Label Space Coding for Multi-label Classification

Hsuan-Tien Lin
National Taiwan University
3rd TWSIAM Annual Meeting, 05/30/2015

joint works with
Farbound Tai (MLD Workshop 2010, NC Journal 2012) &
Yao-Nan Chen (NIPS Conference 2012)
Multi-label Classification

Which Fruit?

apple  orange  strawberry  kiwi

multi-class classification: classify input (picture) to **one category** (label)
Which Fruits?

? : \{orange, strawberry, kiwi\}

apple  orange  strawberry  kiwi

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

**: \{\text{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 Classification

Multi-label Classification Setup

**Given**

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

- **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 \( x \) (by exploiting hidden relations/combinations between labels)

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

**multi-label classification: hot and important**
Compression Coding

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>$\mathcal{Y}_1 = {\text{o}}$</td>
<td>0 (N)</td>
<td>1 (Y)</td>
<td>0 (N)</td>
<td>$y_1 = [0, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_2 = {\text{a, o}}$</td>
<td>1 (Y)</td>
<td>1 (Y)</td>
<td>0 (N)</td>
<td>$y_2 = [1, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_3 = {\text{a, s}}$</td>
<td>1 (Y)</td>
<td>0 (N)</td>
<td>1 (Y)</td>
<td>$y_3 = [1, 0, 1]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_4 = {\text{o}}$</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>

subset $\mathcal{Y}$ of $2^{\{1, 2, \ldots, L\}} \Leftrightarrow$ 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 $c_n = Py_n$ 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**: $g(x) = \text{find closest sparse binary vector to } P^Tr(x)$

Compressive Sensing:

- **efficient in training**: random projection w/ $M \ll L$
- **inefficient in testing**: time-consuming decoding
From Coding View to Geometric View

<table>
<thead>
<tr>
<th>Label Set</th>
<th>Binary Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{Y}_1 = {o}$</td>
<td>$y_1 = [0, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_2 = {a, o}$</td>
<td>$y_2 = [1, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_3 = {a, s}$</td>
<td>$y_3 = [1, 0, 1]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_4 = {o}$</td>
<td>$y_4 = [0, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{Y}_5 = {}$</td>
<td>$y_5 = [0, 0, 0]$</td>
</tr>
</tbody>
</table>

Length-$L$ binary code $\Leftrightarrow$ vertex of hypercube $\{0, 1\}^L$
Geometric Interpretation of Binary Relevance

Binary Relevance: project to the **natural axes** & classify
Geometric Interpretation of Compressive Sensing

Compressive Sensing:
- project to **random flat** (linear subspace)
- learn “on” the flat; decode to **closest sparse vertex**

other (better) flat? other (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

Compressible 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

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

\[
\frac{1}{L} |g(x) \Delta y| \leq \text{const} \cdot \left( \|r(x) - Py\|_2^2 + \|y - P^T Py\|_2^2 \right)
\]

- \( \|r(x) - c\|_2^2 \): prediction error from input to codeword
- \( \|y - P^T c\|_2^2 \): 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 \( g(x) = \text{round}(P^T r(x)) \),

\[
\frac{1}{L} \sum_{i=1}^{L} |g(x) \triangle y| \leq \text{const} \cdot \left( \| r(x) - P y \|_2^2 + \| y - P^T P y \|_2^2 \right)
\]

Hamming loss

- \( \| y - P^T c \|_2^2 \): encoding error, minimized during encoding
- \( \| r(x) - c \|_2^2 \): 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 \( \{(x_n, y_n)\} \) to \( \{(x_n, c_n)\} \) by \( c_n = Py_n \) with the \( M \times L \) **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 \( Y^T X X^\dagger Y \), key (linear) label correlations that are “easy to learn”

---

CPLST: project to **key learnable** correlations —can also pair with **kernel regression** (non-linear)
Hamming Loss Comparison: Full-BR, PLST & CS

- **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

- CPLST better than PLST: better performance across all dimensions
- Similar findings across data sets and regression algorithms
1. **Compression Coding** (Tai & Lin, MLD Workshop 2010; NC Journal 2012)
   - condense for efficiency: better (than BR) approach PLST
   - key tool: PCA from Statistics/Signal Processing

2. **Learnable-Compression Coding** (Chen & Lin, NIPS Conference 2012)
   - condense learnably for better efficiency: better (than PLST) approach CPLST
   - key tool: Ridge Regression from Statistics (+ PCA)

More......

- error-correcting code instead of compression, with improved decoding (Ferng and Lin, IEEE TNNLS 2013)
- multi-label classification with arbitrary loss (Li and Lin, ICML 2014)
- dynamic instead of static coding, binary instead of real coding, (...)

Thank you! Questions?