Label Space Coding for Multi-label Classification

Hsuan-Tien Lin

National Taiwan University

RIKEN AIP Center, 08/30/2019

joint works with
Farbound Tai (MLD Workshop 2010, NC Journal 2012) &
Yao-Nan Chen (NeurIPS Conference 2012) &
Kuan-Hao Huang (ECML Conference ML Journal Track 2017)

Which Fruit**s**?

?: {orange, strawberry, kiwi}

apple

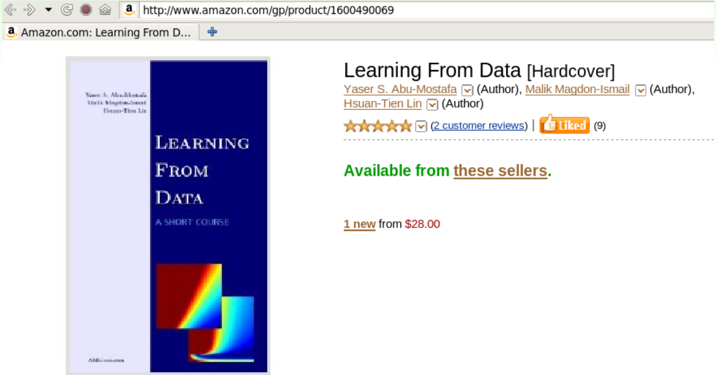
orange

strawberry

kiwi

multi-label classification:
classify input to **multiple (or no)** categories

What **Tags**?



?: {machine learning, ~~data structure~~, data mining, ~~object oriented programming~~, artificial intelligence, ~~compiler~~, ~~architecture~~, ~~chemistry~~, textbook, ~~children book~~, ~~. . . etc.~~ }

another **multi-label** classification problem:
tagging input to multiple categories

Binary Relevance: Multi-label Classification via Yes/No

Binary Classification

{yes, no}

Multi-label w/ L classes: L yes/no questions

machine learning (Y), data structure (N), data mining (Y), OOP (N), AI (Y), compiler (N), architecture (N), chemistry (N), textbook (Y), children book (N), *etc.*

- **Binary Relevance** approach: transformation to **multiple isolated binary classification**
- disadvantages:
  - **isolation**—hidden relations not exploited (e.g. ML and DM highly correlated, ML subset of AI, textbook & children book disjoint)
  - **unbalanced**—few yes, many no

Binary Relevance: simple (& good) benchmark with known disadvantages

Multi-label Classification Setup

Given

N examples (input $\mathbf{x}_n$, label-set $\mathcal{Y}_n$) $\in \mathcal{X} \times 2^{\{1,2,\cdots L\}}$

- `fruits`: $\mathcal{X} =$ encoding(pictures), $\mathcal{Y}_n \subseteq \{1, 2, \cdots, 4\}$
- `tags`: $\mathcal{X} =$ encoding(merchandise), $\mathcal{Y}_n \subseteq \{1, 2, \cdots, L\}$

Goal

a multi-label classifier $g(\mathbf{x})$ that **closely predicts** the label-set $\mathcal{Y}$ associated with some **unseen** inputs $\mathbf{x}$ (by exploiting hidden relations/combinations between labels)

- **Hamming loss**: averaged symmetric difference $\frac{1}{L}|g(\mathbf{x}) \triangle \mathcal{Y}|$

multi-label classification: hot and important

From Label-set to Coding View

	label set	apple	orange	strawberry	**binary code**
	$\mathcal{Y}_1 = \{\text{o}\}$	0 (N)	1 (Y)	0 (N)	$\mathbf{y}_1 = [0, 1, 0]$
	$\mathcal{Y}_2 = \{\text{a, o}\}$	1 (Y)	1 (Y)	0 (N)	$\mathbf{y}_2 = [1, 1, 0]$
	$\mathcal{Y}_3 = \{\text{a, s}\}$	1 (Y)	0 (N)	1 (Y)	$\mathbf{y}_3 = [1, 0, 1]$
	$\mathcal{Y}_4 = \{\text{o}\}$	0 (N)	1 (Y)	0 (N)	$\mathbf{y}_4 = [0, 1, 0]$
	$\mathcal{Y}_5 = \{\}$	0 (N)	0 (N)	0 (N)	$\mathbf{y}_5 = [0, 0, 0]$

subset $\mathcal{Y}$ of $2^{\{1,2,\cdots,L\}}$ $\Leftrightarrow$ **length-L binary code $\mathbf{y}$**

Existing Approach: Compressive Sensing

General Compressive Sensing

sparse (many 0) binary vectors $\mathbf{y} \in \{0, 1\}^L$ can be **robustly compressed** by projecting to $M \ll L$ basis vectors $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_M\}$

Compressive Sensing for Multi-label Classification (Hsu et al., 2009)

1. **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with some M by L random matrix $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_M]^T$
2. **learn**: get regression function $\mathbf{r}(\mathbf{x})$ from $\mathbf{x}_n$ to $\mathbf{c}_n$
3. **decode**: $g(\mathbf{x})$ = find closest sparse binary vector to $\mathbf{P}^T\mathbf{r}(\mathbf{x})$

Compressive Sensing:

- efficient in training: random projection w/ $M \ll L$
- inefficient in testing: time-consuming decoding

bettr projection? faster decoding?

Our Contributions

Compression Coding & Learnable-Compression Coding

A Novel Approach for Label Space Compression

- algorithmic: first known algorithm for **feature-aware dimension reduction** with **fast decoding**
- theoretical: justification for **best learnable projection**
- practical: **consistently better performance** than compressive sensing (& binary relevance)

will now introduce the key ideas behind the approach

Faster Decoding: Round-based

Compressive Sensing Revisited

- **decode**: $g(\mathbf{x})$ = find closest sparse binary vector to $\tilde{\mathbf{y}} = \mathbf{P}^T\mathbf{r}(\mathbf{x})$

For any given "intermediate prediction" (real-valued vector) $\tilde{\mathbf{y}}$,

- find closest sparse binary vector to $\tilde{\mathbf{y}}$: slow optimization of ℓ_1-regularized objective
- find closest any binary vector to $\tilde{\mathbf{y}}$: fast

$$g(\mathbf{x}) = \text{round}(\mathbf{y})$$

round-based decoding: simple & faster alternative

Better Projection: Principal Directions

Compressive Sensing Revisited

- **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with some M by L random matrix $\mathbf{P}$

- random projection: arbitrary directions
- best projection: principal directions

principal directions: best approximation to desired output $\mathbf{y}_n$ during compression (**why?**)

Novel Theoretical Guarantee

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

If $g(\mathbf{x}) = round(\mathbf{P}^T \mathbf{r}(\mathbf{x}))$,

$$\underbrace{\frac{1}{L}|g(\mathbf{x}) \triangle \mathcal{Y}|}_{\textit{Hamming loss}} \leq const \cdot \left(\underbrace{\|\mathbf{r}(\mathbf{x}) - \overbrace{\mathbf{Py}}^{\mathbf{c}}\|^2}_{\textit{learn}} + \underbrace{\|\mathbf{y} - \mathbf{P}^T \overbrace{\mathbf{Py}}^{\mathbf{c}}\|^2}_{\textit{compress}} \right)$$

- $\|\mathbf{r}(\mathbf{x}) - \mathbf{c}\|^2$: prediction error from input to codeword
- $\|\mathbf{y} - \mathbf{P}^T\mathbf{c}\|^2$: encoding error from desired output to codeword

principal directions: best approximation to desired output $\mathbf{y}_n$ during compression (**indeed**)

Proposed Approach 1: Principal Label Space Transform

From Compressive Sensing to PLST

1. **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with the M by L principal matrix $\mathbf{P}$
2. **learn**: get regression function $\mathbf{r}(\mathbf{x})$ from $\mathbf{x}_n$ to $\mathbf{c}_n$
3. **decode**: $g(\mathbf{x}) = \text{round}(\mathbf{P}^T\mathbf{r}(\mathbf{x}))$

- principal directions: via Principal Component Analysis on $\{\mathbf{y}_n\}_{n=1}^{N}$
- physical meaning behind $\mathbf{p}_m$: key (linear) label correlations

PLST: improving CS by projecting to **key correlations**

Theoretical Guarantee of PLST Revisited

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

If $g(\mathbf{x}) = \mathit{round}(\mathbf{P}^T\mathbf{r}(\mathbf{x}))$,

$$\underbrace{\frac{1}{L}|g(\mathbf{x}) \triangle \mathcal{Y}|}_{\text{Hamming loss}} \le \mathit{const} \cdot \left(\underbrace{\|\mathbf{r}(\mathbf{x}) - \overbrace{\mathbf{P}\mathbf{y}}^{\mathbf{c}}\|^2}_{\text{learn}} + \underbrace{\|\mathbf{y} - \mathbf{P}^T \overbrace{\mathbf{P}\mathbf{y}}^{\mathbf{c}}\|^2}_{\text{compress}} \right)$$

- $\|\mathbf{y} - \mathbf{P}^T\mathbf{c}\|^2$: encoding error, minimized during encoding
- $\|\mathbf{r}(\mathbf{x}) - \mathbf{c}\|^2$: prediction error, minimized during learning
- but good encoding may not be easy to learn; vice versa

PLST: minimize two errors separately (**sub-optimal**)
(*can we do better by minimizing jointly?*)

Proposed Approach 2: Conditional Principal Label Space Transform

can we do better by minimizing jointly?

Yes and easy for ridge regression (closed-form solution)

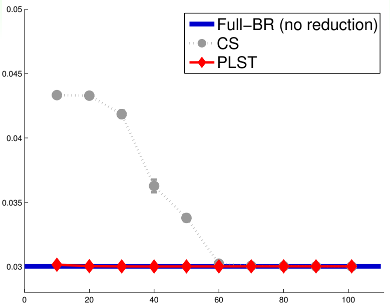
From PLST to CPLST

1. **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with the M by L conditional principal matrix $\mathbf{P}$
2. **learn**: get regression function $\mathbf{r}(\mathbf{x})$ from $\mathbf{x}_n$ to $\mathbf{c}_n$, ideally using ridge regression
3. **decode**: $g(\mathbf{x}) = \text{round}(\mathbf{P}^T\mathbf{r}(\mathbf{x}))$

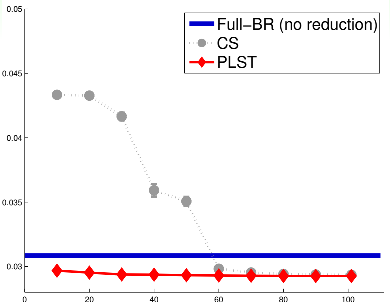
- conditional principal directions: top eigenvectors of $\mathbf{Y}^T\mathbf{X}\mathbf{X}^\dagger\mathbf{Y}$, key (linear) label correlations that are "easy to learn"

CPLST: project to **key learnable correlations**
—can also pair with **kernel regression (non-linear)**

Hamming Loss Comparison: Full-BR, PLST & CS



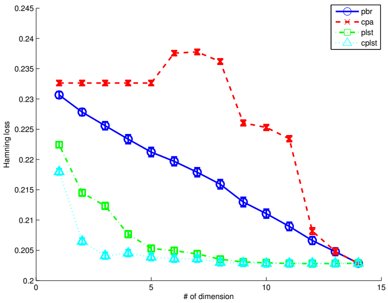
mediamill (Linear Regression)



mediamill (Decision Tree)

- PLST better than Full-BR: fewer dimensions, similar (or **better**) performance
- PLST better than CS: faster, **better** performance
- similar findings across **data sets** and **regression algorithms**

Hamming Loss Comparison: PLST & CPLST



yeast (Linear Regression)

- CPLST better than PLST: better performance across all dimensions
- similar findings across **data sets** and **regression algorithms**

Multi-label Classification Setup Revisited

Given

N examples (input $\mathbf{x}_n$, label-set $\mathcal{Y}_n$) $\in \mathcal{X} \times 2^{\{1,2,\cdots L\}}$

- `fruits`: $\mathcal{X} =$ encoding(pictures), $\mathcal{Y}_n \subseteq \{1, 2, \cdots, 4\}$
- `tags`: $\mathcal{X} =$ encoding(merchandise), $\mathcal{Y}_n \subseteq \{1, 2, \cdots, L\}$

Goal

a multi-label classifier $g(\mathbf{x})$ that **closely predicts** the label-set $\mathcal{Y}$ associated with some **unseen** inputs $\mathbf{x}$

- **Hamming loss**: averaged symmetric difference $\frac{1}{L}|g(\mathbf{x}) \bigtriangleup \mathcal{Y}|$

next: going beyond Hamming loss

Cost Functions for Multi-label Classification

Goal

a multi-label classifier $g(\mathbf{x})$ that **closely predicts** the label-set $\mathcal{Y}$ associated with some **unseen** inputs $\mathbf{x}$

- **Hamming loss**: averaged symmetric difference $\frac{1}{L}|g(\mathbf{x}) \triangle \mathcal{Y}|$

Other Evaluation of Closeness

- cost function $c(\mathbf{y}, \tilde{\mathbf{y}})$: the penalty of predicting $\mathbf{y}$ as $\tilde{\mathbf{y}}$
- e.g. 0/1 loss: strict match of $\tilde{\mathbf{y}}$ to $\mathbf{y}$
- e.g. F1 cost: 1 - geometric mean of precision & recall of $\tilde{\mathbf{y}}$ w.r.t. $\mathbf{y}$

Cost-Sensitive Multi-Label Classification (CSMLC)

Given

N examples (input $\mathbf{x}_n$, label-set $\mathcal{Y}_n$) $\in \mathcal{X} \times 2^{\{1,2,\cdots L\}}$

- `fruits`: $\mathcal{X} = \text{encoding(pictures)}$, $\mathcal{Y}_n \subseteq \{1, 2, \cdots, 4\}$
- `tags`: $\mathcal{X} = \text{encoding(merchandise)}$, $\mathcal{Y}_n \subseteq \{1, 2, \cdots, L\}$

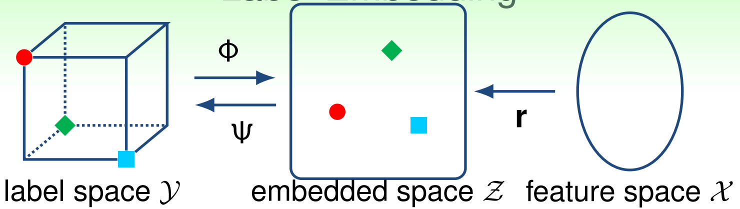
and desired cost function $c(\mathbf{y}, \tilde{\mathbf{y}})$

Goal

a multi-label classifier $g(\mathbf{x})$ that **closely predicts** the label-set-vector $\mathbf{y}$ associated with some **unseen** inputs $\mathbf{x}$—i.e. low $c(\mathbf{y}, g(\mathbf{x}))$.

next: label space coding for CSMLC

Label Embedding



Training Stage

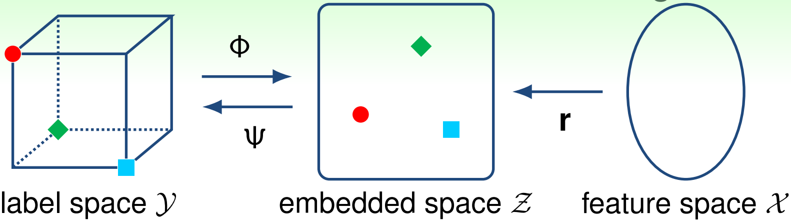
- embedding function Φ: label vector $\mathbf{y} \to$ embedded vector $\mathbf{z}$
- learn a regressor $\mathbf{r}$ from $\{(\mathbf{x}_n, \mathbf{z}_n)\}_{n=1}^{N}$

Predicting Stage

- for testing instance $\mathbf{x}$, predicted embedded vector $\tilde{\mathbf{z}} = \mathbf{r}(\mathbf{x})$
- decoding function Ψ: $\tilde{\mathbf{z}} \to$ predicted label vector $\tilde{\mathbf{y}}$

(C)PLST: linear projection embedding
+ round-based decoding

Cost-Sensitive Label Embedding



Existing Works

- **label embedding**: PLST, CPLST, FaIE, RA*k*EL, ECC-based [Tai et al., 2012; Chen 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 Φ and decoding function Ψ (cost-sensitive embedded vectors $\mathbf{z}$)

Our Contributions

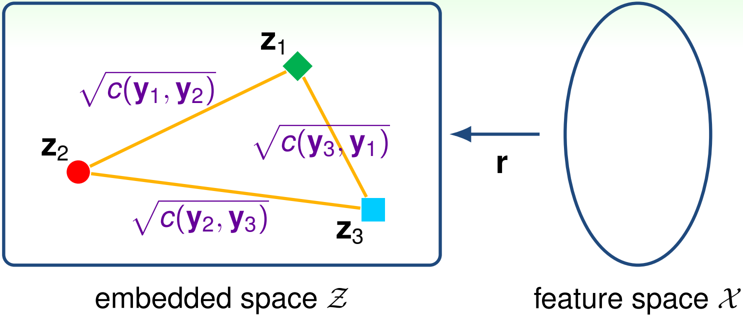
Cost-sensitive Coding

A Novel Approach for Label Space Compression

- algorithmic: first known algorithm for **cost-sensitive dimension reduction**
- theoretical: justification for **cost-sensitive label embedding**
- practical: **consistently better performance** than CPLST across different costs

will now introduce the key ideas behind the approach

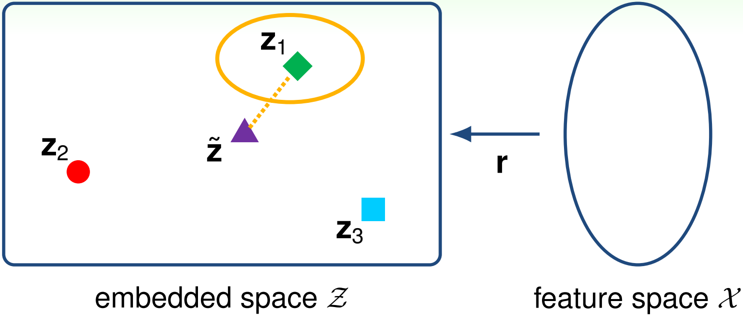
Cost-Sensitive Embedding



Training Stage

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

Cost-Sensitive Decoding



Predicting Stage

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

Theoretical Explanation

Theorem (Huang and Lin, 2017)

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

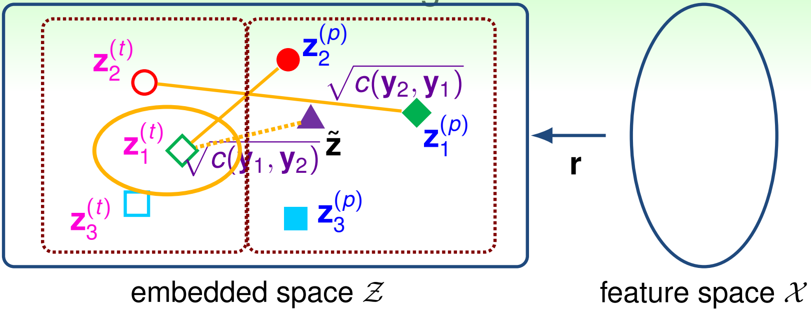
Optimization

- embedding error $\rightarrow$ multidimensional scaling
- regression error $\rightarrow$ regression $\mathbf{r}$

Challenge

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

Mirroring Trick



- two roles of $\mathbf{y}_i$: **ground truth role $\mathbf{y}_i^{(t)}$** and **prediction role $\mathbf{y}_i^{(p)}$**
- $\sqrt{c(\mathbf{y}_i, \mathbf{y}_j)} \Rightarrow$ predict $\mathbf{y}_i$ as $\mathbf{y}_j \Rightarrow$ for $\mathbf{z}_i^{(t)}$ and $\mathbf{z}_j^{(p)}$
- $\sqrt{c(\mathbf{y}_j, \mathbf{y}_i)} \Rightarrow$ predict $\mathbf{y}_j$ as $\mathbf{y}_i \Rightarrow$ for $\mathbf{z}_i^{(p)}$ and $\mathbf{z}_j^{(t)}$
- learn **regression function r** from $\mathbf{z}_i^{(p)}, \mathbf{z}_2^{(p)}, \ldots, \mathbf{z}_L^{(p)}$
- find **nearest embedded vector** of $\tilde{\mathbf{z}}$ from $\mathbf{z}_1^{(t)}, \mathbf{z}_2^{(t)}, \ldots, \mathbf{z}_L^{(t)}$

Cost-Sensitive Label Embedding with Multidimensional Scaling

Training Stage of **CLEMS**

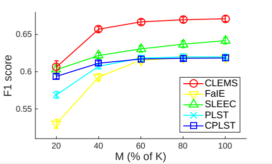
- given training instances $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ and cost function c
- determine two roles of embedded vectors $\mathbf{z}_n^{(t)}$ and $\mathbf{z}_n^{(p)}$ for label vector $\mathbf{y}_n$
- embedding function $\Phi\colon \mathbf{y}_n \to \mathbf{z}_i^{(p)}$
- learn a regression function $\mathbf{r}$ from $\{(\mathbf{x}_n, \Phi(\mathbf{y}_n))\}_{n=1}^{N}$

Predicting Stage of **CLEMS**

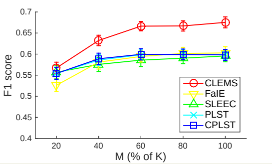
- given the testing instance $\mathbf{x}$
- obtain the predicted embedded vector by $\tilde{\mathbf{z}} = \mathbf{r}(\mathbf{x})$
- decoding $\Psi(\cdot) = \Phi^{-1}$(nearest neighbor) $= \Phi^{-1}\left(\operatorname{argmin} d(\mathbf{z}_n^{(t)}, \cdot)\right)$
- prediction $\tilde{\mathbf{y}} = \Psi(\tilde{\mathbf{z}})$

Comparison with Label Embedding Algorithms

F1 score (↑)

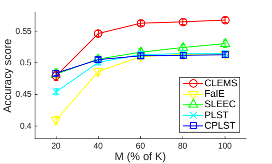


yeast

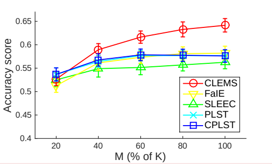


birds

Accuracy score (↑)

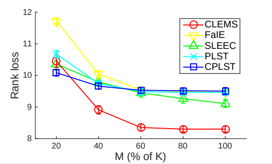


yeast

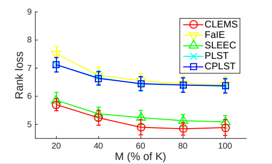


birds

Rank loss (↓)



yeast



birds

CLEMS is the best across different criteria and dimensions

Comparison with Cost-Sensitive Algorithms

data	F1 score (↑)			Accuracy score (↑)			Rank loss (↓)		
	CLEMS	CFT	PCC	CLEMS	CFT	PCC	CLEMS	CFT	PCC
emot.	**0.676**	0.640	0.643	**0.589**	0.557	–	1.484	1.563	**1.467**
scene	**0.770**	0.703	0.745	**0.760**	0.656	–	0.672	0.723	**0.645**
yeast	**0.671**	0.649	0.614	**0.568**	0.543	–	**8.302**	8.566	8.469
birds	**0.677**	0.601	0.636	**0.642**	0.586	–	4.886	4.908	**3.660**
med.	**0.814**	0.635	0.573	**0.786**	0.613	–	5.170	5.811	**4.234**
enron	**0.606**	0.557	0.542	**0.491**	0.448	–	29.40	26.64	**25.11**
lang.	**0.375**	0.168	0.247	**0.327**	0.164	–	31.03	34.16	**19.11**
flag	**0.731**	0.692	0.706	**0.615**	0.588	–	2.930	3.075	**2.857**
slash	**0.568**	0.429	0.503	**0.538**	0.402	–	4.986	5.677	**4.472**
CAL.	**0.419**	0.371	0.391	**0.273**	0.237	–	1247	1120	**993**
arts	**0.492**	0.334	0.349	**0.451**	0.281	–	9.865	10.07	**8.467**
EUR.	**0.670**	0.456	0.483	**0.650**	0.450	–	89.52	129.5	**43.28**

- **generality for CSMLC**: CLEMS $=$ CFT $>$ PCC
  - PCC requires an efficient inference rule
- **performance**: CLEMS $\approx$ PCC $>$ CFT

Conclusion

1. Compression Coding (Tai & Lin, MLD Workshop 2010; NC Journal 2012 with 172 citations)
   —**condense** for efficiency: better (than BR) approach PLST
   — key tool: PCA from Statistics/Signal Processing
2. Learnable-Compression Coding (Chen & Lin, NIPS Conference 2012 with 114 citations)
   —**condense learnably** for better efficiency: better (than PLST) approach CPLST
   — key tool: Ridge Regression from Statistics (+ PCA)
3. Cost-sensitive Coding (Huang & Lin, ECML Conference ML Journal Track 2017)
   —**condense cost-sensitively** towards application needs: better (than CPLST) approach CLEMS
   — key tool: Multidimensional Scaling from Statistics

Thank you! Questions?