Goals of the talk

1. To understand the geometry of different approaches for multi-label classification

2. To appreciate how the Machine Learning techniques further improve the multi-label classification methods

3. To learn how to evaluate the multi-label classification methods
Agenda

• Motivation & Problem definition
• Solutions
• Advanced solutions
• Evaluation metrics
• Toolboxes
• Summary
Notation

- \( X \in \mathbb{R}^m \): feature vector variable (input)
- \( Y \in \mathbb{R}^d \): class vector variable (output)
  - \( x = \{x_1, \ldots, x_m\} \): feature vector instance
  - \( y = \{y_1, \ldots, y_d\} \): class vector instance

- In a shorthand, \( P(Y=y|X=x) = P(y|x) \)

- \( D_{\text{train}} \): training dataset; \( D_{\text{test}} \): test dataset
Motivation

• Traditional classification
  
  • Each data instance is associated with a single class variable
Motivation

• An issue with traditional classification
  • In many real-world applications, each data instance can be associated with multiple class variables

• Examples
  • A news article may cover multiple topics, such as politics and economics
  • An image may include multiple objects as building, road, and car
  • A gene may be associated with several biological functions
Problem Definition

- Multi-label classification (MLC)
  - Each data instance is associated with **multiple binary class variables**
  - Objective: assign each instance the **most probable assignment** of the class variables

\[ h : \mathbf{X} \in \mathbb{R}^m \rightarrow \mathbf{Y} \in \{0, 1\}^d \]

Class 1 ∈ { R, B }
Class 2 ∈ { o, △ }
A simple solution

• Idea

• Transform a multi-label classification problem to multiple single-label classification problems

• Learn $d$ independent classifiers for $d$ class variables
Binary Relevance (BR) [Clare and King, 2001; Boutell et al, 2004]

* Idea

- Transform a multi-label classification problem to multiple single-label classification problems
- Learn \( d \) independent classifiers for \( d \) class variables

* Illustration

<table>
<thead>
<tr>
<th>( D_{train} )</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( Y_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n=1 )</td>
<td>0.7</td>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( n=2 )</td>
<td>0.6</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( n=3 )</td>
<td>0.1</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( n=4 )</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( n=5 )</td>
<td>0.8</td>
<td>0.9</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( h_1 \) : \( X \rightarrow Y_1 \)

\( h_2 \) : \( X \rightarrow Y_2 \)

\( h_3 \) : \( X \rightarrow Y_3 \)
Binary Relevance (BR) [Clare and King, 2001; Boutell et al, 2004]

• Advantages
  • Computationally efficient

• Disadvantages
  • Does not capture the dependence relations among the class variables
  • Not suitable for the objective of MLC
    • Does not find the most probable assignment
    • Instead, it maximizes the marginal distribution of each class variable
Binary Relevance (BR) \cite{Clare and King, 2001; Boutell et al, 2004}

- Marginal vs. Joint: a motivating example

- Question: find the most probable assignment (MAP: maximum a posteriori) of $Y = (Y_1, Y_2)$

| $P(Y_1, Y_2 | X=x)$ | $Y_1 = 0$ | $Y_1 = 1$ | $P(Y_2 | X=x)$ |
|---------------------|-----------|-----------|---------------|
| $Y_2 = 0$           | 0.2       | 0.45      | 0.65          |
| $Y_2 = 1$           | 0.35      | 0         | 0.35          |
| $P(Y_1 | X=x)$       | 0.55      | 0.45      |               |

⇒ Prediction on the joint (MAP): $Y_1 = 1$, $Y_2 = 0$
⇒ Prediction on the marginals: $Y_1 = 0$, $Y_2 = 0$

- We want to maximize the joint distribution of $Y$ given observation $X = x$; i.e.,

$$h^*(x) = \arg \max_y P(Y = y | X = x)$$
Another simple solution

• **Idea**
  
  • Transform *each label combination* to *a class value*
  
  • Learn *a multi-class classifier* with the new class values
Label Powerset (LP) [Tsoumakas and Vlahavas, 2007]

- **Idea**
  - Transform each label combination to a class value
  - Learn a multi-class classifier with the new class values

- **Illustration**

<table>
<thead>
<tr>
<th>$D_{train}$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=1$</td>
<td>0.7</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=2$</td>
<td>0.6</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=3$</td>
<td>0.1</td>
<td>0.9</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=4$</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$n=5$</td>
<td>0.8</td>
<td>0.9</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$h_{LP} : X \rightarrow Y_{LP}$
Label Powerset (LP) [Tsoumakas and Vlahavas, 2007]

- **Advantages**
  - Learns the full joint of the class variables
  - Each of the new class values maps to a label combination

- **Disadvantages**
  - The number of choices in the new class can be exponential \( |Y_{LP}| = O(2^d) \)
  - Learning a multi-class classifier on exponential choices is expensive
  - The resulting class distribution would be sparse and imbalanced
  - Only predicts the label combinations that are seen in the training set
BR vs. LP

- BR and LP are two extreme MLC approaches
  - **BR** maximizes the marginals on each class variable; while **LP** directly models the joint of all class variables
  - BR is computationally more efficient; but does not consider the relationship among the class variables
  - LP considers the relationship among the class variables by modeling the full joint of the class variables; but can be computationally very expensive
Agenda

✓ Motivation

• Solutions
• Advanced solutions
• Evaluation metrics
• Toolboxes
• Summary
Solutions

• Section agenda
  • Solutions *rooted on BR*
  • Solutions *rooted on LP*
  • Other solutions
Solutions rooted on BR

• **BR: Binary Relevance** [Clare and King, 2001; Boutell et al, 2004]
  - Models independent classifiers $P(y_i|x)$ on each class variable
  - Does **not** learn the class dependences

• Key extensions from BR
  - Learn the class dependence relations **by adding new class-dependent features** : $P(y_i|x, \{\text{new \ features}\})$
Solutions rooted on BR

• Idea: layered approach
  
  • Layer-1: Learn and predict on $D_{train}$, using the BR approach
  
  • Layer-2: Learn $d$ classifiers on the original features and the output of layer-1

• Existing methods
  
  • Classification with Heterogeneous Features (CHF) [Godbole et al, 2004]
  
  • Instance-based Logistic Regression (IBLR) [Cheng et al, 2009]
Classification with Heterogeneous Features (CHF)

• Illustration

<table>
<thead>
<tr>
<th>$D_{train}$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=1$</td>
<td>0.7</td>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=2$</td>
<td>0.6</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=3$</td>
<td>0.1</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=4$</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=5$</td>
<td>0.8</td>
<td>0.9</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$X_{CHF}$

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$h_{br1}(X)$</th>
<th>$h_{br2}(X)$</th>
<th>$h_{br3}(X)$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=1$</td>
<td>0.7</td>
<td>0.4</td>
<td>.xx</td>
<td>.xx</td>
<td>.xx</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$n=2$</td>
<td>0.6</td>
<td>0.2</td>
<td>.xx</td>
<td>.xx</td>
<td>.xx</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$n=3$</td>
<td>0.1</td>
<td>0.9</td>
<td>.xx</td>
<td>.xx</td>
<td>.xx</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=4$</td>
<td>0.3</td>
<td>0.1</td>
<td>.xx</td>
<td>.xx</td>
<td>.xx</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=5$</td>
<td>0.8</td>
<td>0.9</td>
<td>.xx</td>
<td>.xx</td>
<td>.xx</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$h_{br1}$ : $X \rightarrow Y_1$

$h_{br2}$ : $X \rightarrow Y_2$

$h_{br3}$ : $X \rightarrow Y_3$
Instance-based Logistic Regression (IBLR)

• Illustration

<table>
<thead>
<tr>
<th>(D_{\text{train}})</th>
<th>(X_1)</th>
<th>(X_2)</th>
<th>(Y_1)</th>
<th>(Y_2)</th>
<th>(Y_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n=1)</td>
<td>0.7</td>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(n=2)</td>
<td>0.6</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(n=3)</td>
<td>0.1</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(n=4)</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(n=5)</td>
<td>0.8</td>
<td>0.9</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\text{KNN Score})</th>
<th>(\lambda_1)</th>
<th>(\lambda_2)</th>
<th>(\lambda_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\cdot \cdot\cdot)</td>
<td>(\cdot \cdot\cdot)</td>
<td>(\cdot \cdot\cdot)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(k=3)</th>
<th>(Y_1)</th>
<th>(Y_2)</th>
<th>(Y_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(1)</td>
<td>(0)</td>
<td></td>
</tr>
<tr>
<td>(0)</td>
<td>(0)</td>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>(1)</td>
<td>(0)</td>
<td>(0)</td>
<td></td>
</tr>
</tbody>
</table>

\(h_1\) : \(X_{\text{IBLR}} \rightarrow Y_1\)

\(h_2\) : \(X_{\text{IBLR}} \rightarrow Y_2\)

\(h_3\) : \(X_{\text{IBLR}} \rightarrow Y_3\)
Solutions rooted on BR: CHF & IBLR

• **Advantages**
  
  • Model the class dependences by enriching the feature space using the *layer-1* classifiers

• **Disadvantages**
  
  • Learn the dependence relations in an *indirect way*
  
  • The predictions are not stable
Solutions rooted on LP

- LP: Label Powerset [Tsoumakas and Vlahavas, 2007]
  - Models a multi-class classifier on the enumeration of all possible class assignment
  - Can create exponentially many classes and computationally very expensive

- Key extensions from LP
  - Prune the infrequent class assignments from the consideration to reduce the size of the class assignment space
  - Represent the joint distribution more compactly
Pruned problem transformation (PPT) [Read et al, 2008]

- Class assignment conversion in PPT
  - Prune infrequent class assignment sets
  - User specifies the threshold for “infrequency”

<table>
<thead>
<tr>
<th>$D_{train}$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=1$</td>
<td>0.7</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=2$</td>
<td>0.6</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=3$</td>
<td>0.1</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=4$</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=5$</td>
<td>0.1</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=6$</td>
<td>0.2</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=7$</td>
<td>0.2</td>
<td>0.2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=8$</td>
<td>0.2</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=9$</td>
<td>0.7</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=10$</td>
<td>0.9</td>
<td>0.9</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_{train-LP}$</th>
<th>$Y_{LP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=1$</td>
<td>1</td>
</tr>
<tr>
<td>$n=2$</td>
<td>1</td>
</tr>
<tr>
<td>$n=3$</td>
<td>0</td>
</tr>
<tr>
<td>$n=4$</td>
<td>2</td>
</tr>
<tr>
<td>$n=5$</td>
<td>0</td>
</tr>
<tr>
<td>$n=6$</td>
<td>2</td>
</tr>
<tr>
<td>$n=7$</td>
<td>2</td>
</tr>
<tr>
<td>$n=8$</td>
<td>0</td>
</tr>
<tr>
<td>$n=9$</td>
<td>1</td>
</tr>
<tr>
<td>$n=10$</td>
<td>3</td>
</tr>
</tbody>
</table>

$|Y_{LP}| = 4$
Pruned problem transformation (PPT) [Read et al, 2008]

- Class assignment conversion in PPT
  - Prune infrequent class assignment sets
  - User specifies the threshold for “infrequency”

<table>
<thead>
<tr>
<th>$D_{train}$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=1$</td>
<td>0.7</td>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=2$</td>
<td>0.6</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=3$</td>
<td>0.1</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=4$</td>
<td>0.3</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=5$</td>
<td>0.1</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=6$</td>
<td>0.2</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=7$</td>
<td>0.2</td>
<td>0.2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$n=8$</td>
<td>0.2</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$n=9$</td>
<td>0.7</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=10$</td>
<td>0.9</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$n=11$</td>
<td>0.9</td>
<td>0.9</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_{train-PPT}$</th>
<th>$Y_{PPT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=1$</td>
<td>1</td>
</tr>
<tr>
<td>$n=2$</td>
<td>1</td>
</tr>
<tr>
<td>$n=3$</td>
<td>0</td>
</tr>
<tr>
<td>$n=4$</td>
<td>2</td>
</tr>
<tr>
<td>$n=5$</td>
<td>0</td>
</tr>
<tr>
<td>$n=6$</td>
<td>2</td>
</tr>
<tr>
<td>$n=7$</td>
<td>2</td>
</tr>
<tr>
<td>$n=8$</td>
<td>0</td>
</tr>
<tr>
<td>$n=9$</td>
<td>1</td>
</tr>
<tr>
<td>$n=10$</td>
<td>1</td>
</tr>
<tr>
<td>$n=11$</td>
<td>2</td>
</tr>
</tbody>
</table>

$|Y_{PPT}| = 3$
Solutions rooted on LP: \textit{PPT}

- Advantages
  - Simple add-on to the LP method that focuses on key relationships
  - Models the full joint more efficiently

- Disadvantages
  - Based on an ad-hoc pruning heuristic
    - Mapping to lower dimensional label space is not clear
  - (As LP) Only predicts the label combinations that are seen in the training set
Other solution: MLKNN [Zhang and Zhou, 2007]

- Multi-label k-Nearest Neighbor (MLKNN) [Zhang and Zhou, 2007]
  - Learn a classifier for each class (as BR) by combining k-nearest neighbor with Bayesian inference
  - Application is limited as KNN
    - Does not produce a model
    - Does not work well on high-dimensional data
Multi-label output coding

• Key idea
  • Motivated by the error-correcting output coding (ECOC) scheme [Dietterich 1995; Bose & Ray-Chaudhuri 1960] in communication
  • Solve the MLC problems using lower dimensional codewords
  • An output coding MLC method usually consists of three parts:
    • Encoding: Convert output vectors $Y$ into codewords $Z$
    • Prediction: Perform regression from $X$ to $Z$; say $R$
    • Decoding: Recover the class assignments $Y$ from $R$
Multi-label output coding

- Existing methods
  - OC (Output Coding) with Compressed Sensing (OCCS) [Hsu et al, 2009]
  - Principle Label Space Transformation (PLST) [Tai and Lin, 2010]
  - OC with Canonical Correlation Analysis (CCAOC) [Zhang and Schneider, 2011]
  - Maximum Margin Output Coding (MMOC) [Zhang and Schneider, 2012]
**Principle Label Space Transformation (PLST)** [Tai and Lin, 2010]

- **Encoding**: Convert output vectors $Y$ into codewords $Z$, using the singular vector decomposition (SVD)
  - $Z = V^T Y = (V_1^T Y, \ldots, V_q^T Y)$, where $V$ is a $d \times q$ projection vector ($d > q$)

- **Prediction**: Perform regression from $X$ to $Z$; say $R$

- **Decoding**: Recover the class labels $Y$ from $R$ using SVD
  - Achieved by optimizing a combinatorial loss function
Multi-label output coding

- Existing methods are differentiated from one to another mainly by the encoding/decoding schemes they apply.

<table>
<thead>
<tr>
<th>Method</th>
<th>Key difference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OCCS</strong></td>
<td>Uses compressed sensing [Donoho 2006] for encoding and decoding</td>
</tr>
<tr>
<td><strong>PLST</strong></td>
<td>Uses singular vector decomposition (SVD) [Johnson &amp; Wichern 2002] for encoding and decoding</td>
</tr>
<tr>
<td><strong>CCAOC</strong></td>
<td>Uses canonical correlation analysis (CCA) [Johnson &amp; Wichern 2002] for encoding and mean-field approximation for decoding</td>
</tr>
<tr>
<td><strong>MMOC</strong></td>
<td>Uses SVD for encoding and maximum margin formulation for decoding</td>
</tr>
</tbody>
</table>
Multi-label output coding

• Advantages
  • Show excellent prediction performances

• Disadvantages
  • Only able to predict the single best output for a given input
    • Cannot estimate probabilities for different input-output pairs
  • Not scalable
    • Encoding and decoding steps rely on matrix decomposition, whose complexities are sensitive to $d$ and $N$
  • Cannot be generalized to non-binary cases
Section summary

Independent classifiers: BR, CHF, IBLR
Enriched feature space

Want to achieve something better

Pruned label combo: PPT
All possible label combo: LP

Others: Output coding (MLKNN)

BR
CHF
IBLR

?
Agenda

✓ Motivation
✓ Solutions

• Advanced solutions
• Evaluation metrics
• Toolboxes
• Summary
Advanced solutions

• Section agenda
  • Extensions using probabilistic graphical models (PGMs)
  • Extensions using ensemble techniques
Extensions using PGMs

- Probabilistic Graphical Models (PGMs)
  - PGM refers to a family of distributions on a set of random variables that are compatible with all the probabilistic independence propositions encoded in a graph
  - A smart way to formulate exponentially large probability distribution without paying an exponential cost
  - Using PGMs, we can reduce the model complexity
  - PGM = Multivariate statistics + Graphical structure
Extensions using PGMs

- **Representation: Two types**
  - Undirected graphical models (UGMs)
    - Also known as Markov networks (MNs)
  - Directed graphical models (DGMs)
    - Also known as Bayesian networks (BNs)

\[ X_1 \quad \text{node: variable} \]

\[ X_2 \quad \text{edge: correlation} \]

Undirected (MN)

\[ X_1 \quad \text{edge: causal relation} \]

Directed (BN)
Extensions using **PGMs**

- How PGMs reduce the model complexity?
  - Key idea: Exploit the **conditional independence (CI)** relations among variables!!

  - Conditional independence (CI): Random variables $A$, $B$ are conditionally independent given $C$, if $P(A,B|C) = P(A|C)P(B|C)$

- **UGM** and **DGM** offer a set of graphical notations for CI

**CI representation in UGM**

\[
A \perp B \mid C
\]
Extensions using PGMs

- How PGMs reduce the model complexity?
  - Key idea: Exploit the conditional independence (CI) relations among variables!!
  - Conditional independence (CI): Random variables $A$, $B$ are conditionally independent given $C$, if $P(A,B|C) = P(A|C)P(B|C)$

- UGM and DGM offer a set of graphical notations for CI

\[
\begin{array}{c|c|c}
| & A \perp B | C & A \perp B | C \\
\hline
\text{CI representations in DGM} & A \perp B | C & A \perp B | C \\
\end{array}
\]
Extensions using PGMs

- PGMs have been an excellent representation / formulation tool for the MLC problems
- The dependences among features ($X$) and class variables ($Y$) can be represented easily with PGMs
- By exploiting the conditional independence, we can make the computation simpler
Extensions using PGMs

• Existing methods

• Undirected models (Markov networks)
  • Multi-label Conditional Random Field (ML-CRF) [Ghamrawi and McCallum, 2005; Pakdaman et al, 2014]
  • Composite Marginal Models (CMM) [Zhang and Schneider, 2012]

• Directed models (Bayesian networks)
  • Multi-dimensional Bayesian Classifiers (MBC) [van der Gaag and de Waal, 2006]
  • Classifier Chains (CC) [Read et al, 2009]
  • Conditional Tree-structured Bayesian Networks (CTBN) [Batal et al, 2013]
Multi-dimensional Bayesian Networks (MBC) [van der Gaag and de Waal, 2006]

• Key idea
  • Model the full joint of input and output using a Bayesian network
  • Use graphical structures to represent the dependence relations among the input and output variables

• Example MBC \((d = 3, m = 4)\)

The joint distribution \(P(X, Y)\) is represented by the decomposition

\[
X = X_1|X_2 \cdot X_2|X_3 \cdot X_3 \cdot X_4|X_2
\]

and

\[
Y = Y_1|Y_2 \cdot X_2 \cdot Y_3|Y_2
\]
Multi-dimensional Bayesian Networks (MBC) [van der Gaag and de Waal, 2006]

- **Advantages**
  - The full joint distribution of the feature and class variables can be represented efficiently using the Bayesian network

- **Disadvantages**
  - Models the relations among the feature variables which do not carry much information in modeling the multi-label relations
Multi-label Conditional Random Fields (MLCRF) [Pakdaman et al, 2014]

• Key idea

• Model the conditionals $P(Y|X)$ to capture the relations among the class variables conditioned on the feature variables

• Learn a pairwise Markov network to model the relations between the input and output variables

• Representation

$$P(Y|X) = \frac{\prod_{i=1}^{d} \prod_{j=1}^{d} \psi_{i,j}(Y_i, Y_j, X) \phi_i(Y_i, X)}{Z}$$

($\psi_{i,j}$ and $\phi_i$ are the potentials of $Y_i$, $Y_j$, $X$; and $Z$ is the normalization term)
Multi-label Conditional Random Fields (MLCRF) [Pakdaman et al, 2014]

• Advantages
  • Directly models the conditional joint distribution $P(Y|X)$

• Disadvantages
  • Learning and prediction is computationally very demanding
    • To perform an inference, the normalization term $Z$ should be computed, which is usually very costly
    • The iterative parameter learning process requires inference at each step whose computational cost is even more expensive
  • In practice, approximate inference techniques are applied to make the model usable
Classifier Chains (CC) [Read et al, 2009]

- **Key idea**
  - Model $P(Y|X)$ using a directed chain network, where all preceding classes in the chain are conditioning the following class variables.

- **Representation**

\[
P(Y|X) = \prod_{i=1}^{d} P(Y_i | X, \pi(Y_i))
\]

\[
= \prod_{i=1}^{d} P(Y_i | X, Y_1, ..., Y_{i-1})
\]

\[
= P(Y_1 | X) \cdot P(Y_2 | X, Y_1) \cdot ... \cdot P(Y_d | X, Y_1, ..., Y_{d-1})
\]
Classifier Chains (CC) \[\text{[Read et al, 2009]}\]

- **Learning**

  \[
P(Y|X) = \prod_{i=1}^{d} P(Y_i|X, \pi(Y_i))
  \]

  ![Classifier Chains Diagram](image)

  - No structure learning (random chain order)
  - Parameter learning is performed on the decomposed CPDs:
    \[
    \text{argmax}_{\theta} P(Y_i|X, \pi(Y_i); \theta)
    \]

- **Prediction**

  - Performed by greedy maximization of each factors (CPDs):
    \[
    \text{argmax}_{Y_i} P(Y_i|X, \pi(Y_i); \theta)
    \]
Conditional Tree-structured Bayesian Networks (CTBNs) [Batal et al, 2013]

• Key idea
  • Learn $P(Y|X)$ using a tree-structured Bayesian network of the class labels
  • Tree-structures can be seen as restricted chains, where each class variable has at most one parent class variable

• Example CTBN

This network represents:

$$P(y_1, y_2, y_3, y_4|x) = P(y_3|x) \cdot P(y_2|x, y_3) \cdot P(y_1|x, y_2) \cdot P(y_4|x, y_2)$$
Conditional Tree-structured Bayesian Networks (CTBNs) [Batal et al, 2013]

- Learning

\[
P(Y|X) = \prod_{i=1}^{d} P(Y_i|X, \pi(Y_i))
\]

- Structure learning by optimizing conditional log-likelihood

1. Define a complete weighted directed graph, whose edge weights is equal to conditional log-likelihood

2. Find the maximum branching tree from the graph
   (* Maximum branching tree = maximum weighted directed spanning tree)
Conditional Tree-structured Bayesian Networks (CTBNs) [Batal et al, 2013]

• Learning

\[ P(Y|X) = \prod_{i=1}^{d} P(Y_i|X, \pi(Y_i)) \]

• Structure learning by optimizing conditional log-likelihood

• Parameter learning is performed on the decomposed CPDs

• Prediction

• Exact MAP prediction is performed by a belief propagation (max-product) algorithm
## CC vs. CTBN

<table>
<thead>
<tr>
<th><strong>CC</strong></th>
<th><strong>CTBN</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>![Diagram of CC]</td>
<td>![Diagram of CTBN]</td>
</tr>
</tbody>
</table>

\[
P(Y|X) = \prod_{i=1}^{d} P(Y_i|X, \pi(Y_i))
\]

- **Decomposes the joint probability along with the chain structure**
- **No structure learning (label ordering is given at random)**
- **Maximizes the marginals along the chain (suboptimal solution)**
- **Errors in prediction propagate to the following label prediction**

\[
P(Y|X) = \prod_{i=1}^{d} P(Y_i|X, \pi(Y_i))
\]

- **Decomposes the joint probability along with the tree structure**
- **Tree structure is learned using a score-based algorithm**
- **Performs exact MAP prediction (linear time optimal solution)**
- **The tree-structure assumption may restrict its modeling ability**
Advanced solutions

• Section agenda
  ✓ Extensions using probabilistic graphical models (PGMs)

• Extensions using ensemble techniques
Extensions using ensemble techniques

• Ensemble techniques
  • Techniques of training multiple classifiers and combining their predictions to produce a single classifier

• Ensemble techniques can further improve the performance of MLC classifiers
  • Objective: Use a combination of simpler classifiers to improve predictions
Extensions using ensemble techniques

- Existing methods
  - Ensemble of CCs (ECC) [Read et al, 2009]
  - Mixture of CTBNs (MC) [Hong et al, 2014]
Ensemble of Classifier Chains (ECC) [Read et al, 2009]

• Recall CC

\[ P(Y|X) = \prod_{i=1}^{d} P(Y_i|X, \pi(Y_i)) \]

• Key Idea
  
  • Create user-specified number of CC’s on random subsets of data with random orderings of the class labels
  
  • Predict by majority vote over all base classifiers
Ensemble of Classifier Chains (ECC) [Read et al, 2009]

• Advantages
  • Often times, the performance improves

• Disadvantages
  • Ad-hoc ensemble implementation
    • Learns base classifiers on random subsets of data with random label ordering
    • Ensemble decisions are made by simple averaging over the base models and often inaccurate
Mixture of CTBNs (MC) [Hong et al, 2014]

- Motivation
  - If the underlying dependency structure in data is more complex than a tree structure, a single CTBN cannot model the data properly

- Key idea
  - Use the *Mixtures-of-Trees* [Meila and Jordan, 2000] framework to learn multiple CTBNs and use them for prediction
Mixture of CTBNs (MC) [Hong et al., 2014]

- MC defines the multivariate posterior distribution of class vector $P(y|x) = P(y_1, \ldots, y_d|x)$ as

$$P(y|x) = \sum_{k=1}^{K} \lambda_k P(y|x, T_k)$$

$$= \sum_{k=1}^{K} \lambda_k \prod_{i=1}^{d} P(y_i|x, y_{\pi(i,T)})$$

- $P(y|x, T_k)$ is the $k$-th mixture component defined by a CTBN $T_k$
- $\lambda_k$ is the mixture coefficient representing the weight of the $k$-th component (influence of the $k$-th CTBN model $T_k$ to the mixture)
Mixture of CTBNs (MC) [Hong et al, 2014]

- An example MC

\[ P(y|x) = \sum_{k=1}^{K} \lambda_k P(y|x, T_k) \]
Mixture of CTBNs (MC) [Hong et al, 2014]

- Parameter learning

- Objective: Optimize the model parameters (CTBN parameters $\{\theta_1, \ldots, \theta_K\}$ and mixture coefficients $\{\lambda_1, \ldots, \lambda_K\}$)

- Idea (apply EM)

1. Associate each instance $(x^{(n)}, y^{(n)})$ with a hidden variable $z^{(n)} \in \{1, \ldots, K\}$ indicating which CTBN it belongs to.

2. Iteratively optimize the expected complete log-likelihood:

$$E \left[ \sum_{n=1}^{N} \log P(y^{(n)}, z^{(n)}|x^{(n)}) \right]$$

$$= E \left[ \sum_{n=1}^{N} \sum_{k=1}^{K} 1[z^{(n)} = k] \left[ \log \lambda_k + \log P(y^{(n)}|x^{(n)}, T_k) \right] \right]$$
Mixture of CTBNs (MC) [Hong et al, 2014]

- **Structure learning**
  - **Objective:** Find *multiple CTBN structures* from data
  - **Idea** (*boosting-like heuristic*)
    1. On each addition of a new structure to the mixture, recalculate the weight of each data instance \((\omega)\) such that it represents the relative “hardness” of the instance
    2. Learn the best tree structure by optimizing the weighted conditional log-likelihood:

\[
\sum_{n=1}^{N} \sum_{i=1}^{d} \omega^{(n)} \log P(y_{i}^{(n)}|x^{(n)}, y_{\pi(i,T)}^{(n)})
\]
Mixture of CTBNs (MC) [Hong et al, 2014]

• Prediction
  • Objective: Find the maximum a posteriori (MAP) prediction for a new instance \( x \)

• Idea
  1. Search the space of all class assignments by defining a Markov chain
  2. Use an annealed version of exploration procedure to speed up the search
Mixture of CTBNs (MC)  [Hong et al, 2014]

- **Advantages**
  - Learns an ensemble model for MLC in a principled way
  - Produces accurate and reliable results

- **Disadvantages**
  - Iterative optimization process in learning requires a large amount of time
Agenda

✓ Motivation
✓ Solutions
✓ Advanced solutions

• Evaluation metrics
• Toolboxes
• Summary
Evaluation metrics

- Evaluation of MLC methods is more difficult than that of single-label classification

- Measuring the Hamming accuracy is not sufficient for the goal of MLC

  - Hamming accuracy (HA) = \( \frac{1}{N} \sum_{n=1}^{N} \left[ h(x^{(n)}) \Delta y^{(n)} \right] \)

  - HA measures the individual accuracy on each class variable, which can be optimized by the binary relevance (BR) model

- We want to find “jointly accurate” class assignments

- We want to measure if the model predicts all the labels correctly

  - Exact match accuracy (EMA) = \( \frac{1}{N} \sum_{n=1}^{N} \left[ h(x^{(n)}) = y^{(n)} \right] \)
Evaluation metrics

• **Exact match accuracy (EMA)**
  \[
  EMA = \frac{1}{N} \sum_{n=1}^{N} \left[ h(x^{(n)}) = y^{(n)} \right]
  \]
  - EMA evaluate if the prediction is correct on all class variables
  - Most appropriate metric for MLC
    - We are looking for the most probable assignment of classes
    - It can be too strict

• **Multi-label accuracy (MLA)**
  \[
  MLA = \frac{1}{N} \sum_{n=1}^{N} \frac{h(x^{(n)}) \cap y^{(n)}}{h(x^{(n)}) \cup y^{(n)}}
  \]
  - MLA evaluate the Jaccard index between prediction and true class assignments
  - It is less strict than EMA; overestimates the model accuracy
Evaluation metrics

- **Conditional log-likelihood loss (CLL-loss)**
  \[
  \text{CLL-loss} = \sum_{n=1}^{N} - \log P(y^{(n)}|x^{(n)}; M)
  \]
  
  - Reflects the model fitness

- **F1-scores: harmonics mean of precision and recall**
  - **Micro F1**
    \[
    \text{Micro F1} = \frac{1}{N} \sum_{n=1}^{N} \frac{2 \times TP^{(n)}}{2 \times TP^{(n)} + FP^{(n)} + FN^{(n)}}
    \]
    
    - Computes the F1-score on each instance and then average out
  - **Macro F1**
    \[
    \text{Macro F1} = \frac{1}{d} \sum_{i=1}^{d} \frac{2 \times TP^{(i)}}{2 \times TP^{(i)} + FP^{(i)} + FN^{(i)}}
    \]
    
    - Computes the F1-score on each class and then average out
Agenda

✓ Motivation
✓ Solutions
✓ Advanced solutions
✓ Evaluation metrics

• Toolboxes
• Summary
Toolboxes

- **MEKA**: a Multi-label Extension to WEKA
  http://meka.sourceforge.net/

- **Mulan**: a Java library for Multi-label Learning
  http://mulan.sourceforge.net/

- **LibSVM MLC Extension (BR and LP)**
  http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/multilabel/

- **LAMDA Lab (Nanjing Univ., China) Code Repository**

- **Prof. Min-Ling Zhang** (Southeast Univ., China)
  http://cse.seu.edu.cn/old/people/zhangml/Resources.htm#codes
Summary

Independent classifiers

Enriched feature space

Want to achieve something better

Pruned label combo.

All possible label combo.

Others

Output coding

MLKNN
References


References


References


Thanks!