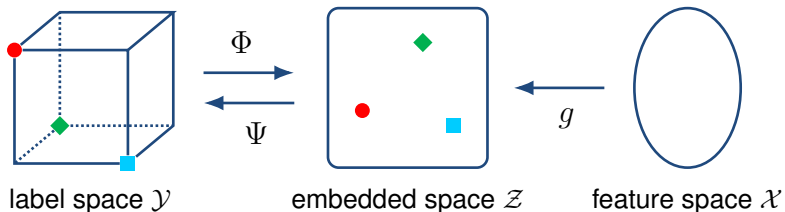
## Cost-sensitive multi-label classification (CSMLC)

- ▶ given training instances  $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$  and **cost function**  $c$
- ▶ learn a **predictor**  $h$  from **both**  $\mathcal{D}$  **and**  $c$
- ▶ for testing instance  $(\mathbf{x}, \mathbf{y})$ , prediction  $\tilde{\mathbf{y}} = h(\mathbf{x})$
- ▶ let the prediction  $\tilde{\mathbf{y}}$  is close to ground truth  $\mathbf{y}$

## Evaluation of closeness

- ▶ **cost function**  $c(\mathbf{y}, \tilde{\mathbf{y}})$ : the penalty of predicting  $\mathbf{y}$  as  $\tilde{\mathbf{y}}$
- ▶ Hamming loss, 0/1 loss, Rank loss, F1 score(loss), Accuracy score(loss)

# Label Embedding



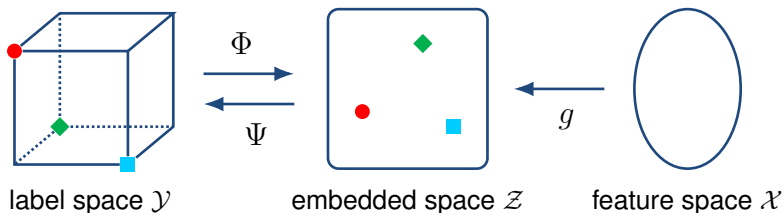
## Training stage

- ▶ **embedding function  $\Phi$** : label vector  $\mathbf{y} \rightarrow$  embedded vector  $\mathbf{z}$
- ▶ train a regressor  $g$  from  $\{(\mathbf{x}^{(n)}, \mathbf{z}^{(n)})\}_{n=1}^N$

## Predicting stage

- ▶ for testing instance  $\mathbf{x}$ , predicted embedded vector  $\tilde{\mathbf{z}} = g(\mathbf{x})$
- ▶ **decoding function  $\Psi$** : predicted embedded vector  $\tilde{\mathbf{z}} \rightarrow$  predicted label vector  $\tilde{\mathbf{y}}$

# Cost-Sensitive Label Embedding



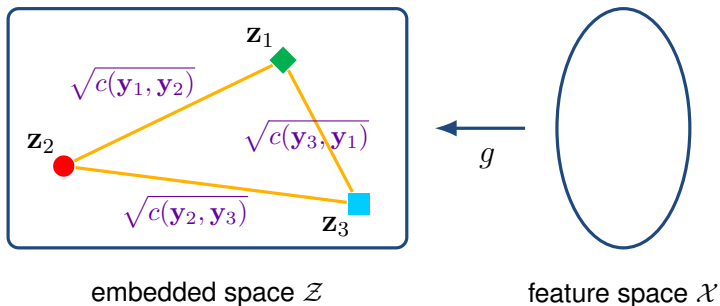
## Existing works

- ▶ **label embedding**: PLST, FaIE, RA<sub>k</sub>EL, ECC-based [Tai et al., 2012; Lin et al., 2014; Tsoumakas et al., 2011; Ferng et al., 2013]
- ▶ **cost-sensitivity**: CFT, PCC [Li et al., 2014; Dembczynski et al., 2010]
- ▶ **cost-sensitivity + label embedding**: no existing works

## Cost-sensitive label embedding

- ▶ consider **cost function**  $c$  when designing **embedding function**  $\Phi$  and **decoding function**  $\Psi$  (cost-sensitive embedded vectors  $z$ )

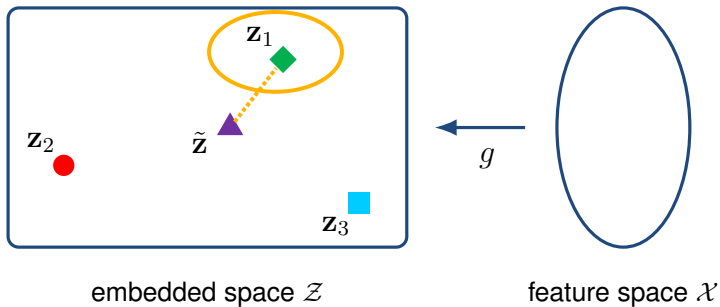
# Cost-Sensitive Label Embedding (Training)



## Training stage

- ▶ distances between embedded vectors  $\Leftrightarrow$  cost information
- ▶ larger (smaller) distance  $d(\mathbf{z}_i, \mathbf{z}_j) \Leftrightarrow$  higher (lower) cost  $c(\mathbf{y}_i, \mathbf{y}_j)$
- ▶  $d(\mathbf{z}_i, \mathbf{z}_j) \approx \sqrt{c(\mathbf{y}_i, \mathbf{y}_j)}$

# Cost-Sensitive Label Embedding (Predicting)



## Predicting stage

- ▶ for testing instance  $\mathbf{x}$ , predicted embedded vector  $\tilde{\mathbf{z}} = g(\mathbf{x})$
- ▶ find **nearest embedded vector**  $\mathbf{z}_q$  of  $\tilde{\mathbf{z}}$
- ▶ cost-sensitive prediction  $\tilde{\mathbf{y}} = \mathbf{y}_q$



# Cost-Sensitive Label Embedding (Theorem)

## Theorem

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

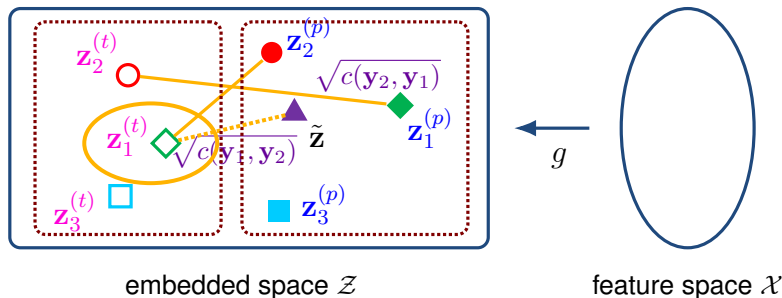
## Optimization

- ▶ **embedding error**  $\rightarrow$  multidimensional scaling (manifold learning)
- ▶ **regression error**  $\rightarrow$  regressor  $g$

## Challenge

- ▶ **asymmetric cost function vs. symmetric distance?**
- ▶  $c(\mathbf{y}_i, \mathbf{y}_j) \neq c(\mathbf{y}_j, \mathbf{y}_i)$  vs.  $d(\mathbf{z}_i, \mathbf{z}_j)$

# Mirroring Trick



- ▶ two roles of  $y$ : **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  $\mathbf{z}_1^{(p)}, \mathbf{z}_2^{(p)}, \dots, \mathbf{z}_L^{(p)}$
- ▶ find **nearest embedded vector** of  $\tilde{\mathbf{z}}$  from  $\mathbf{z}_1^{(t)}, \mathbf{z}_2^{(t)}, \dots, \mathbf{z}_L^{(t)}$

# Candidate Set

## Challenge

- ▶ label vector  $\mathbf{y} \in \mathcal{Y} \subseteq \{0, 1\}^K$
- ▶  $2^K$  possible label vectors (too many)
- ▶ what is the important(useful) label vectors?

## Candidate Set

- ▶ consider a **candidate set**  $\mathcal{S}$  instead of  $\mathcal{Y}$
- ▶ only label vectors in  $\mathcal{S}$  are embedded
- ▶  $\mathcal{S}_{train}$  (all the label vectors in training set) is a reasonable choice

# Cost-Sensitive Label Embedding with Multidimensional Scaling

## Training stage of CLEMS

- ▶ given training instances  $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$  and cost function  $c$
- ▶ determine the candidate set  $\mathcal{S}$
- ▶ find  $\mathbf{z}_i^{(t)}$  and  $\mathbf{z}_i^{(p)}$  for all  $\mathbf{y}_i \in \mathcal{S}$  by **multidimensional scaling**
- ▶  $\Phi: \mathbf{y}_i \rightarrow \mathbf{z}_i^{(p)}$
- ▶ train a multi-target regressor  $g$  from  $\{(\mathbf{x}^{(n)}, \Phi(\mathbf{y}^{(n)}))\}_{n=1}^N$

## Predicting stage of CLEMS

- ▶ given the testing instance  $(\mathbf{x}, \mathbf{y})$
- ▶  $\Psi(\cdot) = \Phi^{-1}(\text{nearest neighbor}) = \Phi^{-1}(\text{argmin } d(\mathbf{z}_i^{(t)}, \cdot))$
- ▶ obtain the predicted embedded vector by  $\tilde{\mathbf{z}} = g(\mathbf{x})$
- ▶ prediction  $\tilde{\mathbf{y}} = \Psi(\tilde{\mathbf{z}})$

# Experiments

## Lists of experiments

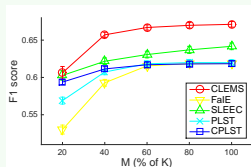
- ▶ **comparison with label embedding algorithms**
  - ▶ LSDR algorithms ( $\dim \mathcal{Z} < \dim \mathcal{Y}$ )
  - ▶ LSDE algorithms ( $\dim \mathcal{Z} \geq \dim \mathcal{Y}$ )
- ▶ **comparison with cost-sensitive algorithms**
  - ▶ condensed filter tree (CFT) [Li et al., 2014]

## Settings

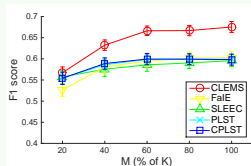
- ▶ 50% for training, 25% for validation, and 25% for testing
- ▶ base learner: random forest classifier or random forest regressor
- ▶ evaluation criteria
  - ▶ **F1 score**  $\frac{2\|\mathbf{y} \cap \tilde{\mathbf{y}}\|_1}{\|\mathbf{y}\|_1 + \|\tilde{\mathbf{y}}\|_1}$  ( $\uparrow$ )
  - ▶ **Accuracy score**  $\frac{\|\mathbf{y} \cap \tilde{\mathbf{y}}\|_1}{\|\mathbf{y} \cup \tilde{\mathbf{y}}\|_1}$  ( $\uparrow$ )
  - ▶ **Rank loss**  $\sum_{\mathbf{y}[i] > \mathbf{y}[j]} (\mathbb{I}[\tilde{\mathbf{y}}[i] < \tilde{\mathbf{y}}[j]] + \frac{1}{2} \mathbb{I}[\tilde{\mathbf{y}}[i] = \tilde{\mathbf{y}}[j]])$  ( $\downarrow$ )
- ▶ average results of 20 experiments

# Comparison with LSDR Algorithms

## F1 score ( $\uparrow$ )

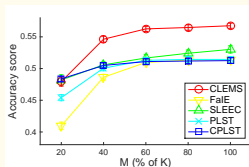


yeast

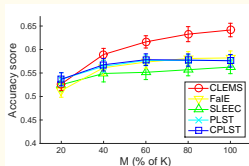


birds

## Accuracy score ( $\uparrow$ )

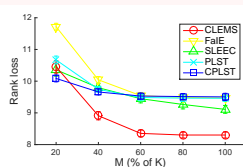


yeast

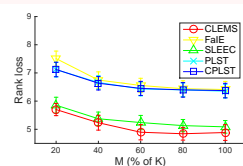


birds

## Rank loss ( $\downarrow$ )



yeast

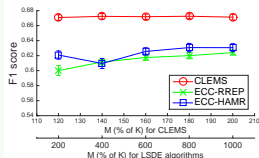


birds

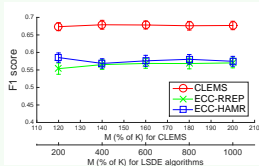
CLEMS outperforms LSDR algorithms on all the cost functions!

# Comparison with LSDE Algorithms

## F1 score ( $\uparrow$ )

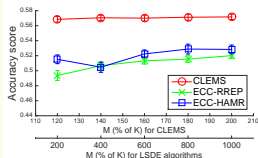


yeast

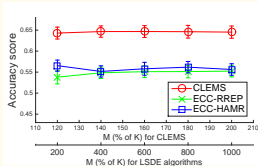


birds

## Accuracy score ( $\uparrow$ )

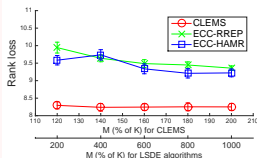


yeast

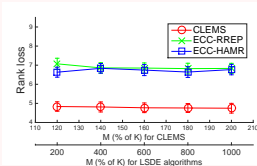


birds

## Rank loss ( $\downarrow$ )



yeast



birds

CLEMS outperforms LSDE algorithms on all the cost functions!

# Comparison with Cost-sensitive Algorithms

Table: Comparison with CFT

	F1 score ( $\uparrow$ )		Accuracy score ( $\uparrow$ )		Rank loss ( $\downarrow$ )	
	CFT	CLEMS	CFT	CLEMS	CFT	CLEMS
emo.	0.640	<b>0.676</b>	0.557	<b>0.589</b>	1.563	<b>1.484</b>
scene	0.703	<b>0.770</b>	0.656	<b>0.760</b>	0.723	<b>0.672</b>
yeast	0.649	<b>0.671</b>	0.543	<b>0.568</b>	8.566	<b>8.302</b>
birds	0.601	<b>0.674</b>	0.586	<b>0.642</b>	4.908	<b>4.886</b>
med.	0.635	<b>0.814</b>	0.613	<b>0.786</b>	5.811	<b>5.170</b>
enron	0.557	<b>0.606</b>	0.448	<b>0.491</b>	<b>26.64</b>	29.40
CAL.	0.371	<b>0.419</b>	0.237	<b>0.273</b>	<b>1120.8</b>	1247.9
EUR.	0.456	<b>0.670</b>	0.450	<b>0.650</b>	129.53	<b>89.52</b>

CLEMS outperforms CFT in most of the cases!



# Conclusion

- ▶ **algorithm design:** cost-sensitive label embedding algorithm (CLEMS)
  - ▶ mapping and nearest neighbor view for the efficient decoding function
  - ▶ embed the cost information in distance by multidimensional scaling
  - ▶ mirroring trick for the asymmetric cost function
  - ▶ candidate set to reduce the computational burden
- ▶ **theoretical guarantee:**
  - ▶ prove the upper bound of the predicted cost for CLEMS
- ▶ **empirical performance:**
  - ▶ CLEMS outperforms the existing LSDR and LSDE algorithms
  - ▶ CLEMS is better than the state-of-the-art cost-sensitive algorithms

**Thank you! Any question?**