Feature-aware Label Space Dimension Reduction for Multi-label Classification

Hsuan-Tien Lin

Computer Science and Information Engineering
National Taiwan University

05/04/2013, AI Forum

first part: with Farbound Tai, in Neural Computation 2012
second part: with Yao-Nan Chen, in NIPS 2012
A Short Introduction

Hsuan-Tien Lin

- Associate Professor, CSIE, National Taiwan University
- Secretary General, TAAI
- Co-author of the introductory ML textbook “Learning from Data: A Short Course” (Amazon ML best seller!)
- Leader of the Computational Learning Laboratory

**goal: make machine learning more realistic**

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

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
Powerset: Multi-label Classification via Multi-class

Multi-class w/ \( L = 4 \) classes

4 possible outcomes
\( \{a, o, s, k\} \)

Multi-label w/ \( L = 4 \) classes

\( 2^4 = 16 \) possible outcomes
\( 2^{\{a, o, s, k\}} \)

\( \phi, a, o, s, k, ao, as, ak, os, ok, sk, aos, aok, ask, osk, aok \}

- **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
What Tags?

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

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

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

**multi-label classification**: **hot and important**
<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 \( 2^{\{1,2,\ldots,L\}} \) \( \iff \) 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**: \( g(x) = \) 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>$\mathcal{V}_1 = {o}$</td>
<td>$y_1 = [0, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{V}_2 = {a, o}$</td>
<td>$y_2 = [1, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{V}_3 = {a, s}$</td>
<td>$y_3 = [1, 0, 1]$</td>
</tr>
<tr>
<td>$\mathcal{V}_4 = {o}$</td>
<td>$y_4 = [0, 1, 0]$</td>
</tr>
<tr>
<td>$\mathcal{V}_5 = {}$</td>
<td>$y_5 = [0, 0, 0]$</td>
</tr>
</tbody>
</table>

Length-$L$ binary code $\iff$ 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

Two Novel Approaches for Label Space Dimension Reduction

- **algorithmic**: scheme for **fast decoding**
- **theoretical**: justification for **best projection**, one **feature-unaware** and the other **feature-aware**
- **practical**: **significantly better performance** than compressive sensing (& binary relevance)

will now introduce the key ideas behind the approaches
Faster Decoding: Round-based

Compressive Sensing Revisited

- **decode**: \( g(x) = \text{sparse binary vector that } P\text{-projects closest to } r(x) \)

For any given “prediction on subspace” \( r(x) \),
- find **sparse binary vector** that \( P\text{-projects closest to } r(x) \): *slow*
  —optimization of \( \ell_1\)-regularized objective
- find **any binary vector** that \( P\text{-projects closest to } r(x) \): *fast*

\[ g(x) = \text{round}(P^T r(x)) \text{ for orthogonal } P \]

**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 **dimension reduction** (why?)
Novel Theoretical Guarantee

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

If \( g(x) = \text{round}(P^T r(x)) \) (\( p_m \) orthogonal to each other),

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

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

\text{Hamming loss}

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

From Compressive Sensing to PLST

1. **compress**: transform \( \{(x_n, y_n)\} \) to \( \{(x_n, c_n)\} \) by \( c_n = P y_n \) 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 \( \{y_n\}_{n=1}^N \)
  —BTW, improvements when shifting \( y_n \) by its estimated mean

- **physical meaning behind** \( p_m \): key (linear) label correlations

**PLST**: improving CS by projecting to **key correlations**
<table>
<thead>
<tr>
<th></th>
<th>PLST</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compress</td>
<td>projection through SVD (principal directions)</td>
<td>random basis projection (random directions)</td>
</tr>
<tr>
<td>Learn</td>
<td>multi-output regression</td>
<td></td>
</tr>
<tr>
<td>Decode</td>
<td>round-based (fast)</td>
<td>sparsity-based (slower)</td>
</tr>
</tbody>
</table>

practical performance?
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
Theoretical Guarantee of PLST Revisited

**Linear Transform + Learn + Round-based Decoding**

**Theorem (Tai and Lin, 2012)**

If $g(x) = \text{round}(P^Tr(x))$,

$$\frac{1}{L} |g(x) \triangle \mathcal{Y}| \leq \text{const} \cdot \left( \left\| r(x) - P^T \mathbf{c} \right\|^2 + \left\| \mathbf{y} - P^T P \mathbf{y} \right\|^2 \right)$$

- $\left\| \mathbf{y} - P^T \mathbf{c} \right\|^2$: encoding error, minimized during encoding
- $\left\| r(x) - \mathbf{c} \right\|^2$: 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 even better by minimizing **jointly**?)
The In-Sample Optimization Problem

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

- start from a well-known tool, linear regression, as \( r(r(X)) = XW \)
- for fixed \( P \): a closed-form solution for learn is \( W^* = X^\dagger PY \)
- substitute \( W^* \) to objective function, then ...

Optimal \( P \):
- for learn: top eigenvectors of \( Y^T(I - XX^\dagger)Y \)
- for compress: top eigenvectors of \( Y^T Y \)
- for both: top eigenvectors of \( Y^T XX^\dagger Y \)
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 \( Y^TXX^\dagger Y \)
- **physical meaning behind \( p_m \)**: key (linear) label correlations that are “easy to learn” subject to the features (feature-aware)

### CPLST: **feature-aware** label space dimension reduction
—can also pair with **kernel regression** (non-linear)
## CPLST vs. PLST

<table>
<thead>
<tr>
<th>CPLST</th>
<th>PLST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compress</td>
<td>SVD</td>
</tr>
<tr>
<td>Linear/Kernel Regression + SVD</td>
<td>(principal directions)</td>
</tr>
<tr>
<td>(conditional principal directions)</td>
<td></td>
</tr>
<tr>
<td>Learn</td>
<td>multi-output regression</td>
</tr>
<tr>
<td>Decode</td>
<td>round-based</td>
</tr>
<tr>
<td></td>
<td>(fast)</td>
</tr>
</tbody>
</table>

**practical performance?**
CPLST better than PLST: better performance across all dimensions

similar findings across data sets and regression algorithms (even decision trees)
PLST
- transformation to multi-output regression
- project to principal directions and capture key correlations
- efficient learning (after label space dimension reduction)
- efficient decoding (round)
- sound theoretical guarantee
- good practical performance (better than CS & BR)

CPLST
- project to conditional (feature-aware) principal directions and capture key learnable correlations
- can be kernelized for exploiting feature power
- sound theoretical guarantee (via PLST)
- even better practical performance (than PLST)

Thank you! Questions?