Multi-label Classification

Jesse Read

Department of Signal Theory and Communications
Madrid, Spain

II MLKDD
São Carlos, Brazil. July 16, 2013

Table of Contents

Outline

Label Dependence II

What do we mean when we speak of dependence (correlation, relationships, etc.)?

Unconditional dependence

The joint is **not** the product of the marginals.

$$p(Y_j, Y_k) \neq p(Y_j)p(Y_k)$$

(i.e., there *is* dependence between the j-th and k-th labels)

Conditional dependence

. . . conditioned on the inputs $\mathbf{x}$.

$$p(Y_j, Y_k|\mathbf{x}) \neq p(Y_j|\mathbf{x})p(Y_k|\mathbf{x})$$

(i.e., there *is* conditional dependence between the j-th and k-th labels)

Label Dependence II

Example

A joint distribution $\mathbf{p}(X, Y_1, Y_2)$.

	x	y_1	y_2	$\mathbf{p}(x, y_1, y_2)$
	0	0	0	0.25
	0	0	1	0
	0	1	0	0
	0	1	1	0.25
	1	0	0	0
	1	0	1	0.25
	1	1	0	0.25
	1	1	1	0
$p(y_j = 1) = \sum_x p(y_j \mid x)$		0.5	0.5	1.0

Example from [Dembczyński et al., 2010].

Label Dependence II

Example

A joint distribution $\mathbf{p}(X, Y_1, Y_2)$.

	x	y_1	y_2	$\mathbf{p}(x, y_1, y_2)$
	0	0	0	0.25
	0	0	1	0
	0	1	0	0
	0	1	1	0.25
	1	0	0	0
	1	0	1	0.25
	1	1	0	0.25
	1	1	1	0
$p(y_j = 1) = \sum_x p(y_j \vert x)$		0.5	0.5	1.0

Is there unconditional independence?

$p(y_1 = 0, y_2 = 0) = p(y_1 = 0)p(y_2 = 0) = 0.25$ (YES!)

Example from [Dembczyński et al., 2010].

Label Dependence II

Example

A joint distribution $\mathbf{p}(X, Y_1, Y_2)$.

	x	y_1	y_2	$\mathbf{p}(x, y_1, y_2)$
	0	0	0	0.25
	0	0	1	0
	0	1	0	0
	0	1	1	0.25
	1	0	0	0
	1	0	1	0.25
	1	1	0	0.25
	1	1	1	0
$p(y_j = 1) = \sum_x p(y_j \mid x)$		0.5	0.5	1.0

Is there conditional independence?

$\mathbf{p}_{x=1}(y_1 = 0, y_2 = 0) = 0 \neq \prod_j \mathbf{p}_{x=1}(y_j = 0) = 0.5$ (NO!)

Example from [Dembczyński et al., 2010].

Measuring and Making Use of Label Dependence

- Measuring unconditional label dependence: just look at frequencies, use e.g., mutual information:

$$I(Y_j; Y_k) = \sum_{y_j \in \{0,1\}} \sum_{y_k \in \{0,1\}} \log \left(\frac{p(y_j, y_k)}{p(y_j)p(y_k)} \right)$$

- Measuring conditional label dependence . . . more difficult . . .
  - many/noisy input variables ($\mathbf{x} = [x_1, x_2, \ldots, \ldots, \ldots, x_D]$)
  - few examples per label

. . . although perhaps the most appropriate, how to measure it?

Measuring and Making Use of Label Dependence

Measuring conditional label dependence.

Proposition

Suppose we have labels Y_j and Y_k ...

- If conditionally independent, best to model separately
  e.g., train BR on $Y_j \in \{0, 1\}$, $Y_k \in \{0, 1\}$; predict
  $[y_j, y_k] = [h_j(\mathbf{x}), h_k(\mathbf{x})]$
- If *not* conditionally independent, best to model together
  e.g., Train LP on $Y_{j,k} \in \{00, 01, 10, 11\}$; predict $[y_j, y_k] = h_{j,k}(\mathbf{x})$
- Therefore, if LP performs [significantly] better than BR for modelling these labels, there is conditional label dependence between them, and we should learn them together!

Measuring and Making Use of Label Dependence

The best result is often somewhere in between BR and LP

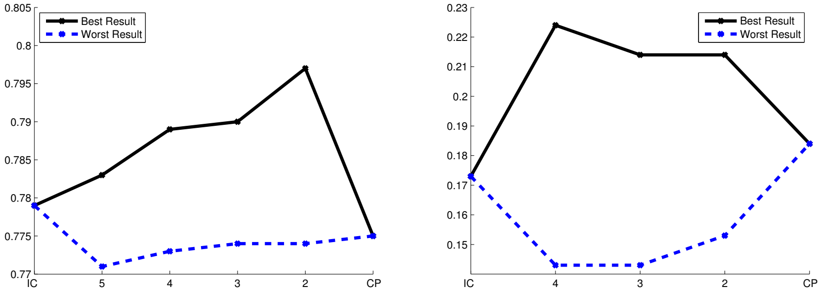


Figure : The best and worst predictive performances for the Music (left) and Parkinson's (right) data for $L, \ldots, 1$ classes, i.e., from BR to LP.

For example, 3 classes: $\{Y_{1,3}, Y_4, Y_{2,5}\}$

Measuring and Making Use of Label Dependence

LPBR [Tenenboim et al., 2010]:

1. Build BR model, e.g., $\mathbf{h} : (h_1, h_2, h_3, h_4, h_5, h_6)$
2. Cluster the most (unconditionally) dependent pair of labels, e.g., Y_2 and Y_5, together
3. Build this model, e.g., $\mathbf{h}' : (h_1, h_{2,5}, h_3, h_4, h_6)$, compare with $\mathbf{h}$ (on some internal evaluation)
4. If $\mathbf{h}'$ performs better, then $\mathbf{h} \leftarrow \mathbf{h}'$
5. Return to Step 2 (Or finish with $\mathbf{h}$)

Example

$$\mathbf{h}(\tilde{\mathbf{x}}) \equiv \left[h_{1,3}(\tilde{\mathbf{x}}), h_4(\tilde{\mathbf{x}}), h_{2,5}(\tilde{\mathbf{x}}), h_6(\tilde{\mathbf{x}})\right]$$

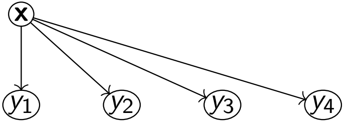
	$Y_{1,3}$	Y_4	$Y_{2,5}$	Y_6
$\hat{\mathbf{y}}$	0, 1	0	0, 0	1

Outline

Binary Relevance (BR): A Probabilistic View

- BR model: $\mathbf{h} = (h_1, \ldots, h_L)$
- each $h_j : \mathcal{X} \to \{0, 1\}$
- for $\tilde{\mathbf{x}}$, predict

$$\begin{aligned}\hat{y}_j &= h_j(\tilde{\mathbf{x}}) \\ &\equiv \operatorname*{argmax}_{y_j \in \{0,1\}} p(y_j | \tilde{\mathbf{x}})\end{aligned}$$



- predictions made independently

$$\mathbf{h}(\tilde{\mathbf{x}}) \equiv [h_1(\tilde{\mathbf{x}}), \ldots, h_L(\tilde{\mathbf{x}})]$$

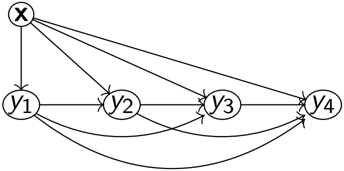
OK, if labels are independent ... but they are not!

$$\boxed{p(\mathbf{y}|\mathbf{x}) \neq \prod_{j=1}^{L} p(y_j|\mathbf{x})}$$

Classifier Chains[1] (CC)

- build $\mathbf{h} = (h_1, \ldots, h_L)$
- each $h_j : \mathcal{X} \times \{0,1\}^{j-1} \to \{0,1\}$
- and, for any $\tilde{\mathbf{x}}$, predict

$$\begin{aligned}\hat{y}_j &= h_j(\tilde{\mathbf{x}}, \hat{y}_1, \ldots, \hat{y}_{j-1}) \\ &\equiv \operatorname*{argmax}_{y_j \in \{0,1\}} p(y_j | \tilde{\mathbf{x}}, \hat{y}_1, \ldots, \hat{y}_{j-1})\end{aligned}$$



- models label dependencies

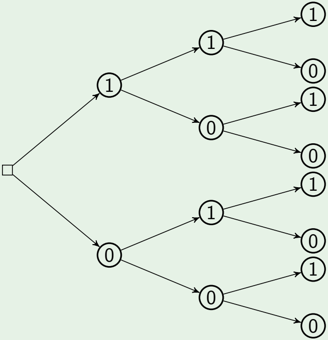
$$\mathbf{h}(\tilde{\mathbf{x}}) = [h_1(\tilde{\mathbf{x}}), h_2(\tilde{\mathbf{x}}, \hat{y}_1), \ldots, h_L(\tilde{\mathbf{x}}, \hat{y}_1, \ldots, \hat{y}_{L-1})]$$

Inspiration from the chain rule (a greedy approximation):

$$\boxed{p(\mathbf{y}|\mathbf{x}) = p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})}$$

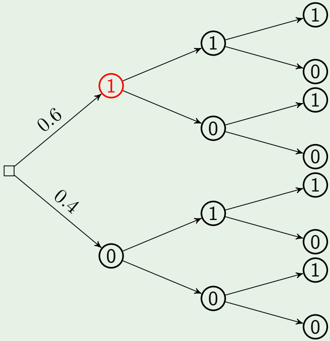
[1] [Read et al., 2009]

Example



$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [?, ?, ?]$

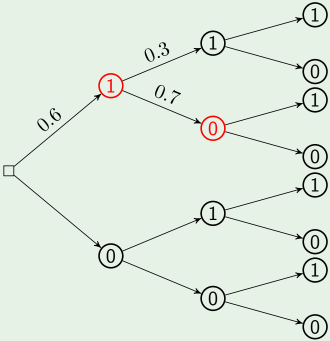
Example



$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [1, ?, ?]$

1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = 1$

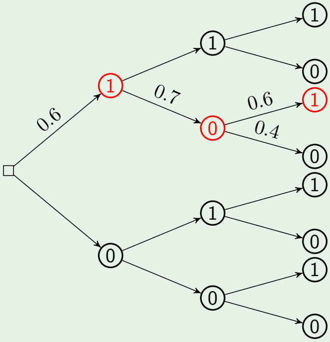
Example



1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = 1$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1) = 0$

$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [1, 0, ?]$

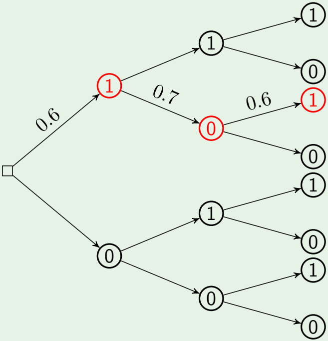
Example



1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = 1$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1) = 0$
3. $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2) = 1$

$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [1, 0, 1]$

Example



1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = 1$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1) = 0$
3. $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2) = 1$

$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [1, 0, 1]$

- similar time complexity to BR in practice (if $L < D$)
- better performance than BR
- can improve (a lot) with Bagging Ensembles of CC (ECC): M CC models, each with a random chain and sample of $\mathcal{D}$.

Bayes Optimal Probabilistic Classifier Chains[2] (PCC)

Issue with CC:

- errors may be propagated down the chain

Bayes-optimal Probabilistic CC, recovers the chain rule:

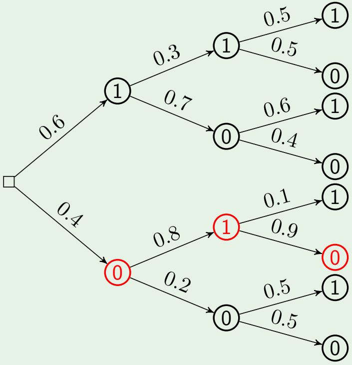
$$\begin{aligned}\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y} \in \{0,1\}^L} p(\mathbf{y}|\mathbf{x}) \\ &= \operatorname*{argmax}_{\mathbf{y} \in \{0,1\}^L} p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})\end{aligned}$$

[2] [Cheng et al., 2010]

Bayes Optimal Probabilistic Classifier Chains[2] (PCC)

Test all possible paths ($\mathbf{y} = [y_1, y_2, \ldots, y_L] \in 2^L$ in total)

Example



1. $p(\mathbf{y} = [0, 0, 0]) = 0.040$
2. $p(\mathbf{y} = [0, 0, 1]) = 0.040$
3. $p(\mathbf{y} = [0, 1, 0]) = 0.288$
4. $\ldots$
5. $p(\mathbf{y} = [1, 1, 1]) = 0.090$

return $\text{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x})$

- better accuracy than CC, but only appropriate for $L \lesssim 15$

[2] [Cheng et al., 2010]

Monte-Carlo search for Classifier Chains (MCC)

MCC [Read et al., 2013]: Sample the tree.

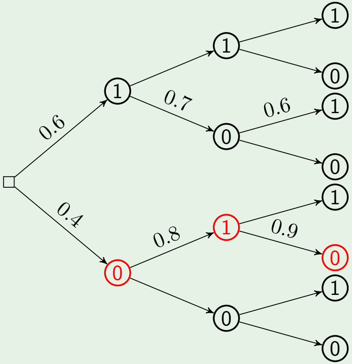
- For $t = 1, \ldots, T$ iterations:
  - sample $\boxed{\mathbf{y}_t \sim p(\mathbf{y}|\mathbf{x})}$ where, for $j = 1, .., L$, $y_j \sim p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$
- Predict

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}_t|t=1,\ldots,T} p(\mathbf{y}_t|\mathbf{x})$$

Monte-Carlo search for Classifier Chains (MCC)

MCC [Read et al., 2013]: Sample the tree.

Example



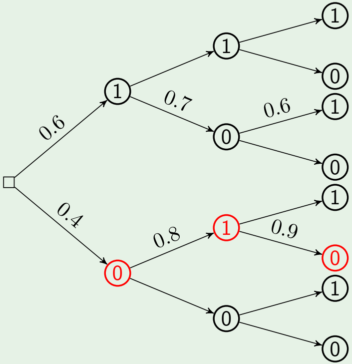
Sample T times ...

- $p([1,0,1]) = 0.6 \cdot 0.7 \cdot 0.6 = 0.252$
- $p([0,1,0]) = 0.4 \cdot 0.8 \cdot 0.9 = 0.288$

return $\mathrm{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x})$

Monte-Carlo search for Classifier Chains (MCC)

Example



Sample T times ...

- $p([1,0,1]) = 0.6 \cdot 0.7 \cdot 0.6 = 0.252$
- $p([0,1,0]) = 0.4 \cdot 0.8 \cdot 0.9 = 0.288$

return $\text{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x})$

- Tractable, unlike PCC (for $T \ll 2^L$), but similar accuracy!

Is the Sequence of Labels in the Chain Important?

Example

$$p(Romance|\tilde{\mathbf{x}})$$
$$p(Comedy|\neg Romance, \tilde{\mathbf{x}})$$
$$p(Action|\neg Romance, \neg Comedy, \tilde{\mathbf{x}})$$
$$\Rightarrow [Action, \neg Comedy, \neg Romance]$$

$$p(Action|\tilde{\mathbf{x}})$$
$$p(Comedy|\neg Action, \tilde{\mathbf{x}})$$
$$p(Romance|\neg Action, Comedy, \tilde{\mathbf{x}})$$
$$\Rightarrow [\neg Action, Comedy, Romance]$$

Different prediction for the same $\tilde{\mathbf{x}}$ (only chain sequence is different)?

Is the Sequence of Labels in the Chain Important?

Example

$$p(Romance|\tilde{\mathbf{x}})$$
$$p(Comedy|\neg Romance, \tilde{\mathbf{x}})$$
$$p(Action|\neg Romance, \neg Comedy, \tilde{\mathbf{x}})$$
$$\Rightarrow [Action, \neg Comedy, \neg Romance]$$

$$p(Action|\tilde{\mathbf{x}})$$
$$p(Comedy|\neg Action, \tilde{\mathbf{x}})$$
$$p(Romance|\neg Action, Comedy, \tilde{\mathbf{x}})$$
$$\Rightarrow [\neg Action, Comedy, Romance]$$

Different prediction for the same $\tilde{\mathbf{x}}$ (only chain sequence is different)?

Define $\mathbf{h_s}$; a Chain Classifier that creates the chain in order $\mathbf{s}$.

- e.g., $\mathbf{s}_1 = [1, 3, 2]$
- e.g., $\mathbf{s}_2 = [2, 1, 3]$

Can we obtain different results for $\mathbf{s}_1$ and $\mathbf{s}_2$?
(Yes! [Read et al., 2013, Kumar et al., 2012])

Is the Sequence of Labels in the Chain Important?

Define $\mathbf{h_s}$; a Chain Classifier that creates the chain in order $\mathbf{s}$.

- e.g., $\mathbf{s}_1 = [1, 3, 2]$
- e.g., $\mathbf{s}_2 = [2, 1, 3]$

Can we obtain different results for $\mathbf{s}_1$ and $\mathbf{s}_2$?
(Yes! [Read et al., 2013, Kumar et al., 2012])

We have, e.g.,

$$y_{s_j} = \underset{\{0,1\}}{\operatorname{argmax}}\, p(y_{s_j} | \mathbf{x}, y_{s_1}, \ldots, y_{s_{j-1}})$$

- Different $\mathbf{s}$ give different results due to finite and noisy data.
- We can walk through the *chain sequences space*; build models $\{\mathbf{h}_{\mathbf{s}_1}\}_{u=1}^{U}$, test against some loss / payoff function $\mathcal{J}(\mathbf{s})$

MCC with **s**-Search ($\mathtt{M_sCC}$)

Monte Carlo Walk through the *chain sequences space*; $\mathbf{s}_1, \ldots, \mathbf{s}_U$; build models $\{\mathbf{h}_{\mathbf{s}_1}\}_{u=1}^{U}$, test against some loss / payoff function, e.g.:

$$\mathcal{J}(\mathbf{s}) := \textsc{ExactMatch}^3(\mathbf{Y}, \mathbf{h}_\mathbf{s}(\mathbf{X}))$$

Example

	u	$\mathbf{s}_u$	$\mathcal{J}(\mathbf{s}_u)$
	0	[4, 2, 0, 1, 3, 5]	0.623
	1	[4, 2, 0, 3, 1, 5]	0.628
	2	[4, 2, 0, 3, 5, 1]	0.638
	3	[4, 0, 2, 3, 5, 1]	0.647
Scene data	5	[4, 0, 5, 2, 3, 1]	0.653
	18	[5, 1, 4, 3, 2, 0]	0.654
	23	[5, 4, 0, 1, 2, 3]	0.664
	128	[3, 5, 1, 0, 2, 4]	0.668
	176	[5, 3, 1, 0, 4, 2]	0.669
	225	[5, 3, 1, 4, 0, 2]	0.670

[3] 1 - [0/1 Loss]

MCC with s-Search (M_sCC)

Example

	u	$\mathbf{s}_u$	$\mathcal{J}(\mathbf{s}_u)$
	0	[4, 2, 0, 1, 3, 5]	0.623
	1	[4, 2, 0, 3, 1, 5]	0.628
	2	[4, 2, 0, 3, 5, 1]	0.638
	3	[4, 0, 2, 3, 5, 1]	0.647
Scene data	5	[4, 0, 5, 2, 3, 1]	0.653
	18	[5, 1, 4, 3, 2, 0]	0.654
	23	[5, 4, 0, 1, 2, 3]	0.664
	128	[3, 5, 1, 0, 2, 4]	0.668
	176	[5, 3, 1, 0, 4, 2]	0.669
	225	[5, 3, 1, 4, 0, 2]	0.670

Use the best $\mathbf{s}_u$ for the final model.

MCC with **s**-Search ($\mathtt{M_sCC}$)

Example

	u	$\mathbf{s}_u$	$\mathcal{J}(\mathbf{s}_u)$
	0	[4, 2, 0, 1, 3, 5]	0.623
	1	[4, 2, 0, 3, 1, 5]	0.628
	2	[4, 2, 0, 3, 5, 1]	0.638
	3	[4, 0, 2, 3, 5, 1]	0.647
Scene data	5	[4, 0, 5, 2, 3, 1]	0.653
	18	[5, 1, 4, 3, 2, 0]	0.654
	23	[5, 4, 0, 1, 2, 3]	0.664
	128	[3, 5, 1, 0, 2, 4]	0.668
	176	[5, 3, 1, 0, 4, 2]	0.669
	225	[5, 3, 1, 4, 0, 2]	0.670

Use the best $\mathbf{s}_u$ for the final model.

- The space is $L!$ large, ...but a little search can go a long way.

MCC with $\mathbf{s}$-Search ($\mathtt{M_sCC}$)

Use the best $\mathbf{s}_u$ for the final model.

Improvements:

- Add temperature to freeze $\mathbf{s}$ from left to right (s_1 to s_L) over time u
  - only need to rebuild $\mathbf{h}_\mathbf{s}$ from the first node changed

$$\mathbf{s}_u = [3, 2, 1, 4, 6, 5]$$
$$\mathbf{s}_{u+1} = [3, 2, 1, 5, 4, 6]$$

  - progressively faster to build each $\mathbf{h}_\mathbf{s}$
- Select a population: $\mathbf{s}_u^{(1)}, \ldots, \mathbf{s}_u^{(M)}$
  - improved predictive performance
  - if each $\mathbf{s}_u^{(m)}$ is random, we recover Ensembles of Classifier Chains ($\mathtt{ECC}$) [Read et al., 2011]

Why not . . .

?

Why not order the chain based on:

- easiest-to-predict labels first
- most-frequent labels first
- unconditional label dependencies: most-‘dependent’ labels last
- conditional dependencies

Why not . . .

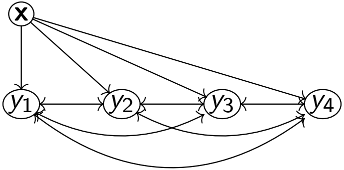
?

Why not order the chain based on:

- easiest-to-predict labels first
- most-frequent labels first
- unconditional label dependencies: most-'dependent' labels last
- conditional dependencies ← this is basically what $\mathtt{M_sCC}$ does!
  - it's a bit slow . . .
  - is there another way to avoid ordering the chain?

Conditional Dependency Networks (CDN)

A fully connected undirected version (no chain sequence) [Guo and Gu, 2011]:



Problem transformation:

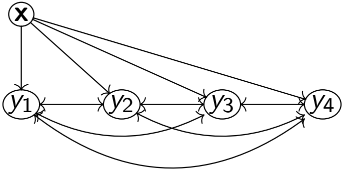
$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

Conditional Dependency Networks (CDN)



Problem transformation:

X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

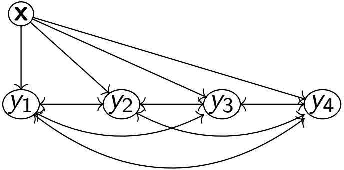
X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

How to do prediction? (where to start from?)

Conditional Dependency Networks (CDN)



Problem transformation:

X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

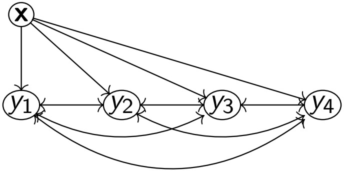
X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

- Gibbs sampling, for $t = 1, \ldots, T$ iterations:

$$y_j \sim p(y_j | \tilde{\mathbf{x}}, y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_L)$$

- collect the marginals $y_1, \ldots, y_L$ at iterations $t = T_{\text{collect}}, \ldots, T$:

Conditional Dependency Networks (CDN)



Problem transformation:

$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

- Avoids need for chain order (no **s**-search, faster training), but
- Inference more expensive.

An Empirical Look

Table : Average predictive performance (5 fold CV, EXACT MATCH[3])

	BR	CC	ECC	PCC	CDN	MCC	$\mathtt{M_sCC}$
params:			$M = 10$		$T = 1000$	$T = 100$	$U = 50$, $M = 10$
Music	0.30	0.29	0.31	0.35	0.30	0.35	**0.37**
Scene	0.54	0.55	0.61	0.64	0.53	0.64	**0.68**
Yeast	0.14	0.15	0.19		0.07	0.21	**0.23**
Genbase	0.94	**0.96**	0.94		0.94	**0.96**	**0.96**
Medical	0.58	0.62	**0.64**		0.60	0.63	0.62
Enron	0.07	0.10	**0.11**		0.07	0.10	0.09
Reuters	0.29	0.35	0.36		0.27	**0.37**	**0.37**
avg. rank	6.14	4.29	3.57		6.43	2.00	1.71

- MCC = PCC's result, but tractable to larger datasets.
- $\mathtt{M_sCC} \succ$ MCC: the chain order makes a difference
- and $\succ$ CDN: can lead to better/faster inference than in a fully connected network
- and $\succ$ ECC ($M = 10$ random chains), but with $\approx 10\times$ less memory

[3] 1-[0/1 LOSS]

An Empirical Look

Table : Average running time (5 fold CV, seconds)

	L	BR	CC	ECC	PCC	CDN	MCC	$\mathrm{M_sCC}$
params:				$M=10$		$T=1000$	$T=100$	$U=50, M=10$
Music	6	0	0	2	1	6	5	18
Scene	6	12	11	44	15	92	90	684
Yeast	14	11	11	66		88	149	731
Genbase	27	11	8	56		573	1695	774
Medical	45	9	11	86		1546	3420	1038
Enron	53	102	92	349		3091	3884	2986
Reuters	101	106	120	1259		14735	1837	4890

- MCC $=$ PCC's result, but tractable to larger datasets.
- $\mathrm{M_sCC} \succ$ MCC: the chain order makes a difference
- and $\succ$ CDN: can lead to better/faster inference than in a fully connected network
- and $\succ$ ECC ($M = 10$ random chains), but with $\approx 10\times$ less memory

From a Chain to a Tree

Why a chain? `CC` (and `MCC`, `PCC`, etc.) can be formulated as:

$$\hat{\mathbf{y}} = p(\mathbf{y}|\tilde{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y}=[y_1,\ldots,y_L]} \prod_{j=1}^{L} p(y_j|\mathbf{pa}_j, \tilde{\mathbf{x}})$$

- If $\mathbf{pa}_j$ (parents of node j) $\equiv \{y_1, \ldots, y_{j-1}\}$ we recover `CC`
- If $\mathbf{pa}_j := \mathbf{s}$ and we recover $\mathtt{M}_{\mathbf{s}}\mathtt{CC}$
- But can define any structure, for example:

$$\begin{tikzcd} y_1 \arrow[r] & y_2 \arrow[r] \arrow[rr, bend right] & y_3 & y_4 \end{tikzcd}$$

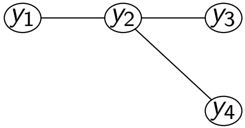
How do we find a good structure?

- label dependencies!
- difficult, but important speed-ups at training and test time!

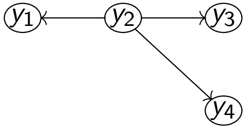
Bayesian Chain Classifiers (BCC)

[Zaragoza et al., 2011]

1. Weight all edges with label dependencies
2. Find a Maximum Spanning Tree (MST)



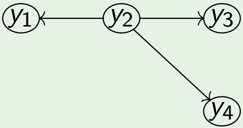
3. Choose some directionality (root node)



4. Employ a CC-type classifier
   Ensembles of BCC: L models, with root notes $j = 1, \ldots, L$

Building Graphs Based on Label Dependency

Example



This is a suitable structure if:

- $Y_4 \perp\!\!\!\perp Y_1 | Y_2$
- $Y_4 \perp\!\!\!\perp Y_3 | Y_2$

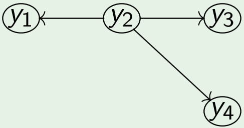
i.e., $P(Y_4|Y_2) \equiv P(Y_4|Y_1, Y_2, Y_3)$

But what about conditional label dependencies?

$$P_{\mathbf{x}}(Y_4|Y_2)$$

Building Graphs Based on Label Dependency

Example



This is a suitable structure if:

- $Y_4 \perp\!\!\!\perp Y_1 | Y_2$
- $Y_4 \perp\!\!\!\perp Y_3 | Y_2$

i.e., $P(Y_4|Y_2) \equiv P(Y_4|Y_1, Y_2, Y_3)$

But what about conditional label dependencies?

$$P_{\mathbf{x}}(Y_4|Y_2)$$

- $\mathtt{M_sCC}$ already does this (for chains)
- Too slow (for large L) !

A Faster Way to Measure Conditional Dependence

LEAD: Learning by Exploiting lAbel Dependency [Zhang and Zhang, 2010]

Proposition

Given two classification problems:

$$y_1 = h_1(\mathbf{x}) + e_1$$

$$y_2 = h_2(\mathbf{x}) + e_2$$

Then: y_1 and y_2 are conditionally independent if e_1 and e_2 are (1) independent *from each other* and (2) independent from $\mathbf{x}$.

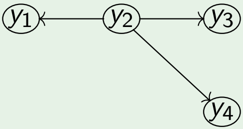
- (2) $\approx$ holds by fitting the model with maximum likelihood
- if (1) two holds, we can claim conditional dependence

A Faster Way to Measure Conditional Dependence

LEAD: Learning by Exploiting lAbel Dependency [Zhang and Zhang, 2010]

1. Train BR $\mathbf{h} : (h_1, \ldots, h_L)$, measure errors $e_1, \ldots, e_L$
2. Learn the Bayesian network structure $\mathcal{G}$ based on $e_1, \ldots, e_L$
3. Construct CC-type $\mathbf{h}$, to predict each $\hat{y}_j$ given $\mathbf{pa}_j, \mathbf{x}$

Example

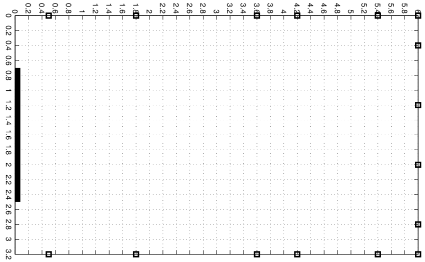


Structured Output Learning as a Multi-label Learning

What if we have many, many labels ...

Example

A structured output problem: detecting the relevant 'pixels' occupied by an object, given some observations

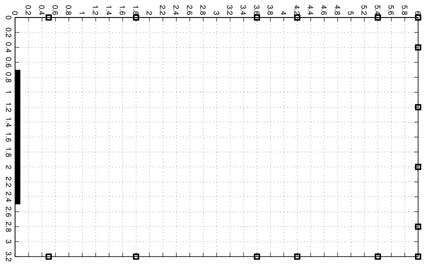


- $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$ just a multi-label problem!
- $\mathbf{x} = [x_1, \ldots, x_D]$ observations / input
- $\mathbf{y} = [y_1, \ldots, y_L]$, where $y_j = 1 \Leftrightarrow j$-th pixel is 'segmented'/relevant

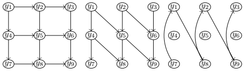
Structured Output Learning as a Multi-label Learning

Example

A structured output problem: detecting the relevant 'pixels' occupied by an object, given some observations



- It does not make sense to use CC here!
- Other methods (BCC, LEAD) may not scale well ...

Use  ...???

Outline

Real-world Considerations for Multi-label Learning

Much of the multi-label literature avoids many real-world issues:

- **Scalability**. We have a lot of data and/or limited resources to spare

Real-world Considerations for Multi-label Learning

Much of the multi-label literature avoids many real-world issues:

- **Scalability**. We have a lot of data and/or limited resources to spare
- **Incremental**. We have to update a model incrementally, we may never see a 'final' training example

Real-world Considerations for Multi-label Learning

Much of the multi-label literature avoids many real-world issues:

- **Scalability**. We have a lot of data and/or limited resources to spare
- **Incremental**. We have to update a model incrementally, we may never see a 'final' training example
- **Concept Drift**. The labelling scheme may change over time (number of labels, label dependencies)

Real-world Considerations for Multi-label Learning

Much of the multi-label literature avoids many real-world issues:

- **Scalability**. We have a lot of data and/or limited resources to spare
- **Incremental**. We have to update a model incrementally, we may never see a 'final' training example
- **Concept Drift**. The labelling scheme may change over time (number of labels, label dependencies)
- **Limited Labelled Data**. Multi-Labelled data can be expensive to obtain, even more so than single-labelled data (whereas unlabelled data is usually easy to get, by the millions . . .).

Real-world Considerations for Multi-label Learning

Much of the multi-label literature avoids many real-world issues:

- **Scalability**. We have a lot of data and/or limited resources to spare
- **Incremental**. We have to update a model incrementally, we may never see a 'final' training example
- **Concept Drift**. The labelling scheme may change over time (number of labels, label dependencies)
- **Limited Labelled Data**. Multi-Labelled data can be expensive to obtain, even more so than single-labelled data (whereas unlabelled data is usually easy to get, by the millions . . .). No label ($y_j = 0$) may not imply negative example of this label

Scalability: Large L (many labels)

- $L > 100 \ldots$
- $L > 1000 \ldots$
- more is not so typical (it becomes another problem)

Scalability: Large L (many labels)

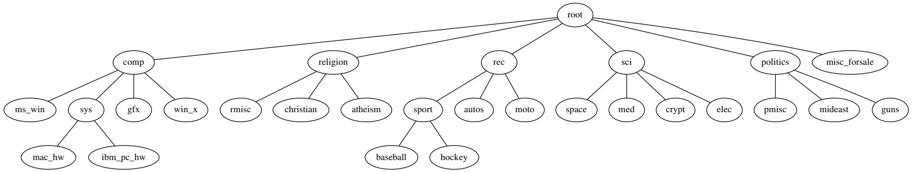
- $L > 100$ ...
- $L > 1000$ ...
- more is not so typical (it becomes another problem)

How to deal with many labels.

- Use a learner/base learner that scales well with L
  - i.e., not LP
  - BR can be distributed easily
- Only model some (the most important) label dependencies
- Take advantage of redundancy in the learning space
  - problem transformation methods may make many copies (e.g., BR/CC: L times) of $\mathbf{X}$ ($D \times N$)
  - ensemble methods make a further M copies (for each model)
  - e.g., [Yan et al., 2007, Read et al., 2011]: random subsets of $\mathcal{D}$, $\mathbf{X}$
- Compress label space, (uncompress after classification)
- Use a hierarchy to break up a problem into smaller subproblems

Hierarchical multi-label classification.

- Some datasets define a hierarchy, e.g., FunCat, Enron, Reuters, *20 Newsgroups*:



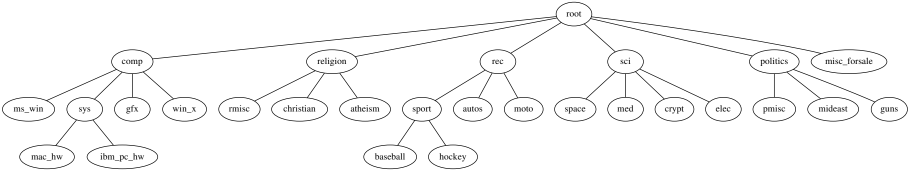
- We can use this predefined hierarchy
  - Classifier at each node, e.g., [Kiritchenko et al., 2006], classifications propagate to the leaves;

$$\mathbf{h}_{\text{root}}(\tilde{\mathbf{x}}) \subseteq \{\texttt{comp}, \texttt{religion}, \texttt{rec}, \texttt{sci}, \texttt{politics}, \texttt{misc\_forsale}\}$$
$$\mathbf{h}_{\text{sci}}(\tilde{\mathbf{x}}) \subseteq \{\texttt{space}, \texttt{med}, \texttt{crypt}, \texttt{elec}\}$$

  - Induction of decision trees for hierarchical multi-label classification [Vens et al., 2008]

Hierarchical multi-label classification.

- Some datasets define a hierarchy, e.g., FunCat, Enron, Reuters, *20 Newsgroups*:



- We can use this predefined hierarchy
  - Classifier at each node, e.g., [Kiritchenko et al., 2006], classifications propagate to the leaves;

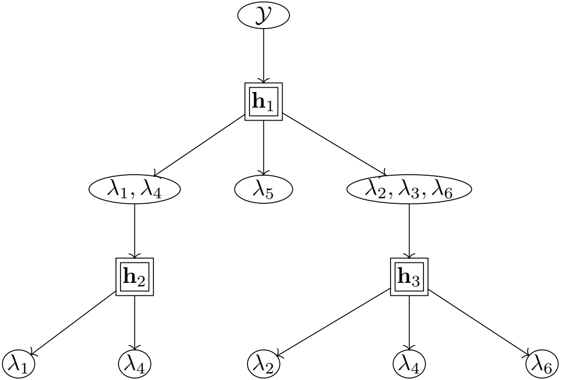
$$\mathbf{h}_{\text{root}}(\tilde{\mathbf{x}}) \subseteq \{\texttt{comp}, \texttt{religion}, \texttt{rec}, \texttt{sci}, \texttt{politics}, \texttt{misc\_forsale}\}$$

$$\mathbf{h}_{\text{sci}}(\tilde{\mathbf{x}}) \subseteq \{\texttt{space}, \texttt{med}, \texttt{crypt}, \texttt{elec}\}$$

  - Induction of decision trees for hierarchical multi-label classification [Vens et al., 2008]
- Or, make up your own hierarchy! `HOMER`: Hierarchy Of Multilabel ClassifiERs [Tsoumakas et al., 2008]

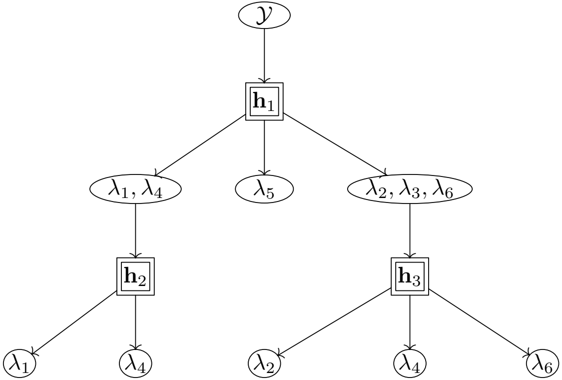
HOMER: Hierarchy Of Multilabel ClassifiERs

For $\mathcal{Y} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6\}$



HOMER: Hierarchy Of Multilabel ClassifiERs

For $\mathcal{Y} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6\}$



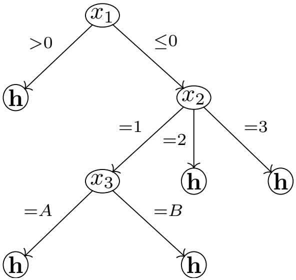
- Either standard, or *balanced* k-means clustering
- Solving several sub problems easier that one big problem
- If well-chosen, little loss in accuracy compared to 'flat' problem
- and can be even be better than pre-defined hierarchy
  (. . . which is just some pre-defined label-dependency information!)

Scalability: Large N (many instances)

1. Use a fast base-classifier
   - e.g., `PW` with perceptrons [Loza Mencía and Fürnkranz, 2008]
   - warning: sometimes behaviour changes unexpectedly under large N
     - ★ for example, the extra input features in `CC` can cause imbalance/overfitting on some problems
2. Multi-label Hoeffding tree
3. Windowed methods

Scalability: Large N (many instances)

1. Use a fast base-classifier
2. Multi-label Hoeffding tree
   - [Read et al., 2012]: Combining the multi-label entropy of `ML-C4.5` with incremental Very Fast Decision Trees
   - A multi-label incremental classifier (e.g., $\mathbf{h} := \texttt{PS}^3$ with NB) at the leaves



   - Can handle up to 10 million instances in hours (single core)
3. Windowed methods

[3] Reminder: a tractable version of `LP`

Scalability: Large N (many instances)

1. Use a fast base-classifier
2. Multi-label Hoeffding tree
3. Windowed methods
   - e.g., `2BR` with `C4.5` in batches [Qu et al., 2009]
   - e.g., multi-label k`NN` on a window of as many instances as possible/appropriate

Scalability: Large N (many instances)

1. Use a fast base-classifier
2. Multi-label Hoeffding tree
3. Windowed methods

But scalability may not be the only consideration . . .

Classification in Data Streams

In **data streams**, there are additional considerations:

- New data instances arrive continually, in sequence;
  - we need a prediction now!
  - we need to update the model incrementally
  - including threshold calibration, etc.
- And potentially infinitely;
  - but resources are finite
  - there is no 'final' training/test instance
- The concept is usually dynamic, and may change over time.
  - label dependencies may change
  - new labels may be created / eliminated

Examples of Multi-label Data Streams

- E-mail
- News
- Forums
- Labels for music, images, documents, etc.

Examples of Multi-label Data Streams

- E-mail
- News
- Forums
- Labels for music, images, documents, etc.

In fact, most of the data we deal with already fits the characteristics of a data stream, even personal document collections:

	$\mathcal{X}$ (data inst.)	$\mathcal{Y}$ (labels)	L	N	D	LC
Music	audio data	emotions	6	593	72	1.87
Scene	image data	scene labels	6	2407	294	1.07
Yeast	genes	biological fns	14	2417	103	4.24
Genbase	genes	biological fns	27	661	1185	1.25
Medical	medical text	diagnoses	45	978	1449	1.25
Enron	e-mails	labels, tags	53	1702	1001	3.38
Reuters	news articles	categories	103	6000	500	1.46
TMC07	textual reports	errors	22	28596	500	2.16
Ohsumed	medical articles	disease cats.	23	13929	1002	1.66
IMDB	plot summaries	genres	28	120919	1001	2.00
20NG	posts	news groups	20	19300	1006	1.03
MediaMill	video data	annotations	101	43907	120	4.38
Del.icio.us	bookmarks	tags	983	16105	500	19.02

Classification in Data Streams

Data-stream classifiers should

- Learn in an online/incremental fashion
  - May include "batch-incremental" methods (maintain a set of batches) but these can have serious disadvantages

Classification in Data Streams

Data-stream classifiers should

- Learn in an online/incremental fashion
  - May include "batch-incremental" methods (maintain a set of batches) but these can have serious disadvantages
- Be efficient

Classification in Data Streams

Data-stream classifiers should

- Learn in an online/incremental fashion
  - May include "batch-incremental" methods (maintain a set of batches) but these can have serious disadvantages
- Be efficient
- Detect concept drift, e.g., changes in:
  - $P(\mathbf{x})$
  - $P(y_j)$
  - $P(y_j|\mathbf{x})$
  - $P(\mathbf{y})$
  - $P(y_j, y_k)$
  - $P(y_j, y_k|\mathbf{x})$

Detecting and Dealing with Concept Drift in Multi-label Data Streams

Detection methods:

- Don't bother – just forget old information regularly
  - 'batch-incremental': drop old batches/models
  - *k*NN sliding window: keep *w* instances in memory
- Use an off-the-shelf drift detection monitor, e.g., [Read et al., 2012] using `ADWIN` [Bifet and Gavaldà, 2007]
  - Feed this monitor some statistic, e.g., ACCURACY
  - Use an ensemble; replace a model when drift is detected
    - weakest model
    - oldest model
  - `BR`-based methods: Replace model of the poorly-performing label
- Multi-label Bayesian Network Classifier [Borchani, 2013]
  - Page-Hinkley test to detect drift
  - adapts the network around each changed node, or starts anew

Open Questions in Multi-label Data Streams

- Modelling label dependency over time
- Addition of new labels arrive / phasing out old labels
- Getting more real-world data
  - we may not have access to the data
  - time consuming to parse
  - difficult to obtain labelled data, most data is unlabelled!

Future Directions in Multi-label Classification

Predictive performance starting to plateau, even for datasets with a small number of labels, e.g., on the Music data:

$\approx$ year	2007	2010	2013
$\approx$ state-of-the-art EXACT MATCH	0.30	0.35	0.37

This suggests that

- at this rate we will not reach/surpass human performance
- modelling label dependencies has its limits
- we need to learn more about features

Future Directions in Multi-label Classification

Predictive performance starting to plateau, even for datasets with a small number of labels, e.g., on the Music data:

≈ year	2007	2010	2013
≈ state-of-the-art EXACT MATCH	0.30	0.35	0.37

This suggests that

- at this rate we will not reach/surpass human performance
- modelling label dependencies has its limits
- we need to learn more about features

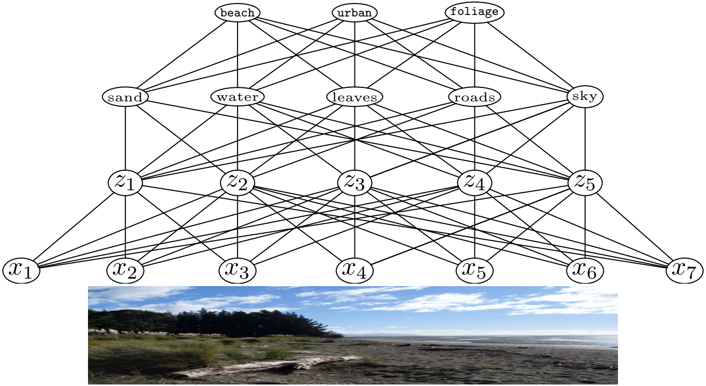
Where to next?

- learn better features
- semi-supervised learning

We can borrow from other areas . . .

What we want

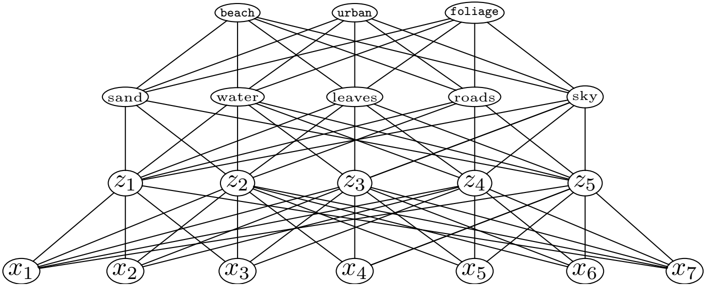
$\hat{Y} = \{\texttt{beach}, \texttt{foliage}\}$



Predicting the labels should be easy (given the right features)!

What we want

$\hat{Y} = \{\texttt{beach}, \texttt{foliage}\}$



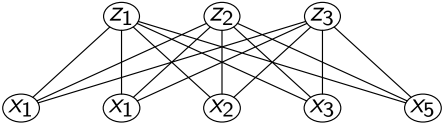
Predicting the labels should be easy (given the right features)!

- We should integrate features in our model, not just *label* dependency
- We already have BPMLL, but it's not particularly competitive

Deep Learning with Restricted Boltzmann Machines[3]

Useful to Multi-label Classification?

A Restricted Boltzmann Machine (RBM). From D input units $x_1, \ldots, x_D$, produce H hidden units $z_1, \ldots, z_H$:



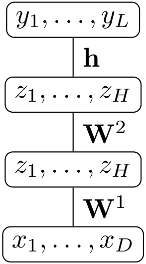
Stack together to make deep belief networks.

- Unsupervised (learns from unlabelled examples)
- Learns a better / more compact feature space
- Incremental (*contrastive divergence*, similar to gradient descent)

[3] See, e.g., [Hinton and Salakhutdinov, 2006]

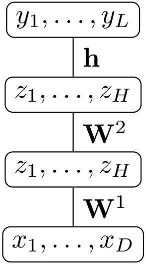
Deep Belief Networks for Multi-label Classification (?)

1. Learn any multi-label classifier **h** ontop of new feature space (to predict labels)
2. Run **h** $\equiv$ `BPMLL`-type algorithm, back propagation of error through weights **W**s
3. Model the features and labels together



Deep Belief Networks for Multi-label Classification (?)

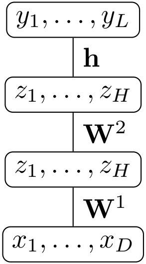
1. Learn any multi-label classifier **h** ontop of new feature space (to predict labels)
2. Run **h** $\equiv$ `BPMLL`-type algorithm, back propagation of error through weights **W**s
3. Model the features and labels together



Possible contribution to multi-label classification:

- memory reduction (from D to H units) – especially beneficial to multi-label problem transformation methods like `BR`, `PW`!
- learn incrementally
- learn from unlabelled examples

Deep Belief Networks for Multi-label Classification (?)



1. Learn any multi-label classifier **h** ontop of new feature space (to predict labels)
2. Run **h** $\equiv$ `BPMLL`-type algorithm, back propagation of error through weights **W**s
3. Model the features and labels together

Possible contribution to multi-label classification:

- memory reduction (from D to H units) – especially beneficial to multi-label problem transformation methods like `BR`, `PW`!
- learn incrementally
- learn from unlabelled examples

Although, currently:

- tuning for hyper parameters is fiddly
- doesn't work well on all datasets

Summary of Current / Future Trends

The days of:

> *"Our novel algorithm X beats BR [or algorithm Y] by modelling label correlations [more efficiently]."*

are becoming more difficult.

We may see more intersection with other areas

- graphical models
- feature generation
- structured output learning
- transfer learning
- semi-supervised learning
- data-stream mining

and application to more challenging problems.

Outline

Summary

- Multi-label Data
- Multi-label Classification
  - problem transformation
  - algorithm adaptation
- Multi-label Evaluation
- Label Dependency
- Advanced Methods
- Advanced Topics
- Open Questions and Future Directions

Conclusions

Multi-label Classification

- A hot topic in machine learning
- Many real-world applications
- Connections to many related areas

Acknowledgemets

Colleagues / Co-Authors:

- (at Yahoo!) Albert Bifet
- (at UC3M) Luca Martino, David Luengo, Pablo Olmos, Fernando Perez-Cruz
- (at UPM) Concha Bielza and Pedro Larrañaga
- (at Waikato) Bernhard Pfahringer, Geoff Holmes, Eibe Frank

Outline

Key References

- Tsoumakas, G. and Katakis, I. (2007). Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13
- Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010). Mining multi-label data. In Maimon, O. and Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*. 2nd edition, Springer
- Madjarov, G., Kocev, D., Gjorgjevikj, D., and Deroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104
- Zhang, M.-L. and Zhou, Z.-H. (2013). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1
- Read, J. (2010). *Scalable Multi-label Classification*. PhD thesis, Department of Computer Science. University of Waikato
- Borchani, H. (2013). *Multi-dimensional classification using Bayesian networks for stationary and evolving streaming data*. PhD thesis, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid

Resources

Software:

- MULAN (WEKA-based library):
  - http://mulan.sourceforge.net
- MEKA (WEKA-based framework):
  - http://meka.sourceforge.net
- CLUS (Decision-Tree/Rule Learning):
  - http://clus.sourceforge.net
- Matlab Code (MLkNN, BPMLL):
  - http://lamda.nju.edu.cn/datacode/MLkNN.htm
- MOA (Data Streams):
  - http://moa.cs.waikato.ac.nz/

Resources

Datasets:

- http://mulan.sourceforge.net/datasets.html
- http://meka.sourceforge.net/#datasets

Tutorials:

- Min-Ling Zhang, MLA'10
  http://lamda.nju.edu.cn/conf/mla10/files/zhangml.pdf
- Eyke Hüllermeier, MLD'10 http://cse.seu.edu.cn/conf/mld10/files/mld10_invitedtalk.pdf
- Concha Bielza, Pedro Larrañaga, U.P.M. http://ocw.upm.es/ciencia-de-la-computacion-e-inteligencia-artificial/machine-learning/contenidos/11.Multilabel.pdf
- This one! http://www.tsc.uc3m.es/~jesse/

End of Part 2

Multi-label Classification

Jesse Read

Department of Signal Theory and Communications
Madrid, Spain

II MLKDD
São Carlos, Brazil. July 16, 2013