Advances in Multi-label Classification

Jesse Read

Department of Signal Theory and Communications
Universidad Carlos III
Madrid, Spain

Universidad de Málaga
June 21, 2011

Introduction

Multi-label classification is the supervised classification task where each data instance may be associated with *multiple* class labels.

Introduction

Multi-label classification is the supervised classification task where each data instance may be associated with *multiple* class labels. Given a predefined set of class-labels, e.g. $\mathcal{L} = \{\texttt{beach}, \texttt{trees}, \texttt{urban}, \texttt{people}\}$ and a set of instances from an input domain, e.g. :

- Multi-class (single-label) Classification: examples are associated with a *single* class label; e.g. $\texttt{beach}$.
- Multi-label Classification: examples are associated with a label *subset*: e.g. $\{\texttt{beach}, \texttt{trees}\}$.

Notation

- Instance $\mathbf{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$
- Class labels: $\mathcal{L} = \{1, 2, \ldots, L\}$
- Label space: $\mathcal{Y} = \{0, 1\}^L$
- Labelset: $\mathbf{y} = [y_1, \ldots, y_L] \in \mathcal{Y}$; $y_j = 1$ if jth label relevant to $\mathbf{x}$; else 0)
- Training set: $\{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \ldots, N\} \subset (\mathcal{X} \times \mathcal{Y})$
- Classification: $h : \mathcal{X} \to \mathcal{Y}$
- Prediction: $\hat{\mathbf{y}} = h(\mathbf{x})$; or
  $\hat{\mathbf{w}} = h(\mathbf{x})$, where $\hat{w}_j \in [0, 1]$; then $\hat{\mathbf{y}} = f_t(\hat{\mathbf{w}})$; $\hat{y}_j = 1_{\hat{w}_j \geq t}$

Notation

- Instance $\mathbf{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$
- Class labels: $\mathcal{L} = \{1, 2, \ldots, L\}$
- Label space: $\mathcal{Y} = \{0, 1\}^L$
- Labelset: $\mathbf{y} = [y_1, \ldots, y_L] \in \mathcal{Y}$; $y_j = 1$ if jth label relevant to $\mathbf{x}$; else 0)
- Training set: $\{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \ldots, N\} \subset (\mathcal{X} \times \mathcal{Y})$
- Classification: $h : \mathcal{X} \to \mathcal{Y}$
- Prediction: $\hat{\mathbf{y}} = h(\mathbf{x})$; or
  $\hat{\mathbf{w}} = h(\mathbf{x})$, where $\hat{w}_j \in [0, 1]$; then $\hat{\mathbf{y}} = f_t(\hat{\mathbf{w}})$; $\hat{y}_j = 1_{\hat{w}_j \geq t}$
- Evaluation:
  - example $\hat{\mathbf{y}}_i = \mathbf{y}_i$ (*labelset accuracy*); OR
  - label $\hat{y}_{ij} = y_{ij}$ of example i (*label accuracy*).

Notation

- Instance $\mathbf{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$
- Class labels: $\mathcal{L} = \{1, 2, \ldots, L\}$
- Label space: $\mathcal{Y} = \{0, 1\}^L$
- Labelset: $\mathbf{y} = [y_1, \ldots, y_L] \in \mathcal{Y}$; $y_j = 1$ if jth label relevant to $\mathbf{x}$; else 0)
- Training set: $\{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \ldots, N\} \subset (\mathcal{X} \times \mathcal{Y})$
- Classification: $h : \mathcal{X} \to \mathcal{Y}$
- Prediction: $\hat{\mathbf{y}} = h(\mathbf{x})$; or
  $\hat{\mathbf{w}} = h(\mathbf{x})$, where $\hat{w}_j \in [0, 1]$; then $\hat{\mathbf{y}} = f_t(\hat{\mathbf{w}})$; $\hat{y}_j = 1_{\hat{w}_j \geq t}$
- Evaluation:
  - example $\hat{\mathbf{y}}_i = \mathbf{y}_i$ (*labelset accuracy*); OR
  - label $\hat{y}_{ij} = y_{ij}$ of example i (*label accuracy*).
  - choose a threshold t for $f_t(\cdot)$

Datasets and Statistics

	N	L	$(\sum \mathbf{y})/N$	$uniq.\mathbf{y}$	Type
Music	593	6	1.87	0.046	media
Scene	2407	6	1.07	0.006	media
Yeast	2417	14	4.24	0.082	biology
Genbase	661	27	1.25	0.048	biology
Medical	978	45	1.25	0.096	medical text
Slashdot	3782	22	1.18	0.041	news
Lang.Log	1460	75	1.18	0.208	forum
Enron	1702	53	3.38	0.442	e-mail
Reuters(avg)	6000	103	1.46	0.147	news
Ohsumed	13929	23	1.66	0.082	medical text
tmc2007	28596	22	2.16	0.047	text
Media Mill	43907	101	4.38	0.149	media
Bibtex	7395	159	2.40	0.386	text
IMDB	120919	28	2.00	0.037	text
del.icio.us*	16105	983	19.02	0.981	text

Challenges

Multi-label learning challenges:

- discovering and modelling label dependencies
- dimensionality (output space of 2^L instead of L)
- measures of evaluation / loss functions

Label Dependence

Label *independence* if:

$$p(\mathbf{Y}) = \prod_{j=1}^{L} p(Y_j)$$

This should never be the case!

Label Dependence

Label *independence* if:

$$p(\mathbf{Y}) = \prod_{j=1}^{L} p(Y_j)$$

This should never be the case!

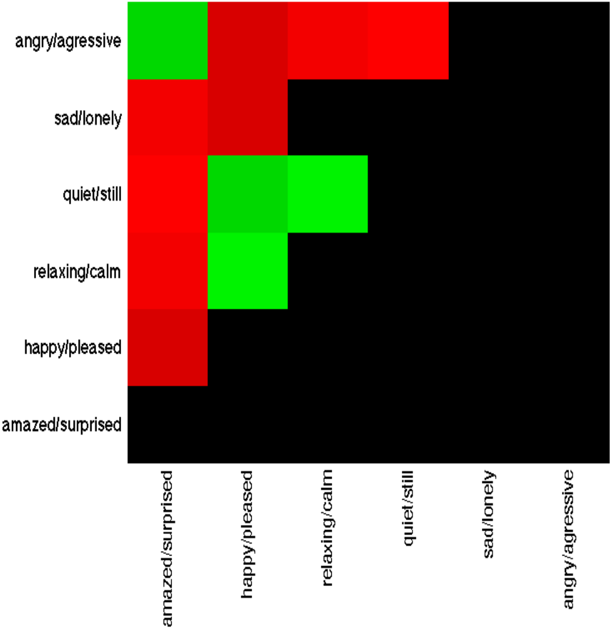
- Unconditional dependence: $P(y_1|y_2)$
- Conditional dependence: $P(y_1|y_2, \mathbf{x})$

It has been widely acknowledged that:

- there are *dependencies* (i.e., correlations) between labels; and
- modelling them *improves predictive performance*; but
- is inherently *expensive* ($\frac{L(L-1)}{2}$ *pairwise*, 2^L *all*).

GOAL: discover *significant* dependencies, and model them appropriately.

Emotions Dataset - Unonditional (In)Dependence



Conditional dependence.
For pairs of labels (y_j, y_k), where

- BR models $y_j \in \{0, 1\}, y_k \in \{0, 1\}$
- FW models $y_{jk} \in \{00, 01, 10, 11\}$

Table: Synthetic data with strong conditional dependence and independence.

	Conditional Dependence		Conditional Independence	
	FW	BR	FW	BR
Subset Acc.	**0.77**	0.70	0.84	**0.89**
Labelset Acc.	**0.45**	0.38	0.59	**0.61**
Label Acc.	**0.94**	0.92	0.97	**0.98**

- Wherever there is label *independence*, BR is the best option.
- Modelling very weak/non-existent label dependencies can be detrimental and, it's computationally expensive! (L vs. $L(L-1)/2$ in this case).

Problem Transformation

Transform a multi-label problem into single-label (binary/multi-class) problems

- Flexible, general, can be more scalable
- Can use any off-the-shelf single-label classifier (kNN, Decision Trees, SVMs, Naive Bayes, *etc.*)

Problem Transformation

Transform a multi-label problem into single-label (binary/multi-class) problems

- Flexible, general, can be more scalable
- Can use any off-the-shelf single-label classifier (kNN, Decision Trees, SVMs, Naive Bayes, *etc.*)

For example:

- Binary Relevance (BR): one binary classifier for each label
- Label Powerset (LP): every label*set* is a single class-label in a multi-class problem
- Copy+Threshold (CT): one L-class multi-class problem, where posterior probabilities are used to decided on multiple labels (e.g. using a threshold).
- Pairwise Classification (PW): decision boundary between each label; essentially a version of CT.

Algorithm Adaptation

Adapt an existing single-label classifier for multi-label classification.

- Usually problem-specific.
- Often use problem transformation internally.

Algorithm Adaptation

Adapt an existing single-label classifier for multi-label classification.

- Usually problem-specific.
- Often use problem transformation internally.

For example:

- Decision Trees, e.g. CLUS [Blockeel et al., 2006]
- kNN, e.g. MLkNN [Zhang and Zhou, 2007]
- Probabilistic, e.g. [McCallum, 1999]

Label Powerset (LP) Transformation

Each combination becomes a single class-label.
If $\mathcal{L} = \{1, 2, \ldots, 6\}$ then we have class-labels $\{000000, 000001, \ldots, 111111\}$ (2^6 in total).

- Usually good performance, but
- worst-case complexity $\min(2^L, N)$ classes; and
- issues with label sparsity and overfitting.

Improving LP

RAndom k-labEL Subsets (RAkEL) [Tsoumakas and Vlahavas, 2007]:

- Train m LP classifiers on label sets $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m$ where each $\mathcal{L}_l \subset \mathcal{L}$ and $|\mathcal{L}_l| = k$; $k < L$.
- e.g. $\mathcal{L}_1 = \{1, 3, 4\}, \mathcal{L}_2 = \{2, 3, 6\}, \mathcal{L}_3 = \{3, 5, 6\}$ $(k = 3, m = 3)$
- complexity reduced to $m \times \min(2^k, N)$, reduces label sparsity and overfitting (because of the ensemble)

Improving LP

RAndom *k*-labEL Subsets (RAkEL) [Tsoumakas and Vlahavas, 2007]:

- Train m LP classifiers on label sets $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m$ where each $\mathcal{L}_l \subset \mathcal{L}$ and $|\mathcal{L}_l| = k$; $k < L$.
- e.g. $\mathcal{L}_1 = \{1, 3, 4\}, \mathcal{L}_2 = \{2, 3, 6\}, \mathcal{L}_3 = \{3, 5, 6\}$ $(k = 3, m = 3)$
- complexity reduced to $m \times \min(2^k, N)$, reduces label sparsity and overfitting (because of the ensemble)

Ensembles of Pruned Sets (EPS) [Read et al., 2008]:

- Find *infrequent*/rare labelsets, then create *pruned sets* to replace them; then train LP.
- e.g. $\mathbf{y}_i = [001101] \rightarrow \mathbf{y}_{ia} = [001100], \mathbf{y}_{ib} = [000101]$
- works because because of 'long-tail' effect: 10% of labelsets associated with 90% of examples
- up to two orders of magnitude faster *in practice*; reduces label sparsity and overfitting (in an ensemble)

Binary Relevance (BR) Transformation

Each label is a separate binary problem: $\mathcal{Y}_1, \ldots, \mathcal{Y}_L$ where each $\mathcal{Y}_j = \{0, 1\}$. $\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x})$ where each $\hat{y}_j = h_j(\mathbf{x})$ for $j = 1, \ldots, L$.

- Good time complexity (L binary models); but
- does not explicitly model label correlations (poor prediction).

Improving BR

Meta-BR:

- label output predictions to train a meta BR classifier: $\hat{\mathbf{y}} = \mathbf{h}_{meta}(\mathbf{h}(\mathbf{x}))$.

Labelset Mapped-BR:

- map output to the closest labelset (from training data): $\phi(\mathbf{h}(\mathbf{x})) \mapsto \hat{\mathbf{y}}_i$.

Improving BR

Meta-BR:

- label output predictions to train a meta BR classifier: $\hat{\mathbf{y}} = \mathbf{h}_{meta}(\mathbf{h}(\mathbf{x}))$.

Labelset Mapped-BR:

- map output to the closest labelset (from training data): $\phi(\mathbf{h}(\mathbf{x})) \mapsto \hat{\mathbf{y}}_i$.

Ensembles of Classifier Chains (ECC) [Read et al., 2009]:

- Pass information along a 'chain' of binary classifiers

Improving BR

Meta-BR:

- label output predictions to train a meta BR classifier: $\hat{\mathbf{y}} = \mathbf{h}_{meta}(\mathbf{h}(\mathbf{x}))$.

Labelset Mapped-BR:

- map output to the closest labelset (from training data): $\phi(\mathbf{h}(\mathbf{x})) \mapsto \hat{\mathbf{y}}_i$.

Ensembles of Classifier Chains (ECC) [Read et al., 2009]:

- Pass information along a 'chain' of binary classifiers
- $\hat{y}_1 = h_1(\mathbf{x})$;

Improving BR

Meta-BR:

- label output predictions to train a meta BR classifier: $\hat{\mathbf{y}} = \mathbf{h}_{meta}(\mathbf{h}(\mathbf{x}))$.

Labelset Mapped-BR:

- map output to the closest labelset (from training data): $\phi(\mathbf{h}(\mathbf{x})) \mapsto \hat{\mathbf{y}}_i$.

Ensembles of Classifier Chains (ECC) [Read et al., 2009]:

- Pass information along a 'chain' of binary classifiers
- $\hat{y}_1 = h_1(\mathbf{x})$; $\hat{y}_2 = h_2(\mathbf{x}, \hat{y}_1)$,

Improving BR

Meta-BR:

- label output predictions to train a meta BR classifier: $\hat{\mathbf{y}} = \mathbf{h}_{meta}(\mathbf{h}(\mathbf{x}))$.

Labelset Mapped-BR:

- map output to the closest labelset (from training data): $\phi(\mathbf{h}(\mathbf{x})) \mapsto \hat{\mathbf{y}}_i$.

Ensembles of Classifier Chains (ECC) [Read et al., 2009]:

- Pass information along a 'chain' of binary classifiers
- $\hat{y}_1 = h_1(\mathbf{x})$; $\hat{y}_2 = h_2(\mathbf{x}, \hat{y}_1)$, $\cdots$, $\hat{y}_L = h_L(\mathbf{x}, \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{L-1})$
- i.e. instead of $P(y_j|\mathbf{x})$, model $P(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$
- use in an ensemble of *random* chains
- high performance, and approximately as fast as BR

Recent Work

A graphical model approach[1]:

1. **identify strongest dependencies**
2. form a model, e.g. 3—4—2 1—5 6
3. model $\{00, 01, 10, 11\}$ between each *connected* pair:
   e.g. $3_{(11)}4_{(10)}2$; $1_{(01)}5$; $6_{(0)} \mapsto [0, 0, 1, 1, 1, 0]$
4. much more efficient than modelling for *all* pairs

[1] work with Fernando Perez Cruz, UC3M, Madrid

Recent Work

A graphical model approach[1]:

1. **identify strongest dependencies**
2. form a model, e.g. 3—4—2 1—5 6
3. model $\{00, 01, 10, 11\}$ between each *connected* pair:
   e.g. $3_{(11)}4_{(10)}2$; $1_{(01)}5$; $6_{(0)} \mapsto [0, 0, 1, 1, 1, 0]$
4. much more efficient than modelling for *all* pairs

k-Labelset mapping:

1. identify strongest dependencies
2. form sets, e.g. {3,4,2}; {1,5}; {6}
3. BR classification, e.g. $\hat{\mathbf{w}} = \mathbf{h}(\mathbf{x})$, where each $\hat{w} \in [0, 1]$
4. use Labelset-mapped BR on sets:
   e.g. $\hat{\mathbf{y}} = \left[\phi\left(\hat{w}_3, \hat{w}_4, \hat{w}_2\right), \phi\left(\hat{w}_1, \hat{w}_5\right), \hat{w}_6\right]$
5. ϕ averages across the top k closest mappings (Euclidean distance); avoids overfitting

[1] work with Fernando Perez Cruz, UC3M, Madrid

Recent Point of Interest: Feature Selection

Most methods look at transforming the label space, but not the feature space. For example, BR:

$$\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_L] = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_L(\mathbf{x})]$$

are all features in X relevant to the jth label (of L labels in total)? Probably not!

Recent Point of Interest: Feature Selection

Most methods look at transforming the label space, but not the feature space. For example, BR:

$$\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_L] = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_L(\mathbf{x})]$$

are all features in X relevant to the jth label (of L labels in total)? Probably not!

- ECC_2 [Read et al., 2011]: random feature selection across the ensemble
- LIFT [Zhang, 2011]: clustering analysis to produce label-specific features

Recent Point of Interest: Data Streams

Data instances arrive continuously and theoretically infinitely.

- incremental nature (labels / label combinations come and go over time – an issue for LP-methods)
- concept drift (label dependencies also evolve over time, and at different rates – not just label concepts)

Recent Point of Interest: Data Streams

Data instances arrive continuously and theoretically infinitely.

- incremental nature (labels / label combinations come and go over time – an issue for LP-methods)
- concept drift (label dependencies also evolve over time, and at different rates – not just label concepts)

Batch-incremental multi-label learning [Qu et al., 2009]:

- can use LP-based methods with SVMs, etc; but
- must parameterise batch size w, initial buffer, etc.
- can only learn from w examples; and only every w examples.

Recent Point of Interest: Data Streams

Data instances arrive continuously and theoretically infinitely.

- incremental nature (labels / label combinations come and go over time – an issue for LP-methods)
- concept drift (label dependencies also evolve over time, and at different rates – not just label concepts)

Batch-incremental multi-label learning [Qu et al., 2009]:

- can use LP-based methods with SVMs, etc; but
- must parameterise batch size w, initial buffer, etc.
- can only learn from w examples; and only every w examples.

Instance-incremental multi-label learning [Read et al., 2010]:

- incremental ECC, EPS, E-HT-PS; using Hoeffding Trees
    - 'preloading' PS
- concept drift monitors (ADWIN)
    - monitoring error rate, or label combinations
    - restart models when drift detected

Summary

- Important to model label dependence;
- but can be computationally expensive; so
- model it only where necessary, and appropriately!
- Ensemble methods work well, but
- reduce redundancy.
- Special considerations for data streams.

Blockeel, H., Schietgat, L., Struyf, J., Clare, A., and Dzeroski, S. (2006).
Hierarchical multilabel classification trees for gene function prediction (extended abstract).
In *Workshop on Probabilistic Modeling and Machine Learning in Structural and Systems Biology*, Tuusula, Finland.

Elisseeff, A. and Weston, J. (2001).
A kernel method for multi-labelled classification.
In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press.

Godbole, S., Sarawagi, S., and Chakrabarti, S. (2002).
Scaling multi-class support vector machines using inter-class confusion.
In *KDD '02: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 513–518.

McCallum, A. K. (1999).
Multi-label text classification with a mixture model trained by EM.
In *Association for the Advancement of Artificial Intelligence workshop on text learning*.

Qu, W., Zhang, Y., Zhu, J., and Qiu, Q. (2009).
Mining multi-label concept-drifting data streams using dynamic classifier ensemble.
In *ACML*.

Read, J., Bifet, A., Holmes, G., and Pfahringer, B. (2010).
Efficient multi-label classification for evolving data streams.
Technical report, University of Waikato, Hamilton, New Zealand.
Working Paper 2010/04.

Read, J., Pfahringer, B., and Holmes, G. (2008).
Multi-label classification using ensembles of pruned sets.
In *ICDM'08: Eighth IEEE International Conference on Data Mining*, pages 995–1000. IEEE.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009).
Classifier chains for multi-label classification.
In *ECML '09: 20th European Conference on Machine Learning*, pages 254–269. Springer.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2011).
Classifier chains for multi-label classification.
Machine Learning.