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

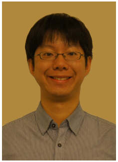
Hsuan-Tien Lin

National Taiwan University

11/6/2012

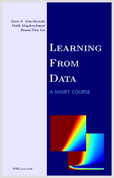
first part: with Farbound Tai, in Neural Computation 2012
second part: with Yao-Nan Chen, to appear in NIPS 2012

A Short Introduction



Hsuan-Tien Lin

- Associate Professor (2012–present; Assistant Professor 2008-2012), Dept. of CSIE, National Taiwan University
- Leader of the Computational Learning Laboratory
- Co-author of the new introductory ML textbook "*Learning from Data: A Short Course*"

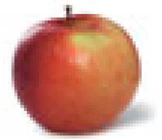


goal: make machine learning more realistic

- multi-class cost-sensitive classification: in ICML '10 (Haifa), 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**s**?

?: $\{$orange, strawberry, kiwi$\}$



apple

orange

strawberry

kiwi

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

Binary Relevance: Multi-label Classification via Yes/No

Binary Classification

{yes, no}

Multi-label w/ L classes: L yes/no questions

apple (N), orange (Y), strawberry (Y), kiwi (Y)

- **Binary Relevance** approach:
  transformation to **multiple isolated binary classification**
- disadvantages:
  - **isolation**—hidden relations (if any) between labels not exploited
  - **unbalanced**—few yes, many no
  - **scalability**—training time $O(L)$

Binary Relevance: simple (& good) benchmark with known disadvantages

Multi-label Classification Setup

Input

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), $\mathcal{Y}_n \subseteq \{1, 2, \cdots, 4\}$

Output

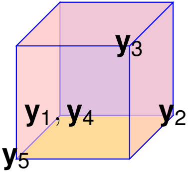
a multi-label classifier $g(\mathbf{x})$ that **closely predicts** the label-set $\mathcal{Y}$ associated with some **unseen** inputs $\mathbf{x}$ (by exploiting hidden relations between labels)

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

multi-label classification: hot and important

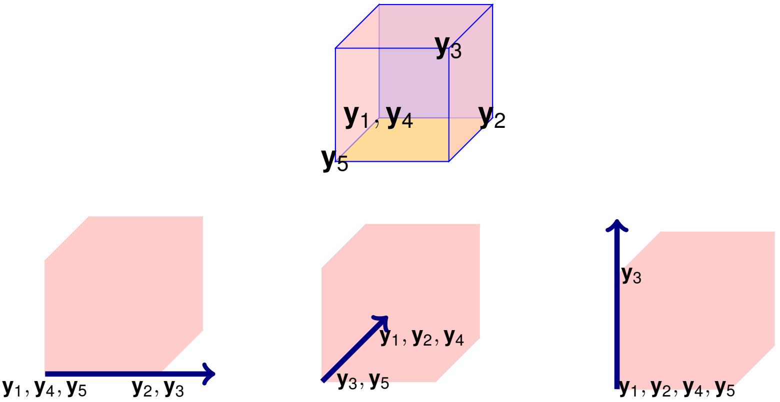
Geometric View of Multi-label Classification

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



subset $\mathcal{Y}$ of $2^{\{1,2,\cdots,L\}}$ $\Leftrightarrow$ **vertex of hypercube** $\{0, 1\}^L$

Geometric Interpretation of Binary Relevance



Binary Relevance: project to L **natural axes** & classify
—can we project to other $M(\ll L)$ directions & learn?

A Label Space Dimension Reduction Approach

General Compressive Sensing

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

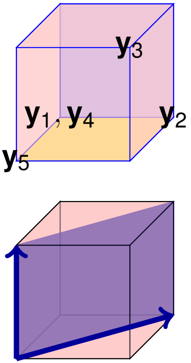
Compressive Sensing for Multi-label Classification (Hsu et al., 2009)

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

Compressive Sensing:

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

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

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

For any given "prediction on subspace" $\mathbf{r}(\mathbf{x})$,

- find sparse binary vector that **P**-projects closest to $\mathbf{r}(\mathbf{x})$: slow —optimization of ℓ_1-regularized objective
- find any binary vector that **P**-projects closest to $\mathbf{r}(\mathbf{x})$: fast

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

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 dimension reduction (**why?**)

Novel Theoretical Guarantee

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

If $g(\mathbf{x}) = round(\mathbf{P}^T\mathbf{r}(\mathbf{x}))$ *(&* $\mathbf{p}_m$ *orthogonal to each other),*

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

- $\|\mathbf{r}(\mathbf{x}) - \mathbf{c}\|^2$: 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 dimension reduction (**indeed**)

Proposed Approach: Principal Label Space Transform

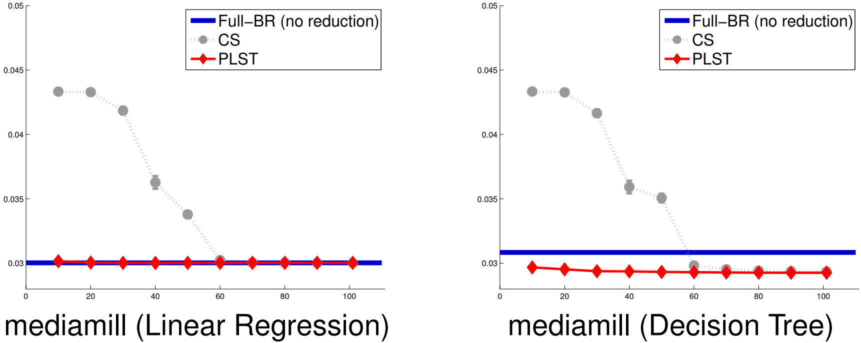
From Compressive Sensing to PLST

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 $\{\mathbf{y}_n\}_{n=1}^N$ —BTW, improvements when shifting $\mathbf{y}_n$ by its estimated mean
- physical meaning behind $\mathbf{p}_m$: key (linear) label correlations

PLST: improving CS by projecting to **key correlations**

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** (can we do even better?)

Theoretical Guarantee of PLST Revisited

Linear Transform + Learn + Round-based Decoding

Theorem (Tai and Lin, 2012)

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

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

- $\|\mathbf{y} - \mathbf{P}^T\mathbf{c}\|^2$: 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 even better by minimizing **jointly**?)

The In-Sample Optimization Problem

$$\min_{\mathbf{r},\mathbf{P}} \left(\underbrace{\|\mathbf{r}(\mathbf{X}) - \mathbf{P}\mathbf{Y}\|^2}_{\text{learn}} + \underbrace{\|\mathbf{Y} - \mathbf{P}^T\mathbf{P}\mathbf{Y}\|^2}_{\text{compress}} \right)$$

- start from a well-known tool, linear regression, as **r**

$$\mathbf{r}(\mathbf{X}) = \mathbf{X}\mathbf{W}$$

- for fixed **P**: a closed-form solution for learn is

$$\mathbf{W}^* = \mathbf{X}^\dagger\mathbf{P}\mathbf{Y}$$

- substitute $\mathbf{W}^*$ to objective function, then **...**

optimal **P**:

for learn	top eigenvectors of $\mathbf{Y}^T(\mathbf{I} - \mathbf{X}\mathbf{X}^\dagger)\mathbf{Y}$
for compress	top eigenvectors of $\mathbf{Y}^T\mathbf{Y}$
for both	top eigenvectors of $\mathbf{Y}^T\mathbf{X}\mathbf{X}^\dagger\mathbf{Y}$

Proposed Approach: Conditional Principal Label Space Transform

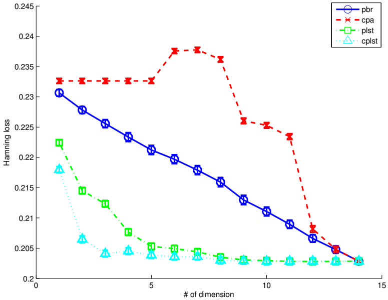
From PLST to CPLST

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

- conditional principal directions: top eigenvectors of $\mathbf{Y}^T\mathbf{X}\mathbf{X}^\dagger\mathbf{Y}$
- physical meaning behind $\mathbf{p}_m$: 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)**

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** (even decision trees)

Conclusion

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?