Label Space Coding for Multi-label Classification

Hsuan-Tien Lin
National Taiwan University
Intel Seminar, 02/11/2015

joint works with

Farbound Tai  (MLD Workshop 2010, NC Journal 2012) &
Chun-Sung Ferng  (ACML Conference 2011, IEEE TNNLS Journal 2013) &
Yao-Nan Chen  (NIPS Conference 2012)
Hsuan-Tien Lin

- Associate Professor, Dept. of CSIE, National Taiwan University
- Leader of the Computational Learning Laboratory
- Co-author of the textbook “Learning from Data: A Short Course” (often ML best seller on Amazon)
- Instructor of the NTU-Coursera Mandarin-teaching ML MOOCs “Machine Learning Foundations” and “Machine Learning Techniques”

**goal: make machine learning more realistic**

- multi-class cost-sensitive classification: in ICML ’10, BIBM ’11, KDD ’12, ACML ’14, etc.
- **multi-label classification** : in ACML ’11, NIPS ’12, ICML ’14, etc.
- online/active learning: in ICML ’12, ACML ’12, ICML ’14, AAAI ’15, etc.
- video search: CVPR ’11
- large-scale data mining (w/ Profs. S.-D. Lin & C.-J. Lin & students):
  - third place of KDDCup ’09, champions of ’10, ’11 (×2), ’12, ’13 (×2)
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
Multi-label Classification

Powerset: Multi-label Classification via Multi-class

**Powerset** approach: transformation to multi-class classification

- **difficulties for large $L$:**
  - **computation** (super-large $2^L$) — hard to construct classifier
  - **sparsity** (no example for some of $2^L$) — hard to discover hidden combination

**Powerset**: feasible only for **small $L$** with enough examples for every combination

---

**Multi-class w/ $L = 4$ classes**

- 4 possible outcomes
  - $\{a, o, s, k\}$

**Multi-label w/ $L = 4$ classes**

- $2^4 = 16$ **possible outcomes**
  - $2\{a, o, s, k\}$
  - $\{\emptyset, a, o, s, k, ao, as, ak, os, ok, sk, aos, aok, ask, osk, aosk\}$

---

H.-T. Lin (NTU)
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 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**

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

- **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 \( x \) (by exploiting hidden relations/combinations between labels)
  
  - **0/1 loss:** any discrepancy \( [g(x) \neq Y] \)
  - **Hamming loss:** averaged symmetric difference \( \frac{1}{L} |g(x) \triangle Y| \)

**multi-label classification: hot and important**
Topics in this Talk

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

2. **Error-correction Coding**
   - expand for accuracy
   - capture hidden combination

3. **Learnable-Compression Coding**
   - condense-by-learnability for **better** efficiency
   - capture hidden & **learnable** correlation
### 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 \( M \) by \( L \) 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 Powerset

Powerset: directly classify to the vertices of hypercube
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 (First Part)

Compression Coding

A Novel Approach for Label Space Compression

- algorithmic: scheme for fast decoding
- theoretical: justification for best projection
- practical: significantly better performance than compressive sensing (& binary relevance)
For any given “intermediate prediction” (real-valued vector) \( \tilde{y} \),

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

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

\textit{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?)
Compression Coding

Novel Theoretical Guarantee

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

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

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

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

principal directions: best approximation to desired output \( y_n \) during compression (indeed)

H.-T. Lin (NTU)
**Proposed Approach: Principal Label Space Transform (PLST)**

1. **Compress**: transform \( \{(x_n, y_n)\} \) to \( \{(x_n, c_n)\} \) by \( c_n = Py_n \) with the \( M \times L \) 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**.
**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**
Semi-summary on PLST

- project to **principal directions** and capture key correlations
- efficient learning (after **label space compression**)
- efficient decoding (**round-based**)
- sound theoretical guarantee + **good practical performance** (better than CS & BR)

**expansion** (channel coding) instead of compression ("lossy" source coding)? YES!
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)
Error-correction Coding

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:** $M \gg L$ (works for small $L$)
Simple Theoretical Guarantee

**Theorem**

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

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

**ECC encode + Larger Multi-label Learning + ECC decode**

**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

\[
L = 4, M = 28 : 1010 \rightarrow \overbrace{1111111}^{28/4 = 7} 000000111111110000000
\]

- permute the bits randomly

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

- majority vote on each original bit

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

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)**

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

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

**HAMR + the special powerset:**

improve RAkEL on code strength
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!**
Error-correction Coding

Different ECCs on 3-label Powerset (scene data set w/ \( L = 6 \))

- 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
transformation to larger multi-label classification

encode via **error-correcting code** and capture label combinations (parity bits)

effective decoding (**error-correcting**)

simple theoretical guarantee + **good practical performance**

- to **improve RAkEL**, replace REP by
  - HAMR $\implies$ lower 0/1 loss, similar Hamming loss
  - BCH $\implies$ even lower 0/1 loss, but higher Hamming loss

- to **improve Binary Relevance**, · · ·
Learnable-Compression Coding

Theoretical Guarantee of PLST Revisited

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

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

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

- \( \| y - P^T c \|^2 \): 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?)
Learnable-Compression Coding

A Novel Approach for Label Space Compression

- algorithmic: first known algorithm for feature-aware dimension reduction
- theoretical: justification for best learnable projection
- practical: consistently better performance than PLST
The In-Sample Optimization Problem

\[ \min_{r,P} \left( \| r(X) - PY \|^2 + \| Y - P^T PY \|^2 \right) \]

- start from a well-known tool: linear regression as \( r \)
  \[ r(X) = XW \]

- for fixed \( P \): a closed-form solution for learn is
  \[ W^* = X^TPY \]

**optimal \( P \):**

| For learn | Top eigenvectors of \( Y^T(I - XX^\dagger)Y \) |
| For compress | Top eigenvectors of \( Y^TY \) |
| For both | Top eigenvectors of \( Y^TXX^\dagger Y \) |
Learnable-Compression Coding

Proposed Approach: **Conditional** Principal Label Space Transform

---

**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 linear regression

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

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

---

**CPLST**: project to **key learnable** correlations —can also pair with **kernel regression** (non-linear)
CPLST better than PLST: better performance across all dimensions

similar findings across data sets and regression algorithms
Semi-summary on CPLST

- project to **conditional** principal directions and capture **key learnable** correlations
- more efficient
- sound theoretical guarantee (via PLST) + **good practical performance** (better than PLST)

CPLST: **state-of-the-art** for label space compression
**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. **Error-correction Coding** (Ferng & Lin, ACML Conference 2011)
   - *expand* for accuracy: better (than REP) code HAMR or BCH
   - key tool: ECC from Information Theory

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

**More......**

- beyond standard ECC-decoding (Ferng and Lin, IEEE TNNLS 2013)
- kernelizing CPLST (Chen and Lin, NIPS 2012)
- multi-label classification with arbitrary loss (Li and Lin, ICML 2014)
- dynamic instead of static coding, combine ML-ECC & PLST/CPLST (...)

H.-T. Lin (NTU)
Thank you! Questions?