Label Space Coding for Multi-label Classification

Hsuan-Tien Lin

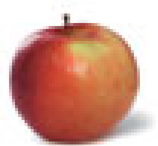
National Taiwan University

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)

Which Fruit?

?



apple

orange

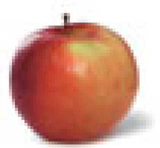
strawberry

kiwi

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

Which Fruits?

?: {orange, strawberry, kiwi}



apple

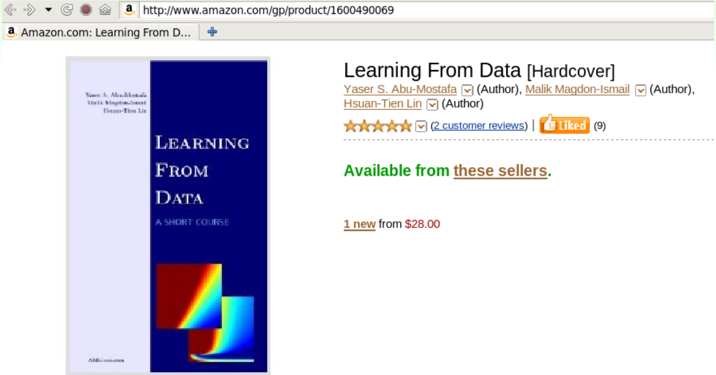
orange

strawberry

kiwi

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

What **Tags**?



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

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

multi-label classification: hot and important

From Label-set to Coding View

	label set	apple	orange	strawberry	**binary code**
	$\mathcal{Y}_1 = \{\text{o}\}$	0 (N)	1 (Y)	0 (N)	$\mathbf{y}_1 = [0, 1, 0]$
	$\mathcal{Y}_2 = \{\text{a, o}\}$	1 (Y)	1 (Y)	0 (N)	$\mathbf{y}_2 = [1, 1, 0]$
	$\mathcal{Y}_3 = \{\text{a, s}\}$	1 (Y)	0 (N)	1 (Y)	$\mathbf{y}_3 = [1, 0, 1]$
	$\mathcal{Y}_4 = \{\text{o}\}$	0 (N)	1 (Y)	0 (N)	$\mathbf{y}_4 = [0, 1, 0]$
	$\mathcal{Y}_5 = \{\}$	0 (N)	0 (N)	0 (N)	$\mathbf{y}_5 = [0, 0, 0]$

subset $\mathcal{Y}$ of $2^{\{1,2,\cdots,L\}}$ $\Leftrightarrow$ **length-L binary code $\mathbf{y}$**

Existing Approach: Compressive Sensing

General Compressive Sensing

sparse (many 0) binary vectors $\mathbf{y} \in \{0,1\}^L$ can be **robustly compressed** by projecting to $M \ll L$ basis vectors $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_M\}$

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

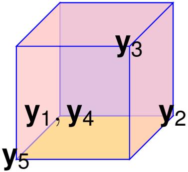
1. **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with some M by L random matrix $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_M]^T$
2. **learn**: get regression function $\mathbf{r}(\mathbf{x})$ from $\mathbf{x}_n$ to $\mathbf{c}_n$
3. **decode**: $g(\mathbf{x})$ = find closest sparse binary vector to $\mathbf{P}^T\mathbf{r}(\mathbf{x})$

Compressive Sensing:

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

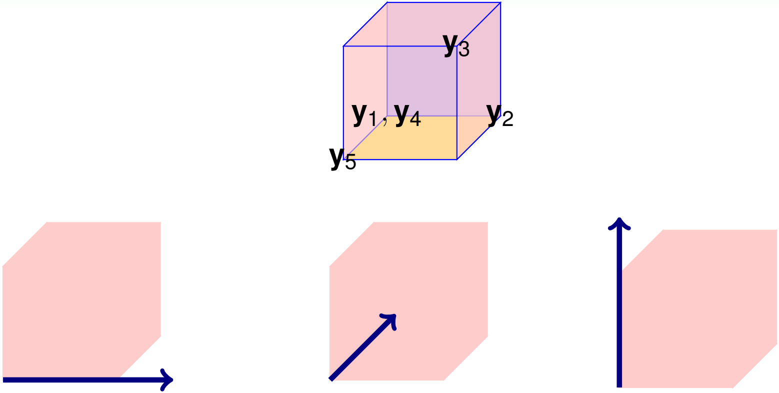
From Coding View to Geometric View

	label set	**binary code**
	$\mathcal{Y}_1 = \{o\}$	$\mathbf{y}_1 = [0, 1, 0]$
	$\mathcal{Y}_2 = \{a, o\}$	$\mathbf{y}_2 = [1, 1, 0]$
	$\mathcal{Y}_3 = \{a, s\}$	$\mathbf{y}_3 = [1, 0, 1]$
	$\mathcal{Y}_4 = \{o\}$	$\mathbf{y}_4 = [0, 1, 0]$
	$\mathcal{Y}_5 = \{\}$	$\mathbf{y}_5 = [0, 0, 0]$



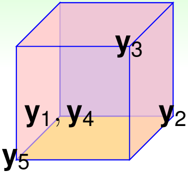
length-L binary code $\Leftrightarrow$ **vertex of hypercube** $\{0, 1\}^L$

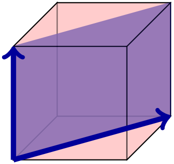
Geometric Interpretation of Binary Relevance



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

Compressive Sensing Revisited

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

For any given "intermediate prediction" (real-valued vector) $\tilde{\mathbf{y}}$,

- find closest sparse binary vector to $\tilde{\mathbf{y}}$: slow optimization of ℓ_1-regularized objective
- find closest any binary vector to $\tilde{\mathbf{y}}$: fast

$$g(\mathbf{x}) = \text{round}(\mathbf{y})$$

round-based decoding: simple & faster alternative

Better Projection: Principal Directions

Compressive Sensing Revisited

- **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with some M by L random matrix $\mathbf{P}$

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

principal directions: best approximation to desired output $\mathbf{y}_n$ during compression (**why?**)

Novel Theoretical Guarantee

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

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

$$\underbrace{\frac{1}{L}|g(\mathbf{x}) \triangle \mathcal{Y}|}_{\text{Hamming loss}} \leq const \cdot \left(\underbrace{\|\mathbf{r}(\mathbf{x}) - \overbrace{\mathbf{Py}}^{\mathbf{c}}\|^2}_{learn} + \underbrace{\|\mathbf{y} - \mathbf{P}^T \overbrace{\mathbf{Py}}^{\mathbf{c}}\|^2}_{compress} \right)$$

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

principal directions: best approximation to desired output $\mathbf{y}_n$ during compression (**indeed**)

Proposed Approach 1: Principal Label Space Transform

From Compressive Sensing to PLST

1. **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with the M by L principal matrix $\mathbf{P}$
2. **learn**: get regression function $\mathbf{r}(\mathbf{x})$ from $\mathbf{x}_n$ to $\mathbf{c}_n$
3. **decode**: $g(\mathbf{x}) = \text{round}(\mathbf{P}^T\mathbf{r}(\mathbf{x}))$

- principal directions: via Principal Component Analysis on $\{\mathbf{y}_n\}_{n=1}^N$
- physical meaning behind $\mathbf{p}_m$: 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 $g(\mathbf{x}) = round(\mathbf{P}^T\mathbf{r}(\mathbf{x}))$,

$$\underbrace{\frac{1}{L}|g(\mathbf{x}) \triangle \mathcal{Y}|}_{\text{Hamming loss}} \leq const \cdot \left(\underbrace{\|\mathbf{r}(\mathbf{x}) - \overbrace{\mathbf{Py}}^{\mathbf{c}}\|^2}_{learn} + \underbrace{\|\mathbf{y} - \mathbf{P}^T\overbrace{\mathbf{Py}}^{\mathbf{c}}\|^2}_{compress} \right)$$

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

Proposed Approach 2: Conditional Principal Label Space Transform

can we do better by minimizing jointly?

Yes and easy for ridge regression (closed-form solution)

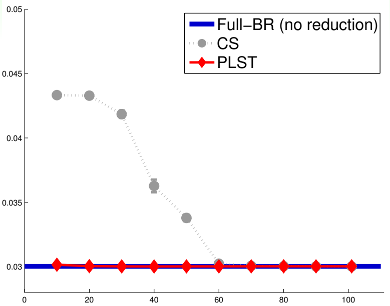
From PLST to CPLST

1. **compress**: transform $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ to $\{(\mathbf{x}_n, \mathbf{c}_n)\}$ by $\mathbf{c}_n = \mathbf{P}\mathbf{y}_n$ with the M by L conditional principal matrix $\mathbf{P}$
2. **learn**: get regression function $\mathbf{r}(\mathbf{x})$ from $\mathbf{x}_n$ to $\mathbf{c}_n$, ideally using ridge regression
3. **decode**: $g(\mathbf{x}) = \text{round}(\mathbf{P}^T\mathbf{r}(\mathbf{x}))$

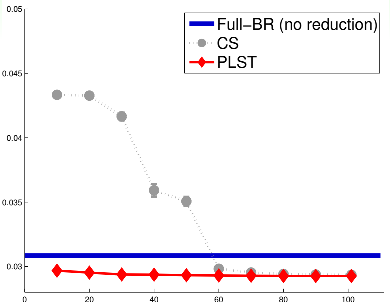
- conditional principal directions: top eigenvectors of $\mathbf{Y}^T\mathbf{X}\mathbf{X}^{\dagger}\mathbf{Y}$, key (linear) label correlations that are "easy to learn"

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

Hamming Loss Comparison: Full-BR, PLST & CS



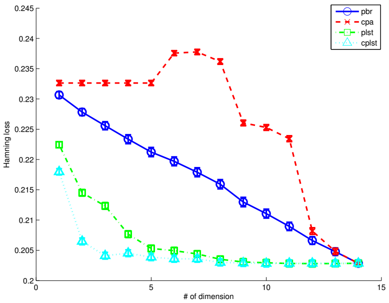
mediamill (Linear Regression)



mediamill (Decision Tree)

- 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



yeast (Linear Regression)

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

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 (RA*k*EL; 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 RA*k*EL (& binary relevance)

Key Idea: Redundant Information

General Error-correcting Codes (ECC)

1011 → 1011010 ⟹ (noisy channel) 1001010 → 1011

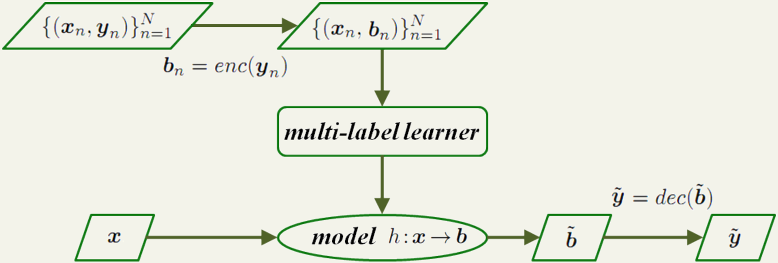
- commonly used in communication systems
- detect & correct errors after transmitting data over a noisy channel
- encode data redundantly

ECC for Machine Learning (successful for multi-class classification)

1011 → 1011010 ⟹ (predictions of **b**) 1001010 → 1011

learn redundant bits $\Longrightarrow$ **correct** prediction **errors**

Proposed Framework: Multi-labeling with ECC



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

PLST: $M \ll L$ (works for large L);
MLECC: $M > L$ (works for small L)

Simple Theoretical Guarantee

ECC encode + Larger Multi-label Learning + ECC decode

Theorem

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

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

PLST: principal directions + decent regression

MLECC: which ECC balances **strength** & **difficulty**?

Simplest ECC: Repetition Code

encoding: $\mathbf{y} \in \{0,1\}^L \rightarrow \mathbf{b} \in \{0,1\}^M$

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

$$L = 4, M = 28 : 1010 \longrightarrow \underbrace{1111111}_{\frac{28}{4}=7}000000011111110000000$$

- permute the bits randomly

decoding: $\tilde{\mathbf{b}} \in \{0,1\}^M \rightarrow \tilde{\mathbf{y}} \in \{0,1\}^L$

- **majority vote** on each original bit

$L = 4$, $M = 28$: strength of repetition code (REP) $= 3$

RA*k*EL = REP (code) + a special powerset (channel)

Slightly More Sophisticated: Hamming Code

HAM(7, 4) Code

- $\{0,1\}^4 \to \{0,1\}^7$ 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. $1011 \longrightarrow 1011010$
- strength $= 1$ (weak)

Our Proposed Code: Hamming on Repetition (HAMR)

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

$L = 4$, $M = 28$: 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)

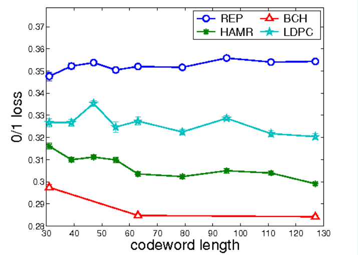
- modern code for **satellite communication**
- connect ECC and Bayesian learning
- approach the theoretical limit in some cases

let's compare!

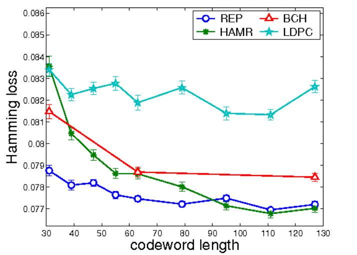
Different ECCs on 3-label Powerset (scene data set w/ $L = 6$)

- learner: special powerset with Random Forests
- REP + special powerset $\approx$ RA*k*EL

0/1 loss



Hamming loss



Comparing to RA*k*EL (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?