Label Space Coding for Multi-label Classification

Hsuan-Tien Lin
National Taiwan University
RIKEN AIP Center, 08/30/2019

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

Which Fruits?

?: \{orange, strawberry, kiwi\}

apple  orange  strawberry  kiwi

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

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

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

<table>
<thead>
<tr>
<th>label set</th>
<th>apple</th>
<th>orange</th>
<th>strawberry</th>
<th>binary code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{Y}_1 = {o}$</td>
<td>0 (N)</td>
<td>1 (Y)</td>
<td>0 (N)</td>
<td>$y_1 = [0, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_2 = {a, o}$</td>
<td>1 (Y)</td>
<td>1 (Y)</td>
<td>0 (N)</td>
<td>$y_2 = [1, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_3 = {a, s}$</td>
<td>1 (Y)</td>
<td>0 (N)</td>
<td>1 (Y)</td>
<td>$y_3 = [1, 0, 1]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_4 = {o}$</td>
<td>0 (N)</td>
<td>1 (Y)</td>
<td>0 (N)</td>
<td>$y_4 = [0, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_5 = {}$</td>
<td>0 (N)</td>
<td>0 (N)</td>
<td>0 (N)</td>
<td>$y_5 = [0, 0, 0]$</td>
</tr>
</tbody>
</table>

For any subset $\mathcal{Y}$ of $2^{\{1,2,\ldots,L\}}$, there is a unique length-$L$ binary code $y$.
Existing Approach: Compressive Sensing

General Compressive Sensing

sparse (many 0) binary vectors $y \in \{0, 1\}^L$ can be **robustly compressed** by projecting to $M \ll L$ basis vectors $\{p_1, p_2, \cdots, p_M\}$

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

1. **compress**: transform $\{(x_n, y_n)\}$ to $\{(x_n, c_n)\}$ by $c_n = P y_n$ with some $M$ by $L$ random matrix $P = [p_1, p_2, \cdots, p_M]^T$
2. **learn**: get regression function $r(x)$ from $x_n$ to $c_n$
3. **decode**: $g(x) = \text{find closest sparse binary vector to } P^T r(x)$

Compressive Sensing:

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

better 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
Compressive Sensing Revisited

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

For any given “intermediate prediction” (real-valued vector) \( \tilde{y} \),

- find closest **sparse** binary vector to \( \tilde{y} \): slow optimization of \( \ell_1 \)-regularized objective
- find closest **any** binary vector to \( \tilde{y} \): fast

\[ g(x) = \text{round}(y) \]

**round-based decoding**: simple & faster alternative
Better Projection: Principal Directions

Compressible Sensing Revisited

- **compress**: transform \( \{(x_n, y_n)\} \) to \( \{(x_n, c_n)\} \) by \( c_n = Py_n \) with some \( M \) by \( L \) random matrix \( P \)

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

**principal directions**: best approximation to desired output \( y_n \) during compression (**why**?)
Novel Theoretical Guarantee

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

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

\[
\frac{1}{L} \left| g(x) \triangle Y \right| \leq \text{const} \cdot \left( \left\| r(x) - P y \right\|^2 \left( \underbrace{c}_{\text{learn}} \right) + \left\| y - P^T P y \right\|^2 \left( \underbrace{c}_{\text{compress}} \right) \right)
\]

- \( \left\| r(x) - c \right\|^2 \): prediction error from input to codeword
- \( \left\| y - P^T c \right\|^2 \): encoding error from desired output to codeword

principal directions: best approximation to desired output \( y_n \) during compression (indeed)
Proposed Approach 1: Principal Label Space Transform

From Compressive Sensing to **PLST**

1. **compress**: transform \( \{(x_n, y_n)\} \) to \( \{(x_n, c_n)\} \) by \( c_n = Py_n \) with the \( M \) by \( L \) **principal** matrix \( P \)

2. **learn**: get regression function \( r(x) \) from \( x_n \) to \( c_n \)

3. **decode**: \( g(x) = \text{round}(P^T r(x)) \)

- principal directions: via **Principal Component Analysis** on \( \{y_n\}_{n=1}^N \)
- physical meaning behind \( p_m \): 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 \( g(x) = \text{round}(P^T r(x)) \),

\[
\frac{1}{L} \left| g(x) \triangle Y \right| \leq \text{const} \cdot \left( \| r(x) - \hat{P}y \|^2 + \| y - P^T \hat{P}y \|^2 \right)
\]

- \( \| y - P^T c \|^2 \): encoding error, minimized during encoding
- \( \| r(x) - c \|^2 \): 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?)

H.-T. Lin (NTU)
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 \( \{(x_n, y_n)\} \) to \( \{(x_n, c_n)\} \) by \( c_n = Py_n \) with the \( M \) by \( L \) **conditional principal** matrix \( P \)

2. **learn**: get regression function \( r(x) \) from \( x_n \) to \( c_n \), ideally using ridge regression

3. **decode**: \( g(x) = \text{round}(P^T r(x)) \)

- conditional principal directions: top eigenvectors of \( Y^T XX^\dagger Y \), key (linear) label correlations that are “easy to learn”

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

<table>
<thead>
<tr>
<th># of dimension</th>
<th>Hamming loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.205</td>
</tr>
<tr>
<td>5</td>
<td>0.21</td>
</tr>
<tr>
<td>10</td>
<td>0.215</td>
</tr>
<tr>
<td>15</td>
<td>0.22</td>
</tr>
</tbody>
</table>

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

---

### 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

Given $N$ examples (input $x_n$, label-set $Y_n) \in \mathcal{X} \times 2^{\{1,2,\cdots,L\}}$

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

#### Goal

A multi-label classifier $g(x)$ that **closely predicts** the label-set $Y$ associated with some **unseen** inputs $x$

- **Hamming loss**: averaged symmetric difference $\frac{1}{L} | g(x) \triangle Y|$
Cost Functions for Multi-label Classification

Goal

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

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

Other Evaluation of Closeness

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

**Given**

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

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

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

**Goal**

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

next: label space coding for CSMLC
Cost-sensitive Coding

**Label Embedding**

- **Label Space** $\mathcal{Y}$
- **Embedded Space** $\mathcal{Z}$
- **Feature Space** $\mathcal{X}$

**Training Stage**

- **embedding function** $\Phi$: label vector $\mathbf{y} \rightarrow$ 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** $\Psi$: $\tilde{\mathbf{z}} \rightarrow$ 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, RAKEL, 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 $\Phi$ and decoding function $\Psi$ (cost-sensitive embedded vectors $z$)
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 Coding

Cost-Sensitive Embedding

Training Stage

- distances between embedded vectors $\Leftrightarrow$ cost information
- larger (smaller) distance $d(z_i, z_j) \Leftrightarrow$ higher (lower) cost $c(y_i, y_j)$
- $d(z_i, z_j) \approx \sqrt{c(y_i, y_j)}$ by multidimensional scaling (manifold learning)
Cost-Sensitive Decoding

- for testing instance $x$, predicted embedded vector $\tilde{z} = r(x)$
- find nearest embedded vector $z_q$ of $\tilde{z}$
- cost-sensitive prediction $\tilde{y} = y_q$
**Theoretical Explanation**

**Theorem (Huang and Lin, 2017)**

\[ c(y, \tilde{y}) \leq 5 \left( (d(z, z_q) - \sqrt{c(y, y_q)})^2 + \|z - r(x)\|^2 \right) \]

- **Optimization**
  - embedding error → multidimensional scaling
  - regression error → regression \( r \)

**Challenge**

- asymmetric cost function vs. symmetric distance?
  i.e. \( c(y_i, y_j) \neq c(y_j, y_i) \) vs. \( d(z_i, z_j) \)
Cost-sensitive Coding

Mirroring Trick

- two roles of $y_i$: ground truth role $y^{(t)}_i$ and prediction role $y^{(p)}_i$
- $\sqrt{c(y_i, y_j)} \Rightarrow$ predict $y_i$ as $y_j \Rightarrow$ for $z^{(t)}_i$ and $z^{(p)}_j$
- $\sqrt{c(y_j, y_i)} \Rightarrow$ predict $y_j$ as $y_i \Rightarrow$ for $z^{(p)}_i$ and $z^{(t)}_j$
- learn regression function $r$ from $z^{(p)}_1, z^{(p)}_2, ..., z^{(p)}_L$
- find nearest embedded vector of $\tilde{z}$ from $z^{(t)}_1, z^{(t)}_2, ..., z^{(t)}_L$
Cost-Sensitive Label Embedding with Multidimensional Scaling

**Training Stage of CLEMS**

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

**Predicting Stage of CLEMS**

- given the testing instance $x$
- obtain the predicted embedded vector by $\tilde{z} = r(x)$
- decoding $\Psi(\cdot) = \Phi^{-1}(\text{nearest neighbor}) = \Phi^{-1}(\text{argmin } d(z^{(t)}_n, \cdot))$
- prediction $\tilde{y} = \Psi(\tilde{z})$
Comparison with Label Embedding Algorithms

### F1 score (↑)

<table>
<thead>
<tr>
<th>M (% of K)</th>
<th>yeast</th>
<th>birds</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.55</td>
<td>0.4</td>
</tr>
<tr>
<td>40</td>
<td>0.58</td>
<td>0.45</td>
</tr>
<tr>
<td>60</td>
<td>0.60</td>
<td>0.5</td>
</tr>
<tr>
<td>80</td>
<td>0.62</td>
<td>0.55</td>
</tr>
<tr>
<td>100</td>
<td>0.65</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### Accuracy score (↑)

<table>
<thead>
<tr>
<th>M (% of K)</th>
<th>yeast</th>
<th>birds</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>40</td>
<td>0.45</td>
<td>0.45</td>
</tr>
<tr>
<td>60</td>
<td>0.5</td>
<td>0.55</td>
</tr>
<tr>
<td>80</td>
<td>0.55</td>
<td>0.6</td>
</tr>
<tr>
<td>100</td>
<td>0.6</td>
<td>0.65</td>
</tr>
</tbody>
</table>

### Rank loss (↓)

<table>
<thead>
<tr>
<th>M (% of K)</th>
<th>yeast</th>
<th>birds</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>60</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>80</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>100</td>
<td>11</td>
<td>9</td>
</tr>
</tbody>
</table>

CLEMS is the best across different criteria and dimensions.
### Comparison with Cost-Sensitive Algorithms

<table>
<thead>
<tr>
<th>data</th>
<th>F1 score (↑)</th>
<th>Accuracy score (↑)</th>
<th>Rank loss (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CLEMS</td>
<td>CFT</td>
<td>PCC</td>
</tr>
<tr>
<td>emot.</td>
<td>0.676</td>
<td>0.640</td>
<td>0.643</td>
</tr>
<tr>
<td>scene</td>
<td>0.770</td>
<td>0.703</td>
<td>0.745</td>
</tr>
<tr>
<td>yeast</td>
<td>0.671</td>
<td>0.649</td>
<td>0.614</td>
</tr>
<tr>
<td>birds</td>
<td>0.677</td>
<td>0.601</td>
<td>0.636</td>
</tr>
<tr>
<td>med.</td>
<td>0.814</td>
<td>0.635</td>
<td>0.573</td>
</tr>
<tr>
<td>enron</td>
<td>0.606</td>
<td>0.557</td>
<td>0.542</td>
</tr>
<tr>
<td>lang.</td>
<td>0.375</td>
<td>0.168</td>
<td>0.247</td>
</tr>
<tr>
<td>flag</td>
<td>0.731</td>
<td>0.692</td>
<td>0.706</td>
</tr>
<tr>
<td>slash</td>
<td>0.568</td>
<td>0.429</td>
<td>0.503</td>
</tr>
<tr>
<td>CAL.</td>
<td>0.419</td>
<td>0.371</td>
<td>0.391</td>
</tr>
<tr>
<td>arts</td>
<td>0.492</td>
<td>0.334</td>
<td>0.349</td>
</tr>
<tr>
<td>EUR.</td>
<td>0.670</td>
<td>0.456</td>
<td>0.483</td>
</tr>
</tbody>
</table>

- **generality for CSMLC**: CLEMS = CFT > PCC
- PCC requires an efficient inference rule
- **performance**: CLEMS ≈ 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?

H.-T. Lin (NTU)