Multi-label Classification with Classifier Chains

Jesse Read

Aalto University School of Science,
Department of Information and Computer Science
and Helsinki Institute for Information Technology
Helsinki, Finland

Helsinki. March 28, 2014
1. Introduction: Multi-label Classification

2. Classifier Chains

3. Classifier ‘Trees’ and ‘Graphs’

4. Reflection, Summary, and Future Work
Introduction: Multi-label Classification

**Binary classification**: Is this a picture of a beach? $\in \{\text{yes, no}\}$

**Multi-class classification**: Which class does this picture belong to?

$\in \{\text{beach, sunset, foliage, field, mountain, urban}\}$

**Multi-label classification**: Which labels are relevant to this picture?

$\subseteq \{\text{beach, sunset, foliage, field, mountain, urban}\}$

i.e., each instance can have multiple labels instead of a single one!
Introduction: Single-label vs. Multi-label

Table: Single-label $Y \in \{0, 1\}$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>

Build classifier $h$, such that $\hat{y} = h(\tilde{x})$. 
**Introduction: Single-label vs. Multi-label**

**Table**: Multi-label $Y_1, \ldots, Y_L \in 2^L$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>$Y_4$</th>
<th>$Y_5$</th>
<th>$Y_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Build classifier(s) $h$ or $h$, such that $\hat{y} = [y_1, \ldots, y_L] = h(\tilde{x})$. 
Introduction: Another Example

Table: The IMDB Dataset

<table>
<thead>
<tr>
<th>example</th>
<th>X_1</th>
<th>X_2</th>
<th>...</th>
<th>X_{1000}</th>
<th>X_{1001}</th>
<th>Y_1</th>
<th>Y_2</th>
<th>...</th>
<th>Y_{27}</th>
<th>Y_{28}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>120919</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dataset</td>
<td>Type</td>
<td>Labels</td>
<td>L</td>
<td>N</td>
<td>D</td>
<td>LC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
<td>-------------------------</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Music</td>
<td>audio data</td>
<td>emotions</td>
<td>6</td>
<td>593</td>
<td>72</td>
<td>1.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scene</td>
<td>image data</td>
<td>scene labels</td>
<td>6</td>
<td>2407</td>
<td>294</td>
<td>1.07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yeast</td>
<td>genes</td>
<td>biological fns</td>
<td>14</td>
<td>2417</td>
<td>103</td>
<td>4.24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genbase</td>
<td>genes</td>
<td>biological fns</td>
<td>27</td>
<td>661</td>
<td>1185</td>
<td>1.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical</td>
<td>medical text</td>
<td>diagnoses</td>
<td>45</td>
<td>978</td>
<td>1449</td>
<td>1.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enron</td>
<td>e-mails</td>
<td>labels, tags</td>
<td>53</td>
<td>1702</td>
<td>1001</td>
<td>3.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reuters</td>
<td>news articles</td>
<td>categories</td>
<td>103</td>
<td>6000</td>
<td>500</td>
<td>1.46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMC07</td>
<td>textual reports</td>
<td>errors</td>
<td>22</td>
<td>28596</td>
<td>500</td>
<td>2.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ohsumed</td>
<td>medical articles</td>
<td>disease cats.</td>
<td>23</td>
<td>13929</td>
<td>1002</td>
<td>1.66</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMDB</td>
<td>plot summaries</td>
<td>genres</td>
<td>28</td>
<td>120919</td>
<td>1001</td>
<td>2.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20NG</td>
<td>posts</td>
<td>news groups</td>
<td>20</td>
<td>19300</td>
<td>1006</td>
<td>1.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MediaMill</td>
<td>video data</td>
<td>annotations</td>
<td>101</td>
<td>43907</td>
<td>120</td>
<td>4.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Del.icio.us</td>
<td>bookmarks</td>
<td>tags</td>
<td>983</td>
<td>16105</td>
<td>500</td>
<td>19.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **L**: number of labels
- **N**: number of examples
- **D**: number of input feature attributes
- **LC**: Label Cardinality $\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} y_{j}^{(i)}$ (average number of labels per example)
Evaluation Metrics

Compare prediction $\hat{y}^{(i)} = h(\tilde{x}^{(i)}) = [\hat{y}_1, \ldots, \hat{y}_L]$ with true labels $y^{(i)}$.

- **0/1 LOSS**: label vectors must match exactly

$$0/1 \text{ LOSS}^1 := \frac{1}{N} \sum_{i=1}^{N} I[\hat{y}^{(i)} \neq y^{(i)}]$$

- **Hamming loss**: predicting all 0s will incur relatively little loss

$$\text{Hamming loss} := \frac{1}{NL} \sum_{i=1}^{N} \sum_{j=1}^{L} I[\hat{y}_j^{(i)} \neq y_j^{(i)}]$$

- It is usually not possible to minimize both at the same time:

- For general evaluation, use multiple and contrasting evaluation measures; other measures: Jaccard Index, F-measure, (can be micro or macro averaged).

---

$^1$Often framed as ExactMatch := $1 - 0/1$ loss
Related Applications

Multi-target / multi-output / multi-dimensional classification / regression

Table: Multi-output; each ‘output’ $Y_j \in \{1, \ldots, K\}$ or $Y_j \in \mathbb{R}$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>$Y_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>F</td>
<td>4</td>
<td>A</td>
<td>0.3</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>M</td>
<td>2</td>
<td>B</td>
<td>0.2</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>M</td>
<td>2</td>
<td>A</td>
<td>0.4</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>F</td>
<td>3</td>
<td>C</td>
<td>0.8</td>
</tr>
</tbody>
</table>
**Structured Output Prediction**

Figure: Structured learning: A multi-label problem with large $L$; underlying *structure*. Here we want to segment the *relevant* ‘pixels’ $y \in 2^L$ occupied by object(s), given some sensor observations $x = [x_1, \ldots, x_d]$. 
$L$ independent models (one for each label): $\mathbf{h} = (h_1, \ldots, h_L)$; where each $h_j : \mathcal{X} \rightarrow \{0, 1\}$

- For input $\tilde{x}$, predict \textit{independently}:

  $$\hat{y}_j = h_j(\tilde{x})$$

  $$\equiv \arg\max_{y_j \in \{0, 1\}} p(y_j | \tilde{x})$$

(probabilistically speaking, although $h_j$ can be any off-the-shelf binary classifier: SVMs, Decision Trees, etc.)

Thus: $\hat{y} = [\hat{y}_1, \ldots, \hat{y}_L] = \mathbf{h}(\tilde{x}) = [h_1(\tilde{x}), \ldots, h_L(\tilde{x})]$
Binary Relevance (BR)

BR may perform poorly. In real multi-label data,

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

Example

In the IMDB dataset,

$$P(y_{adult} = 1, y_{family} = 1) = 0$$

whereas

$$P(y_{adult} = 1)P(y_{family} = 1) > 0$$
Main Challenges in Multi-label Learning

Typical multi-label paper:

“The BR method does not model label co-occurrences / correlations / dependencies. We present a method which does [efficiently] and outperforms BR [and other multi-label methods].”

The main challenge has been to

1. model label dependencies; and
2. do this efficiently.
Classifier Chains\(^2\) (CC)

Inspiration from the chain rule

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

- Build \(h = (h_1, \ldots, h_L)\); each
  \(h_j : \mathcal{X} \times \{0, 1\}^{j-1} \to \{0, 1\}\)
- For any \(\tilde{x}\), predict
  \[
  \hat{y}_j = h_j(\tilde{x}, \hat{y}_1, \ldots, \hat{y}_{j-1}) \\
  \equiv \arg\max_{y_j \in \{0, 1\}} p(y_j|\tilde{x}, \hat{y}_1, \ldots, \hat{y}_{j-1})
  \]
- CC is a greedy approximation; similar complexity to BR.

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

\(^2\)[Read et al., 2009], MLJ

---

Jesse Read (Aalto/HiIT)
\[ \hat{y} = h(\tilde{x}) = [?, ?, ?] \]
Example

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

\[ \hat{y}_1 = h_1(\tilde{x}) = \arg\max_{y_1} p(y_1|\tilde{x}) = 1 \]
\[ \hat{y} = h(\tilde{x}) = [1, 0, ?] \]
\[ \hat{y} = h(\tilde{x}) = [1, 0, 1] \]

1. \( \hat{y}_1 = h_1(\tilde{x}) = \text{argmax}_{y_1} p(y_1 | \tilde{x}) = 1 \)
2. \( \hat{y}_2 = h_2(\tilde{x}, \hat{y}_1) = \ldots = 0 \)
3. \( \hat{y}_3 = h_3(\tilde{x}, \hat{y}_1, \hat{y}_2) = \ldots = 1 \)

Improves over BR; similar build time (if \( L < D \)); parallelizable; able to use any off-the-shelf classifier as \( h_j \). But, errors may be propagated down the chain.
Example

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

1. \(\hat{y}_1 = h_1(\tilde{x}) = \arg\max_{y_1} p(y_1|\tilde{x}) = 1\)
2. \(\hat{y}_2 = h_2(\tilde{x}, \hat{y}_1) = \ldots = 0\)
3. \(\hat{y}_3 = h_3(\tilde{x}, \hat{y}_1, \hat{y}_2) = \ldots = 1\)

- Improves over BR; similar build time (if \(L < D\)); parallelizable; able to use any off-the-shelf classifier as \(h_j\)
- But, errors may be propagated down the chain
Bayes-optimal Probabilistic CC, recovers the chain rule, predicts

\[ \hat{y} = \arg\max_{y \in \{0,1\}^L} p(y|x) \]

\[ = \arg\max_{y \in \{0,1\}^L} \left\{ p(y_1|x) \prod_{j=2}^{L} p(y_j|x, y_1, \ldots, y_{j-1}) \right\} \]

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

---

[Dembczyński et al., 2010], ICML’10
Bayes Optimal Probabilistic Classifier Chains (PCC)

Example

\[ p(y = [0, 0, 0]) = 0.040 \]
\[ p(y = [0, 0, 1]) = 0.040 \]
\[ p(y = [0, 1, 0]) = 0.288 \]
\[ \ldots \]
\[ p(y = [1, 0, 1]) = 0.252 \]
\[ \ldots \]
\[ p(y = [1, 1, 1]) = 0.090 \]

\[
\text{return argmax}_y p(y | \tilde{x})
\]

- Better accuracy than CC, but only appropriate for \( L \lesssim 15 \)

\[ [\text{Dembczyński et al., 2010}, \text{ICML'10} \]
Monte-Carlo search for Classifier Chains\(^4\) (MCC)

**MCC: Sample** the ‘chain’.

1. **For** \( t = 1, \ldots, T \) iterations:
   - **Sample** \( y_t \sim p(y|x) \)
     1. \( y_1 \sim p(y_1|x) \) // \( y_1 = 1 \) with probability \( p(y_1|x) \)
     2. \( y_2 \sim p(y_2|x, y_1, y_2) \)
     3. \ldots
     4. \( y_L \sim p(y_L|x, y_1, \ldots, y_{L-1}) \)

2. **Predict**

\[
\hat{y} = \arg\max_y \ p(y_t|x) \quad y_t|t=1,\ldots,T
\]

---

\(^4\) [Read et al., 2013b], Pattern Recognition
Monte-Carlo search for Classifier Chains (MCC)

MCC: Sample the ‘chain’.

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 $\text{argmax}_{y_t} p(y_t|x)$

---

[Read et al., 2013b], Pattern Recognition
Monte-Carlo search for Classifier Chains\(^4\) (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 \( \arg\max_{y_t} p(y_t|x) \)

- **Tractable**, unlike PCC (for \( T \ll 2^L \)); but similar accuracy (\( \succ \) CC).

\(^4\) [Read et al., 2013b], Pattern Recognition
Is the Sequence of Labels in the Chain Important?

Are these models equivalent?

Are these models equivalent?

\[ \text{vs} \]

\[\text{vs}\]

\[y_4 \rightarrow y_2 \rightarrow y_3 \rightarrow y_1 \]

\[y_4 \rightarrow y_2 \rightarrow y_3 \rightleftharpoons y_1 \]

\[y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow y_4 \]

\[y_1 \rightarrow y_3 \rightarrow y_2 \rightarrow y_4 \]

\[\text{At least, not necessarily} \quad \text{[Kumar et al., 2013, Read et al., 2013a]}\]
Are these models equivalent?

No\(^5\). Although

\[ p(y_2|x)p(y_1|y_2, x) = p(y_1|x)p(y_2|y_1, x) \]

\(p\) is only an estimated distribution; from finite and noisy data. And

\[ p(y_1|x)p(y_2|\hat{y}_1, x), \text{ etc} \ldots \]

\(^5\)At least, not necessarily [Kumar et al., 2013, Read et al., 2013a]
MCC with $s$-Search$^6$ ($M_sCC$)

Monte Carlo walk through the space of chain sequences $s = [s_1, \ldots, s_L]$

For $u = 1, \ldots, U$:

1. build MCC on chain sequence $s_u$
2. test against some loss/payoff function $\mathcal{J}(s_u)$; accept if better (if $\mathcal{J}(s_u) > \mathcal{J}(s_{u-1})$)

Use $h_{s_U}$ as the final model.

---

$^6$[Read et al., 2013a], Pattern Recognition
MCC with $s$-Search\textsuperscript{7} ($M_s$CC)

Monte Carlo walk through the space of chain sequences $s = [s_1, \ldots, s_L]$

- The space is $L!$ large, ... but a little search can go a long way.
- Can add temperature to freeze $s_u$ from left to right over time
- Can use a population of chain sequences: $s_u^{(1)}, \ldots, s_u^{(M)}$
- Another approach is to use ‘beam search’ \textsuperscript{6}

\textsuperscript{6} Beam search algorithms for multi-label learning [Kumar et al., 2013], MLJ
\textsuperscript{7} [Read et al., 2013a], Pattern Recognition
### Average predictive performance (5 fold CV, Exact Match)

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>BR</th>
<th>CC</th>
<th>PCC</th>
<th>MCC</th>
<th>M&lt;sub&gt;S&lt;/sub&gt;CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>params:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Music</td>
<td>6</td>
<td>0.30</td>
<td>0.29</td>
<td>0.35</td>
<td>0.35</td>
<td>0.37</td>
</tr>
<tr>
<td>Scene</td>
<td>6</td>
<td>0.54</td>
<td>0.55</td>
<td>0.64</td>
<td>0.64</td>
<td>0.68</td>
</tr>
<tr>
<td>Yeast</td>
<td>14</td>
<td>0.14</td>
<td>0.15</td>
<td>0.21</td>
<td>0.23</td>
<td></td>
</tr>
<tr>
<td>Genbase</td>
<td>27</td>
<td>0.94</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>Medical</td>
<td>45</td>
<td>0.58</td>
<td>0.62</td>
<td>0.63</td>
<td>0.62</td>
<td></td>
</tr>
<tr>
<td>Enron</td>
<td>53</td>
<td>0.07</td>
<td>0.10</td>
<td>0.10</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>Reuters</td>
<td>101</td>
<td>0.29</td>
<td>0.35</td>
<td>0.37</td>
<td>0.37</td>
<td></td>
</tr>
</tbody>
</table>

- MCC = PCC, but tractable to larger datasets.
- M<sub>S</sub>CC \succ MCC: the chain order makes a difference
An Empirical Look

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

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>BR</th>
<th>CC</th>
<th>PCC</th>
<th>MCC</th>
<th>MsCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>params:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T = 100, U = 50</td>
</tr>
<tr>
<td>Music</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>Scene</td>
<td>6</td>
<td>12</td>
<td>44</td>
<td>15</td>
<td>90</td>
<td>684</td>
</tr>
<tr>
<td>Yeast</td>
<td>14</td>
<td>11</td>
<td>66</td>
<td>149</td>
<td>731</td>
<td></td>
</tr>
<tr>
<td>Genbase</td>
<td>27</td>
<td>11</td>
<td>56</td>
<td>1695</td>
<td>774</td>
<td></td>
</tr>
<tr>
<td>Medical</td>
<td>45</td>
<td>9</td>
<td>86</td>
<td>3420</td>
<td>1038</td>
<td></td>
</tr>
<tr>
<td>Enron</td>
<td>53</td>
<td>102</td>
<td>349</td>
<td>3884</td>
<td>2986</td>
<td></td>
</tr>
<tr>
<td>Reuters</td>
<td>101</td>
<td>106</td>
<td>1259</td>
<td>1837</td>
<td>4890</td>
<td></td>
</tr>
</tbody>
</table>

- \( \text{MCC} = \text{PCC} \), but **tractable to larger datasets.**
- \( \text{MsCC} \succ \text{MCC} \): the chain order makes a difference
- although a little **slower** . . .
Why not order the chain based on:

- Easiest-to-predict labels first
- Most-frequent labels first
- Most-‘dependent’ labels first/last (marginal dependence)
- Empirical performance (i.e., conditional dependence, $M_sCC$)

Performance can be improved most by modelling label dependence.
Label Dependence

Generally, methods model

- **Marginal** dependence (e.g., stacked-BR\(^8\))
- **Conditional** dependence (e.g., MsCC\(^9\))
- **Random** dependence (e.g., RAkEL\(^{10}\))
- **No** dependence (e.g., BR)

---

\(^8\) Discriminative Methods for Multi-labeled Classification [Godbole and Sarawagi, 2004]

\(^9\) Monte Carlo Methods for ... Classifier Chains [Read et al., 2013b]

\(^{10}\) RAndom \(k\)-labEL subsets for Multi-label Classification [Tsoumakas and Vlahavas, 2007]
Marginal vs. Conditional Dependence

**Marginal dependence**
When the joint is **not** the product of the marginals.

\[ p(y_2) \neq p(y_2 | y_1) \]

- Measure the frequencies of co-occurrences in the training data

**Conditional in/dependence**

\[ p(y_2 | y_1, x) \neq p(y_2 | x) \]

- Have to build and measure models / take into account the input space
From a Chain to a Tree

Why a chain (sequence)? We can formulate any structure, with

\[ \hat{y} = p(y|x) = \arg\max_y \prod_{j=1}^L p(y_j|\text{pa}_j, \tilde{x}) \]

where \( \text{pa}_j \) = parents of node \( j \).

- If \( \text{pa}_j := \{y_1, \ldots, y_{j-1}\} \) we recover CC

How do we find a good structure?
- label dependence!
- difficult to find, but can benefit accuracy, train/test time.
Bayesian Chain Classifiers\textsuperscript{11} (BCC)

Employ CC in a ‘tree’ (a ‘Classifier Tree’):

1. Weight all edges with (marginal) label dependencies

\begin{center}
\begin{tikzpicture}
  \node (y1) at (0,0) {$y_1$};
  \node (y2) at (1,0) {$y_2$};
  \node (y3) at (2,0) {$y_3$};
  \node (y4) at (3,0) {$y_4$};
  \path
    (y1) edge (y2)
    (y2) edge (y3)
    (y3) edge (y4)
;\end{tikzpicture}
\end{center}

2. Find a \textbf{maximum spanning tree} (MST)

3. Choose some directionality (a root node)

4. Employ any classifier, e.g., CC with Naive Bayes

\footnote{\textsuperscript{11}Zaragoza et al., 2011, IJCAI 11; and related [Alessandro et al., 2013], ‘Ensemble of Bayes Nets’ for MLC, IJCAI 13 using standard message passing for inference – complexity permitting}
Bayesian Chain Classifiers\textsuperscript{11} (BCC)

Employ CC in a ‘tree’ (a ‘\textbf{Classifier Tree}’):

1. Weight all edges with (marginal) label dependencies
2. Find a \textbf{maximum spanning tree} (MST)

\[
\begin{align*}
&y_1 \\ &\quad \rightarrow \quad y_2 \\ &\quad \quad \rightarrow \quad y_3 \\ &\quad \quad \quad \rightarrow \quad y_4
\end{align*}
\]

3. Choose some directionality (a root node)
4. Employ any classifier, e.g., CC with Naive Bayes

\textsuperscript{11}[Zaragoza et al., 2011], IJCAI 11; and related [Alessandro et al., 2013], ‘Ensemble of Bayes Nets’ for MLC, IJCAI 13 using standard message passing for inference – complexity permitting
Bayesian Chain Classifiers\textsuperscript{11} (BCC)

Employ CC in a ‘tree’ (a ‘Classifier Tree’):

1. Weight all edges with (marginal) label dependencies
2. Find a maximum spanning tree (MST)
3. Choose some directionality (a root node)
4. Employ any classifier, e.g., CC with Naive Bayes

---
\textsuperscript{11} [Zaragoza et al., 2011], IJCAI 11; and related [Alessandro et al., 2013], ‘Ensemble of Bayes Nets’ for MLC, IJCAI 13 using standard message passing for inference – complexity permitting.
Bayesian Chain Classifiers\textsuperscript{11} (BCC)

Employ CC in a ‘tree’ (a ‘Classifier Tree’):

1. Weight all edges with (marginal) label dependencies
2. Find a \textbf{maximum spanning tree} (MST)
3. Choose some directionality (a root node)
4. Employ any classifier, e.g., CC with Naive Bayes

\[ x \rightarrow y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow y_4 \]

\textsuperscript{11} [Zaragoza et al., 2011], IJCAI 11; and related [Alessandro et al., 2013], ‘Ensemble of Bayes Nets’ for MLC, IJCAI 13 using standard message passing for inference – complexity permitting
Bayesian Chain Classifiers\textsuperscript{11} (BCC)

Employ CC in a ‘tree’ (a ‘Classifier Tree’):

1. Weight all edges with (marginal) label dependencies
2. Find a **maximum spanning tree** (MST)
3. Choose some directionality (a root node)
4. Employ any classifier, e.g., CC with Naive Bayes

Can get comparable accuracy to CC (but not always)

Only uses *marginal / unconditional* dependencies.

---

\textsuperscript{11} Zaragoza et al., 2011, IJCAI 11; and related [Alessandro et al., 2013], ‘Ensemble of Bayes Nets’ for MLC, IJCAI 13 using standard message passing for inference – complexity permitting
**Classifier ‘Graphs’ (∼ Bayesian Network)**

LEAD\textsuperscript{12} uses an efficient method to measure conditional label dependence:

**Proposition**

Given two classification problems (e.g., BR with $L = 2$),

$$y_1 = h_1(x) + e_1 \quad \text{and} \quad y_2 = h_2(x) + e_2$$

the dependence between $e_1, e_2 \approx$ the conditional dependence between $Y_1, Y_2$. 

\textsuperscript{12}LEArning with label Dependence [Zhang and Zhang, 2010], KDD ’10
LEAD\textsuperscript{12} uses an efficient method to measure \textit{conditional} label dependence:

1. train BR, $h_1, \ldots, h_L$
2. measure dependence among \textit{errors}, $e_1, \ldots, e_L$
3. find a directed structure
4. plug in (e.g.,) CC

Basically: measure dependence among $e_j$ instead of $Y_j$.

\textsuperscript{12} ‘LEArning with label Dependence’ [Zhang and Zhang, 2010], KDD ’10
Is it worth it?

Is it better to invest resources in one good model; or many approximate (or even random) models? Perhaps the *main challenge* is actually to

1. model-label dependencies get good multi-label predictions; and
2. do this efficiently.
Getting a good Classifier ‘Graph’

Is it necessary (for best performance) to
- model label dependence?
- ... conditional dependence?
- ... ‘complete’ dependence?

Table: Average Jaccard Index: rank, under 5×CV \([L = 6]\).

<table>
<thead>
<tr>
<th></th>
<th>BR</th>
<th>BCC-(R)</th>
<th>BCC-(M)</th>
<th>BCC-(C)</th>
<th>ECC</th>
<th>(P_s)CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music</td>
<td>0.517</td>
<td>0.545 (5)</td>
<td>0.567 (4)</td>
<td>0.582 (3)</td>
<td>0.588 (2)</td>
<td>0.594 (1)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.595</td>
<td>0.646 (3)</td>
<td>0.646 (3)</td>
<td>0.643 (5)</td>
<td>0.647 (2)</td>
<td>0.705 (1)</td>
</tr>
</tbody>
</table>

- BR: independent classifiers
- BCC with Random structure / based on Marginal and Conditional dependence
- ECC: Ensemble of random CC (complete random dependence)
- \(P_s\)CC: best of all \((6! = 720)\) possible chain orders; Bayes-optimal \((2^6)\) inference.
Getting a good Classifier ‘Graph’

Is it necessary (for best performance) to
- model label dependence? Yes
- ... conditional dependence? Not necessarily
- ... ‘complete’ dependence? It helps, but can be expensive.

Table: Average Jaccard Index: rank, under 5×CV [L = 6].

<table>
<thead>
<tr>
<th></th>
<th>BR</th>
<th>BCC-R</th>
<th>BCC-M</th>
<th>BCC-Ć</th>
<th>ECC</th>
<th>PsCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music</td>
<td>0.517</td>
<td>0.545 (5)</td>
<td>0.567 (4)</td>
<td>0.582 (3)</td>
<td>0.588 (2)</td>
<td>0.594 (1)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.595</td>
<td>0.646 (3)</td>
<td>0.646 (3)</td>
<td>0.643 (5)</td>
<td>0.647 (2)</td>
<td>0.705 (1)</td>
</tr>
</tbody>
</table>

- BR: independent classifiers
- BCC with Random structure / based on Marginal and Conditional dependence
- ECC: Ensemble of random CC (complete random dependence)
- PsCC: best of all (6! = 720) possible chain orders; Bayes-optimal (2^6) inference.
Modelling dependence helps; modelling complete dependence is (unsurprisingly) best, but not always practical: BCC has $L - 1$ ‘links’ in the chain, vs $\frac{L(L-1)}{2}$ for ECC and $M_s$CC. Can either

- find one (or several) good model(s); or
- use many random models.

Although ‘randomly dependent’ models can perform quite well,

- “quite well” ≠ very well,
- they are not so interpretable, and
- are not necessarily the most efficient.
A super label is just a class with $> 2$ possible values, e.g.:

$$Y_{1,4} \in \{00, 10, 01\}$$  
(some values can be pruned)

1. Form super-labels based on dependence
2. Prune values
3. Plug in any multi-output-capable classifier (e.g., CC)

Can make this hierarchical (‘meta labels’), as in HOMER$^{13}$.

---

$^{13}$Tsoumakas et al., 2008, ECML/PKDD 2008
Super-Label Classifier

Figure: Performance (Parkinson’s data) for $L, L-1, \ldots, 2, 1$ classes, i.e., from BR to LP.

Table: Performance on the Enron dataset ($L = 53$).

<table>
<thead>
<tr>
<th></th>
<th>BR</th>
<th>SCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Match</td>
<td>0.121</td>
<td>0.169</td>
</tr>
<tr>
<td>Ham. Loss</td>
<td>0.057</td>
<td>0.054</td>
</tr>
<tr>
<td>Time (s)</td>
<td>43.67</td>
<td>9.02</td>
</tr>
<tr>
<td>Num. Labels</td>
<td>53</td>
<td>6</td>
</tr>
<tr>
<td>Values/label</td>
<td>2</td>
<td>4.5</td>
</tr>
</tbody>
</table>

A model based on label dependence can perform more accurately and much faster (including the time to measure dependence).
Other problem transformation methods include

- Labelset approaches, e.g., [Tsoumakas and Vlahavas, 2007]’s RAKEL
  i.e., casting to a multi-class problem ($\in \{0000, \ldots, 0001, 1111\}$)
- Pairwise, e.g., [Furnkranz et al., 2008]
  i.e., casting to pair-wise problems

Well-known algorithm adaptation methods; include multi-label

- Neural Networks, e.g., [Zhang and Zhou, 2006]
- Decision Trees, e.g., [Clare and King, 2001]
- $k$-Nearest Neighbours, e.g., [Zhang and Zhou, 2007]
- Maximum Margin method, e.g., [Elisseeff and Weston, 2002]
Recent Trends and Challenges

Specific to multi-label learning:

1. ‘Big data’, scalability
   - Thousands to millions of labels

2. Data streams
   - Learning (e.g., label dependencies) incrementally
   - Dealing with concept drift in the label space

3. Missing values and partially/weakly labelled data
   - In multi-label classification we often don’t know that they’re missing!
     (If an image is not labelled foliage, does it have no foliage?)
   - Manually-multi-labelled data is even more expensive to obtain
The area of *multi-label classification* has expanded rapidly over the last few years, and is now overlapping with many related areas.

Most attention has focussed on modelling label dependence.

*Classifier chains* is a family of methods suitable for this but there are many different approaches.
The area of *multi-label classification* has expanded rapidly over the last few years, and is now overlapping with many related areas.

Most attention has focussed on modelling label dependence.

*Classifier chains* is a family of methods suitable for this but there are many different approaches.

It is apparent that . . .

- Efforts to find the perfect label-dependency model aren’t always rewarded.
- Multi-label problems are getting much bigger.
- There are many related open problems, for example, data streams.
- A lot of literature from other areas is very relevant.
Thank you!

Questions?