Label Space Coding for Multi-label Classification

Hsuan-Tien Lin
National Taiwan University
NUK Seminar, 12/22/2017

joint works with
Farbound Tai (MLD Workshop 2010, NC Journal 2012) &
Yao-Nan Chen (NIPS Conference 2012) &
Chun-Sung Ferng (ACML Conference 2011, TNNLS Journal 2013)
Multi-label Classification

Which Fruit?

apple  orange  strawberry  kiwi

multi-class classification: classify input (picture) to **one category** (label)
Multi-label Classification

Which Fruits?

?: \{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
Multi-label Classification

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

- **0/1 loss:** any discrepancy $[g(x) \neq \mathcal{Y}]$
- **Hamming loss:** averaged symmetric difference $\frac{1}{L} |g(x) \triangle \mathcal{Y}|$

**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>

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 $\{p_1, p_2, \ldots, p_M\}$.

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 $M \times L$ random matrix $P = [p_1, p_2, \ldots, p_M]^T$.
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.
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
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
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
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?)
Theorem (Tai and Lin, 2012)

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

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

- \( \| r(x) - c \|_2^2 \): 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 \( g(x) = \text{round}(P^Tr(x)) \),

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

- \( \left\| y - P^Tc \right\|_2^2 \): encoding error, minimized during encoding
- \( \left\| r(x) - c \right\|_2^2 \): 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 \( M \) 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 \( Y^T X X^\dagger Y \), 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

yeast (Linear Regression)
Topics in this Talk

1. **Compression Coding**
   - *condense* for efficiency
   - capture hidden correlation

2. **Learnable-Compression Coding**
   - *condense-by-learnability* for **better** efficiency
   - capture hidden & **learnable** correlation

3. **Error-Correction Coding**
   - *expand* for accuracy
   - capture hidden combination
Our Contributions (Second Part)

**Error-correction Coding**

A Novel Framework for Label Space Error-correction

- **algorithmic**: generalize an popular existing algorithm (RAkEL; Tsoumakas & Vlahavas, 2007) and explain through coding view
- **theoretical**: link learning performance to error-correcting ability
- **practical**: explore choices of error-correcting code and obtain better performance than RAkEL (& binary relevance)
Key Idea: Redundant Information

General Error-correcting Codes (ECC)
- commonly used in communication systems
- detect & correct errors after transmitting data over a noisy channel
- encode data redundantly

ECC for Machine Learning (successful for multi-class classification)
- learn redundant bits $\Rightarrow$ correct prediction errors
Proposed Framework: Multi-labeling with ECC

- **encode** to add redundant information $\text{enc}(\cdot): \{0, 1\}^L \rightarrow \{0, 1\}^M$
- **decode** to locate most possible binary vector $\text{dec}(\cdot): \{0, 1\}^M \rightarrow \{0, 1\}^L$
- transformation to larger multi-label classification with labels $\mathbf{b}$

PLST: $M \ll L$ (works for large $L$);
MLECC: $M > L$ (works for small $L$)
ECC encode + Larger Multi-label Learning + ECC decode

**Theorem**

Let \( g(x) = \text{dec}(\tilde{b}) \) with \( \tilde{b} = h(x) \). Then,

\[
\underbrace{\mathbb{I}[g(x) \neq \mathcal{Y}]}_{0/1 \text{ loss}} \leq \text{const.} \cdot \frac{\text{Hamming loss of } h(x)}{\text{ECC strength} + 1}.
\]

**PLST:** principal directions + decent regression

**MLECC:** which ECC balances **strength** & **difficulty**?
Simplest ECC: Repetition Code

**Encoding:** \( y \in \{0, 1\}^L \rightarrow b \in \{0, 1\}^M \)

- **repeat** each bit \( \frac{M}{L} \) times

  \[ L = 4, M = 28 : 1010 \rightarrow \underbrace{1111111}_{28} \ 0000000111111110000000 \]
  \[ \frac{28}{4} = 7 \]

- permute the bits randomly

**Decoding:** \( \tilde{b} \in \{0, 1\}^M \rightarrow \tilde{y} \in \{0, 1\}^L \)

- **majority vote** on each original bit

  \[ L = 4, M = 28: \text{strength of repetition code (REP)} = 3 \]

\[ \text{RAkEL = REP (code) + a special powerset (channel)} \]
Slightly More Sophisticated: Hamming Code

**HAM(7, 4) Code**

- \( \{0, 1\}^4 \rightarrow \{0, 1\}^7 \) via adding 3 **parity bits**
  —physical meaning: **label combinations**
- \( b_4 = y_0 \oplus y_1 \oplus y_3, b_5 = y_0 \oplus y_2 \oplus y_3, b_6 = y_1 \oplus y_2 \oplus y_3 \)
- e.g. 1011 → 1011010
- strength = 1 (weak)

**Our Proposed Code: Hamming on Repetition (HAMR)**

\[
\{0, 1\}^L \xrightarrow{\text{REP}} \{0, 1\}^{4M/7} \xrightarrow{\text{HAM}(7, 4) \text{ on each 4-bit block}} \{0, 1\}^{7M/7}
\]

\( L = 4, M = 28 \): strength of HAMR = 4 **better than REP**!

**HAMR + the special powerset:**

improve RAKEL on **code strength**
Even More Sophisticated Codes

Bose-Chaudhuri-Hocquenghem Code (BCH)
- modern code in **CD players**
- sophisticated extension of Hamming, with **more parity bits**
- codeword length $M = 2^p - 1$ for $p \in \mathbb{N}$
- $L = 4$, $M = 31$, strength of BCH = 5

Low-density Parity-check Code (LDPC)
- modern code for **satellite communication**
- connect ECC and Bayesian learning
- approach the theoretical limit in some cases

let’s compare!
Different ECCs on 3-label Powerset

- learner: special powerset with Random Forests
- REP + special powerset $\approx$ RAKEL

### Comparing to RAKEL (on most of data sets),
- HAMR: better 0/1 loss, similar Hamming loss
- BCH: even better 0/1 loss, pay for Hamming loss
# Conclusion

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)

3. **Error-correction Coding** *(Ferng & Lin, ACML Conference 2011, TNNLS Journal 2013)*
   - *expand* for accuracy: better (than REP) code HAMR or BCH
   - key tool: ECC from Information Theory

Thank you! Questions?