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

Contents

Introduction: Multi-label Classification

Binary classification: Is this a picture of a beach? $\in \{\texttt{yes}, \texttt{no}\}$
Multi-class classification: Which class does this picture belong to?

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

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

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

i.e., each instance can have multiple labels instead of a single one!

Introduction: Single-label vs. Multi-label

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

X_1	X_2	X_3	X_4	X_5	beach Y
1	0.1	3	1	0	0
0	0.9	1	0	1	1
0	0.0	1	1	0	0
1	0.8	2	0	1	1
1	0.0	2	0	1	0
0	0.0	3	1	1	?

Build classifier h, such that $\hat{y} = h(\tilde{\mathbf{x}})$.

Introduction: Single-label vs. Multi-label

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

X_1	X_2	X_3	X_4	X_5	beach Y_1	sunset Y_2	foliage Y_3	mountain Y_4	urban Y_5	field Y_6
1	0.1	3	1	0	0	1	1	0	1	0
0	0.9	1	0	1	1	0	0	0	0	0
0	0.0	1	1	0	0	1	0	0	0	0
1	0.8	2	0	1	1	0	0	1	0	1
1	0.0	2	0	1	0	0	0	1	0	1
0	0.0	3	1	1	?	?	?	?	?	?

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

Introduction: Another Example

Table : The IMDB Dataset

example	*abandoned* X_1	*accident* X_2	... ...	*violent* X_{1000}	*wedding* X_{1001}	horror Y_1	romance Y_2	... ...	comedy Y_{27}	action Y_{28}
1	1	0	...	0	1	0	1	...	0	0
2	0	1	...	1	0	1	0	...	0	0
3	0	0	...	0	1	0	1	...	0	0
4	1	1	...	0	1	1	0	...	0	1
5	1	1	...	0	1	0	1	...	0	1
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋱	⋮	⋮
120919	1	1	...	0	0	0	0	...	0	1

Applications / Datasets

	$\mathcal{X}$ (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

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

- 0/1 LOSS: label vectors must match exactly

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

- HAMMING LOSS: predicting all 0s will incur relatively little loss

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

- 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 $Y_j \in \mathbb{R}$

					sex	cat.	type	rate
X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
x_1	x_2	x_3	x_4	x_5	F	4	A	0.3
x_1	x_2	x_3	x_4	x_5	M	2	B	0.2
x_1	x_2	x_3	x_4	x_5	M	2	A	0.4
x_1	x_2	x_3	x_4	x_5	F	3	C	0.8

Related Applications

Structured Output Prediction

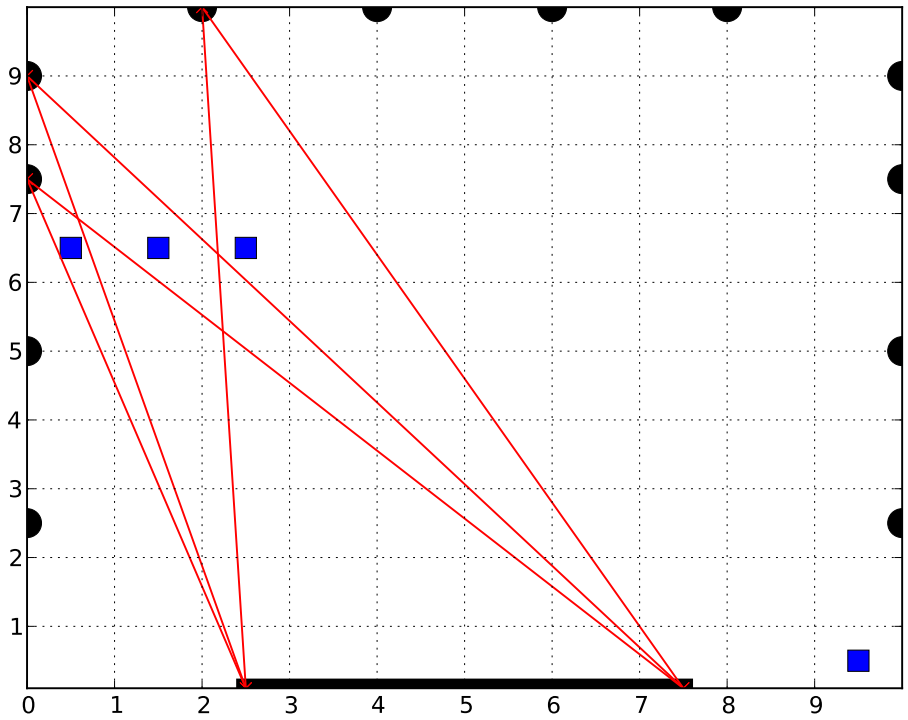


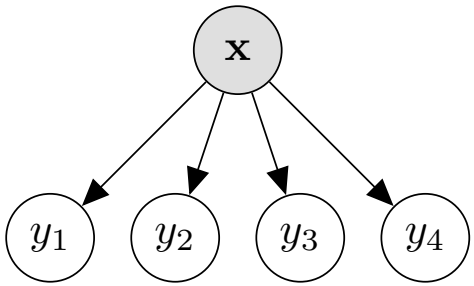
Figure : Structured learning: A multi-label problem with large L; underlying *structure*. Here we want to segment the *relevant* 'pixels' $\mathbf{y} \in 2^L$ occupied by object(s), given some sensor observations $\mathbf{x} = [x_1, \ldots, x_d]$.

Binary Relevance (BR)

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 $\tilde{\mathbf{x}}$, predict *independently*:

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



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

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

Binary Relevance (BR)

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

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

Example

In the IMDB dataset,

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

whereas

$$P(y_{\text{adult}} = 1)P(y_{\text{family}} = 1) > 0$$

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.

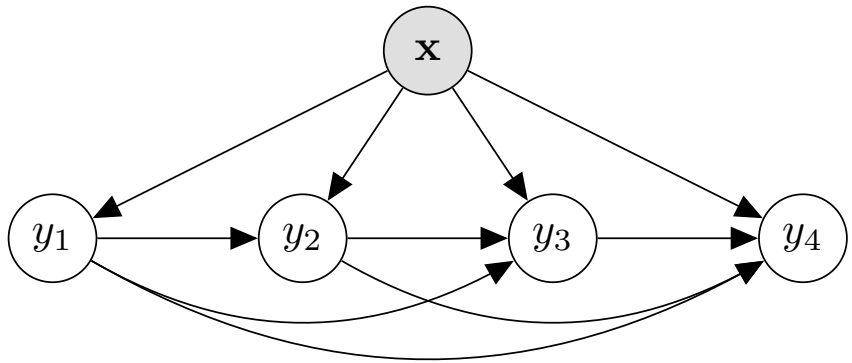
Classifier Chains[2] (CC)

Inspiration from the chain rule

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

- Build $\mathbf{h} = (h_1, \ldots, h_L)$; each $h_j : \mathcal{X} \times \{0,1\}^{j-1} \rightarrow \{0,1\}$
- For any $\tilde{\mathbf{x}}$, predict

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

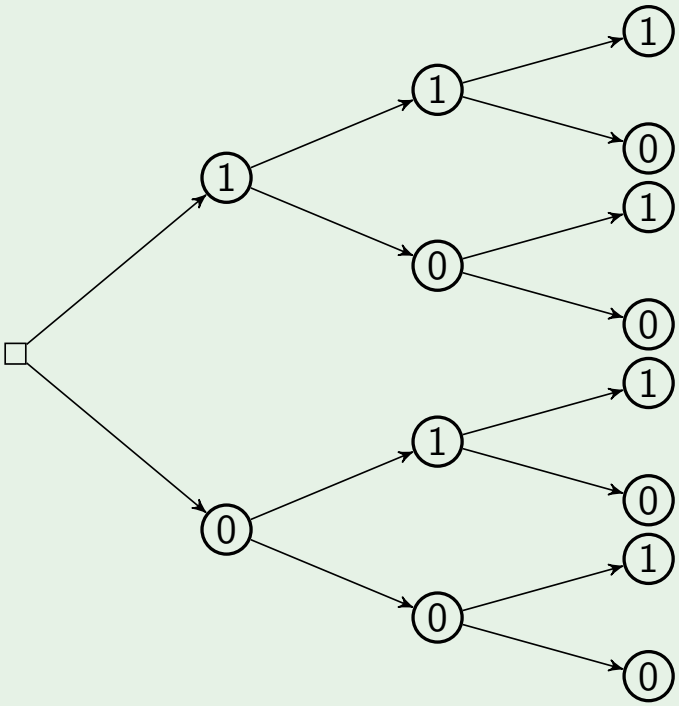


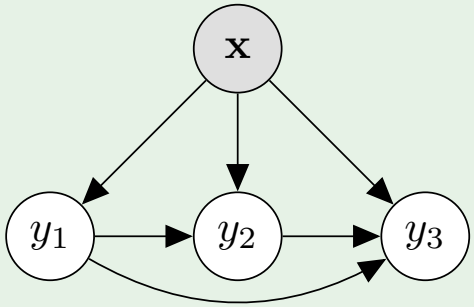
- CC is a greedy approximation; similar complexity to BR.

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

[2] [Read et al., 2009], MLJ

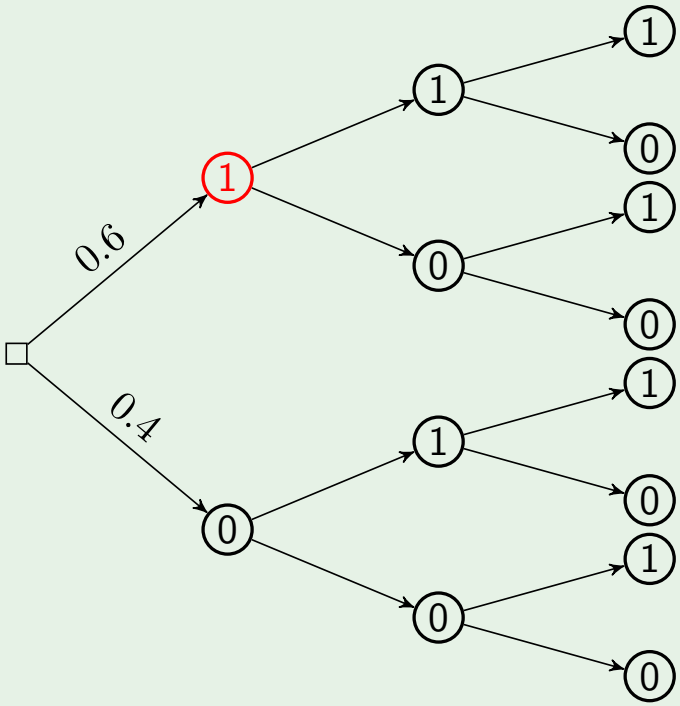
Example

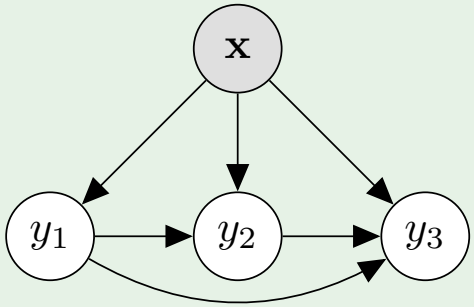




$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [?, ?, ?]$

Example

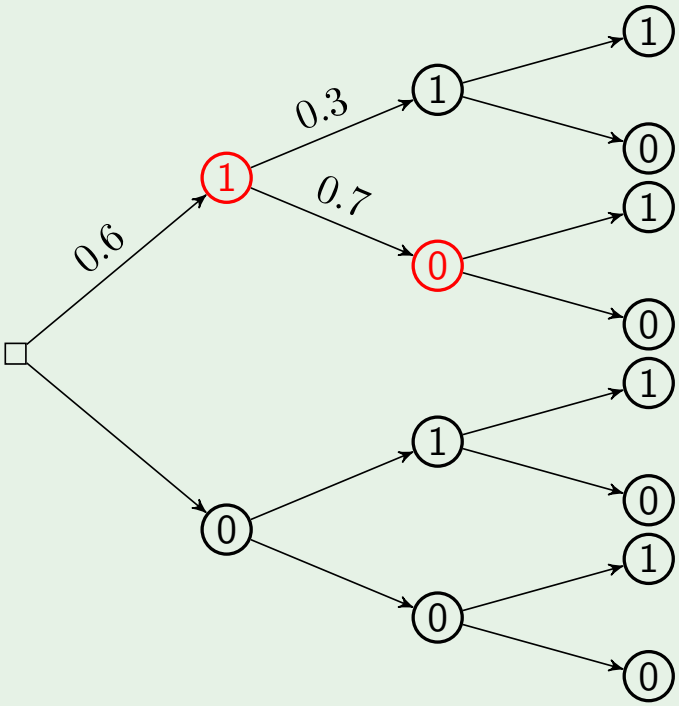


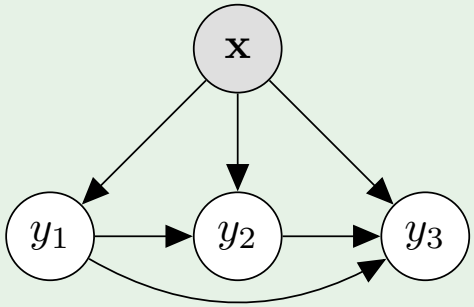


1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = \text{argmax}_{y_1}\, p(y_1|\tilde{\mathbf{x}}) = 1$

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

Example

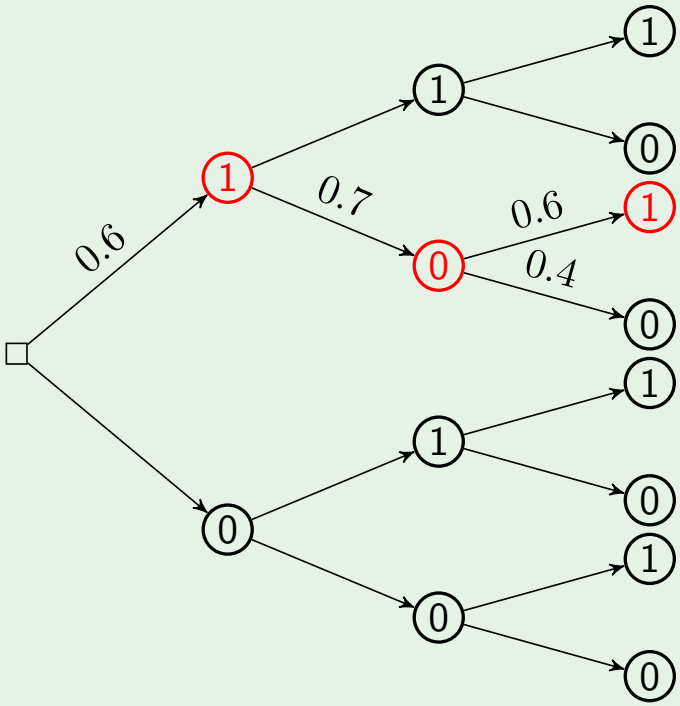


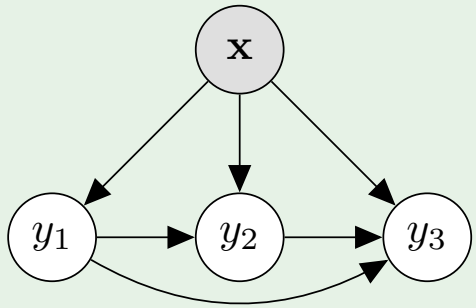


1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = \text{argmax}_{y_1}\ p(y_1|\tilde{\mathbf{x}}) = 1$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1) = \ldots = 0$

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

Example

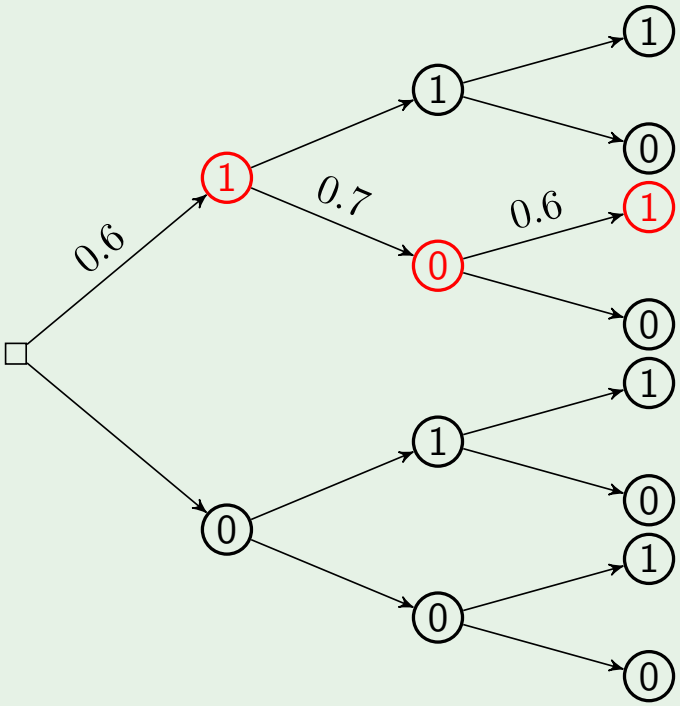


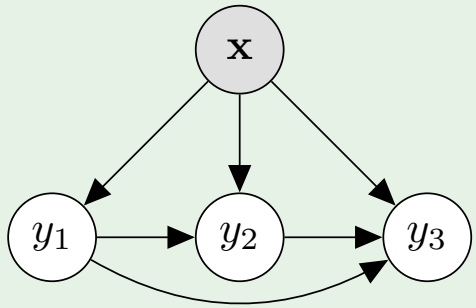


1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = \text{argmax}_{y_1}\ p(y_1|\tilde{\mathbf{x}}) = 1$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1) = \ldots = 0$
3. $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2) = \ldots = 1$

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

Example





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

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

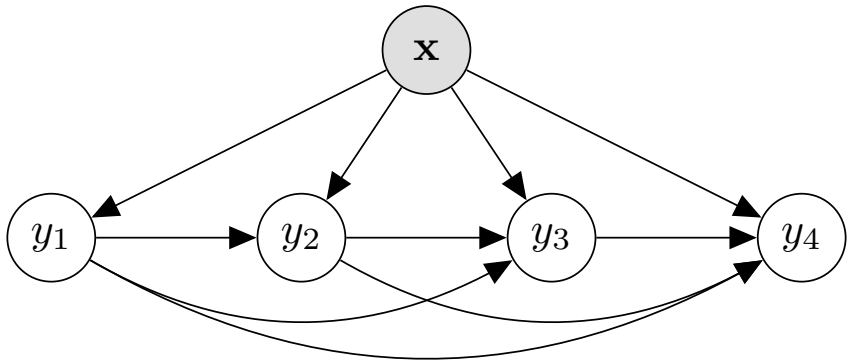
- Improves over BR; similar build time (if $L < D$); parallelizable; able to use any off-the-shelf classifier as h_j
- But, errors may be propagated down the chain

Bayes Optimal Probabilistic Classifier Chains[3] (PCC)

Bayes-optimal Probabilistic CC, recovers the chain rule, predicts

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in\{0,1\}^L}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x})$$

$$= \underset{\mathbf{y}\in\{0,1\}^L}{\operatorname{argmax}} \left\{ p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1}) \right\}$$

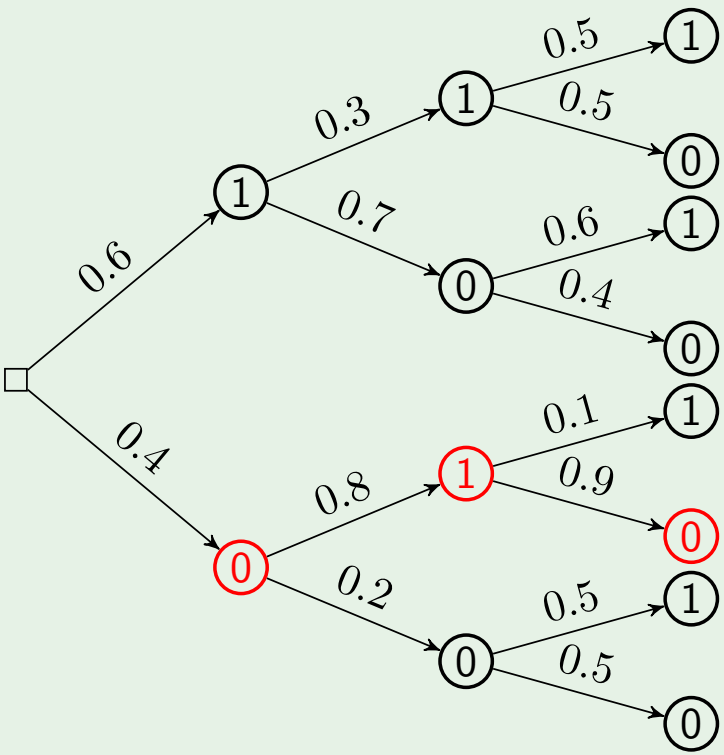


Test all possible paths ($\mathbf{y} = [y_1, \ldots, y_L] \in 2^L$ in total)

[3] [Dembczyński et al., 2010], ICML'10

Bayes Optimal Probabilistic Classifier Chains[3] (PCC)

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. ...
6. $p(\mathbf{y} = [1,0,1]) = 0.252$
7. ...
8. $p(\mathbf{y} = [1,1,1]) = 0.090$

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

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

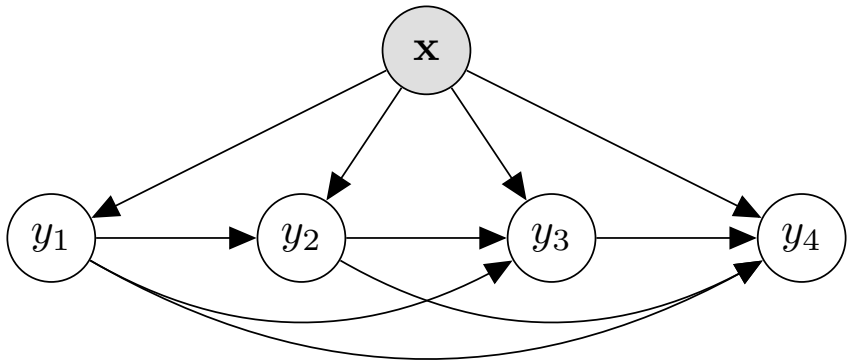
[3] [Dembczyński et al., 2010], ICML'10

Monte-Carlo search for Classifier Chains[4] (MCC)

MCC: Sample the 'chain'.

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

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

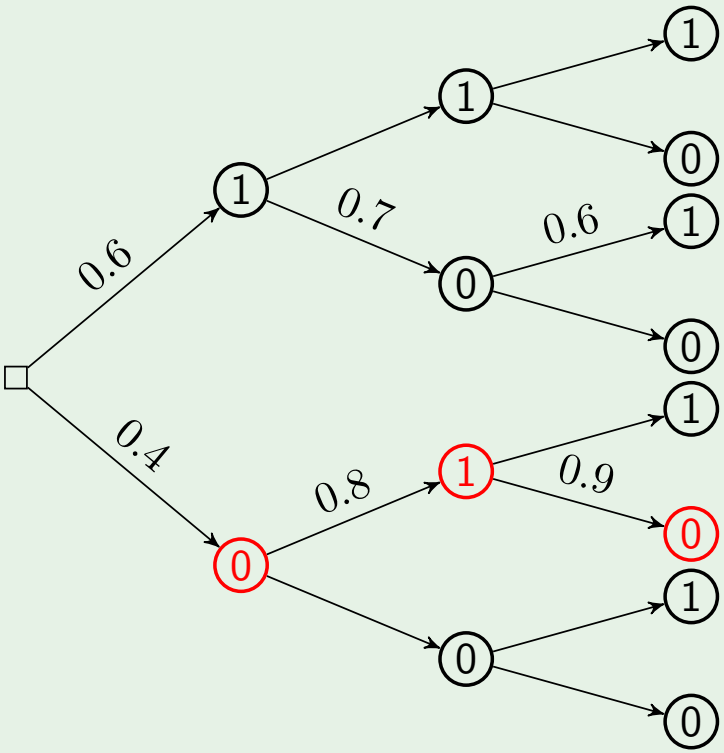


[4] [Read et al., 2013b], Pattern Recognition

Monte-Carlo search for Classifier Chains[4] (MCC)

MCC: Sample the 'chain'.

Example



Sample T times . . .

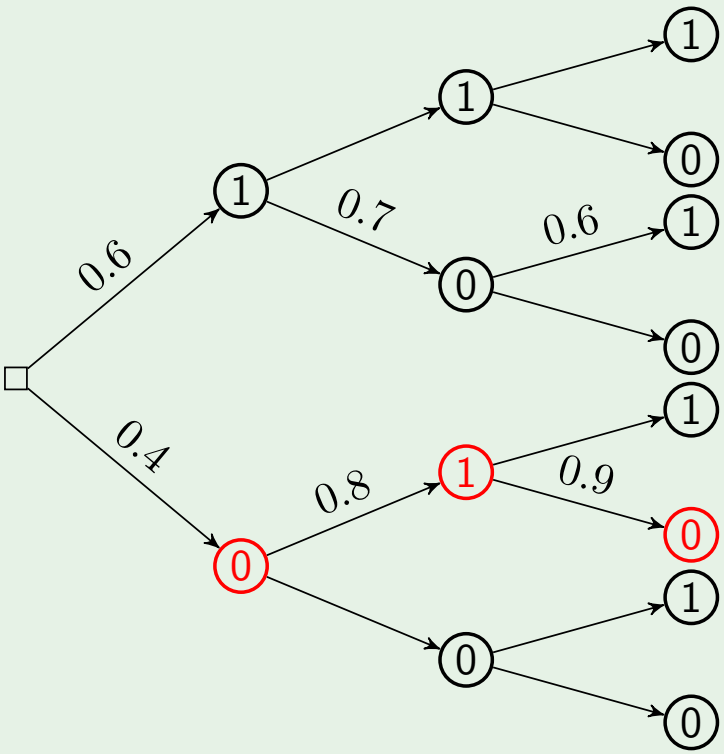
- $p([1,0,1]) = 0.6 \cdot 0.7 \cdot 0.6 = 0.252$
- $p([0,1,0]) = 0.4 \cdot 0.8 \cdot 0.9 = 0.288$

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

[4] [Read et al., 2013b], Pattern Recognition

Monte-Carlo search for Classifier Chains[4] (MCC)

Example



Sample T times ...

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

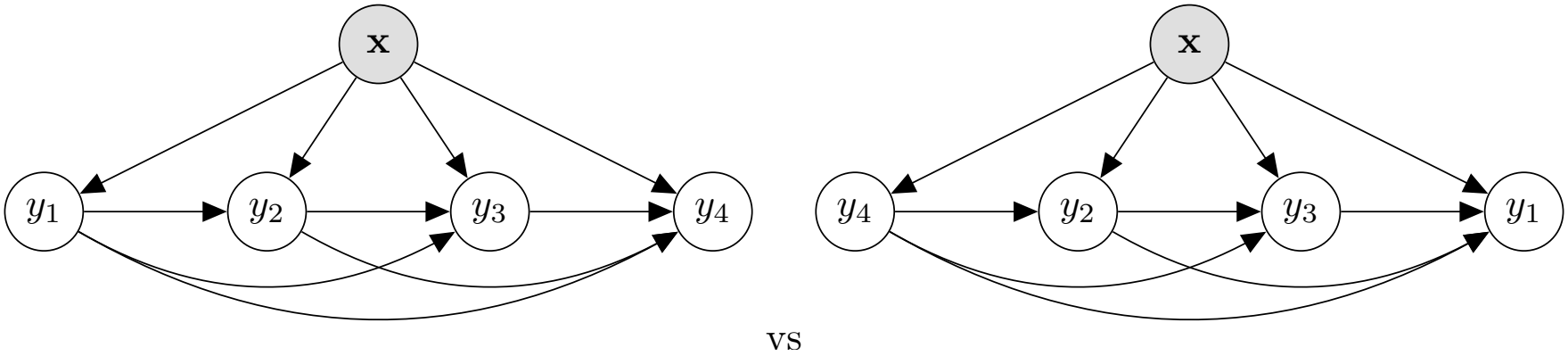
return $\text{argmax}_{\mathbf{y}_t}\, p(\mathbf{y}_t|\mathbf{x})$

- 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?

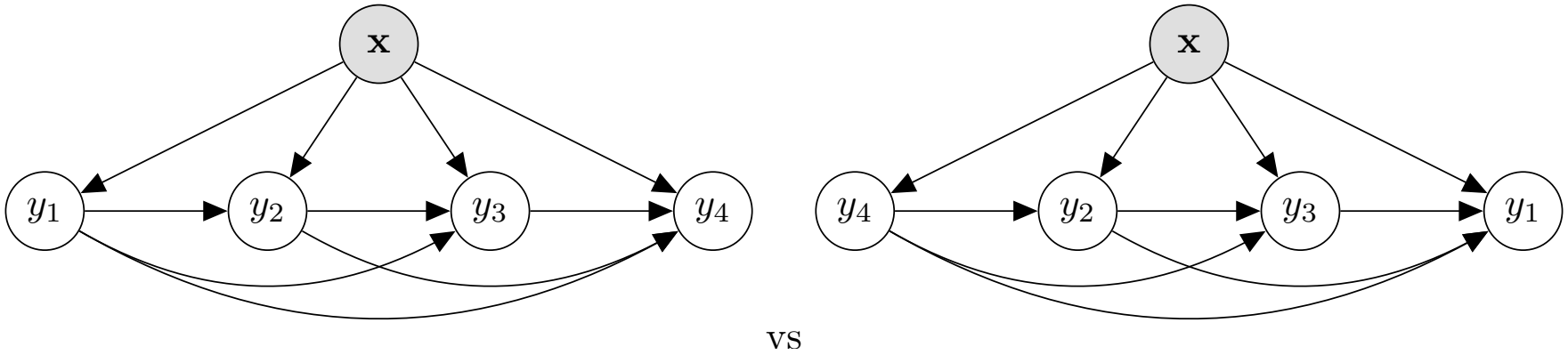
Are these models equivalent?



[5] At least, not necessarily [Kumar et al., 2013, Read et al., 2013a]

Is the Sequence of Labels in the Chain Important?

Are these models equivalent?



No[5]. Although

$$p(y_2|\mathbf{x})p(y_1|y_2,\mathbf{x}) = p(y_1|\mathbf{x})p(y_2|y_1,\mathbf{x})$$

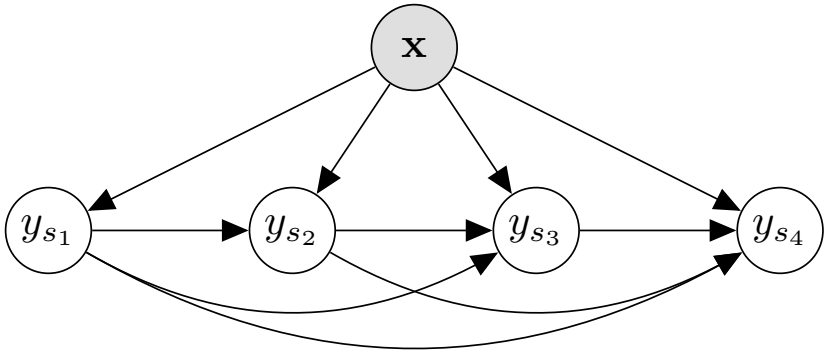
p is only an *estimated* distribution; from finite and noisy data. And

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

[5] At least, not necessarily [Kumar et al., 2013, Read et al., 2013a]

MCC with s-Search[6] ($\mathtt{M_sCC}$)

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



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

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

Use $\mathbf{h}_{\mathbf{s}_U}$ as the final model.

Example

Scene data

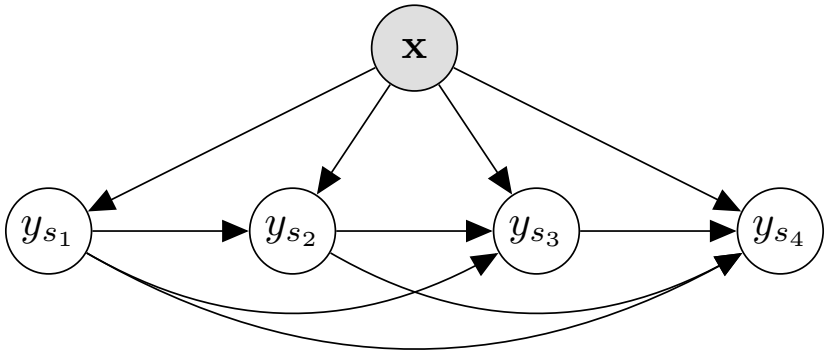
u	$\mathbf{s}_u = [s_1, \ldots, s_L]$	$\mathcal{J}(\mathbf{s}_u)$
0	[4, 2, 0, 1, 3, 5]	0.623
1	[4, 2, 0, 3, 1, 5]	0.628
2	[4, 2, 0, 3, 5, 1]	0.638
3	[4, 0, 2, 3, 5, 1]	0.647
5	[4, 0, 5, 2, 3, 1]	0.653
18	[5, 1, 4, 3, 2, 0]	0.654
23	[5, 4, 0, 1, 2, 3]	0.664
128	[3, 5, 1, 0, 2, 4]	0.668
176	[5, 3, 1, 0, 4, 2]	0.669
225	[5, 3, 1, 4, 0, 2]	0.670

$\mathcal{J}(\mathbf{s}) := \textsc{ExactMatch}(\mathbf{Y}, \mathbf{h}_\mathbf{s}(\mathbf{X}))$ (higher is better)

[6] [Read et al., 2013a], Pattern Recognition

MCC with s-Search[7] ($\mathtt{M_sCC}$)

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



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

[6] *Beam search algorithms for multi-label learning* [Kumar et al., 2013], MLJ

[7] [Read et al., 2013a], Pattern Recognition

An Empirical Look

Table : Average predictive performance (5 fold CV, Exact Match)

	L	BR	CC	PCC	MCC	M_sCC
params:					$T = 100$	$U = 50$
Music	6	0.30	0.29	0.35	0.35	**0.37**
Scene	6	0.54	0.55	0.64	0.64	**0.68**
Yeast	14	0.14	0.15		0.21	**0.23**
Genbase	27	0.94	**0.96**		**0.96**	**0.96**
Medical	45	0.58	0.62		0.63	0.62
Enron	53	0.07	0.10		0.10	0.09
Reuters	101	0.29	0.35		**0.37**	**0.37**

- MCC $=$ PCC, but tractable to larger datasets.
- M_sCC $\succ$ MCC: the chain order makes a difference

An Empirical Look

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

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

- MCC = PCC, but tractable to larger datasets.
- M_sCC $\succ$ MCC: the chain order makes a difference
- although a little **slower** ...

Why not ...

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, $\mathtt{M_sCC}$)

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, $\mathtt{M_s CC}$[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]

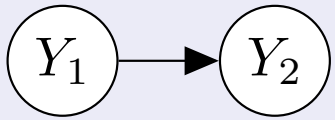
[10] RAndom *k*-labEL subsets for Multi-label Classification [Tsoumakas and Vlahavas, 2007]

Marginal vs. Conditional Dependence

Marginal dependence

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

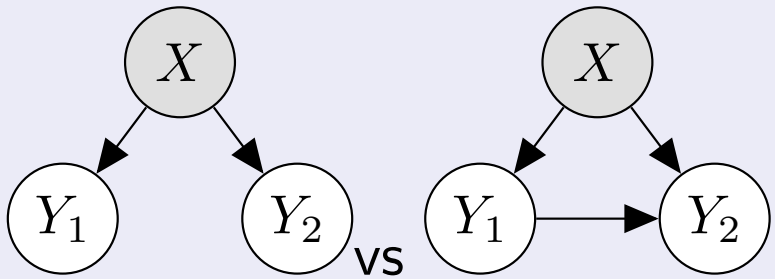
$$p(y_2) \neq p(y_2|y_1)$$



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

Conditional in/dependence

$$p(y_2|y_1, \mathbf{x}) \neq p(y_2|\mathbf{x})$$



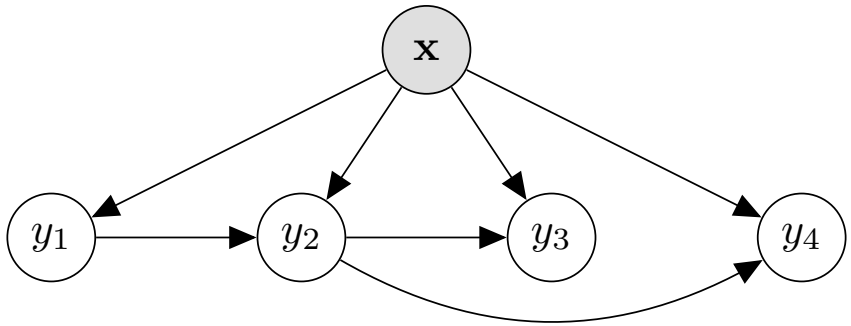
- Have to build and measure models / take into account the input space

From a Chain to a Tree

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

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

where $\mathbf{pa}_j$ = parents of node j.



- If $\mathbf{pa}_j := \{y_1, \ldots, y_{j-1}\}$ we recover CC

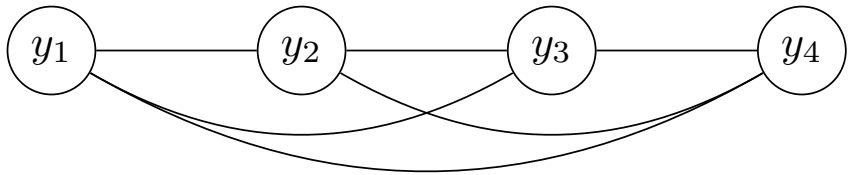
How do we find a good structure?

- label dependence!
- difficult to find, but can benefit accuracy, train/test time.

Bayesian Chain Classifiers[11] (BCC)

Employ CC in a 'tree' (a 'Classifier Tree'):

1. Weight all edges with (marginal) label dependencies



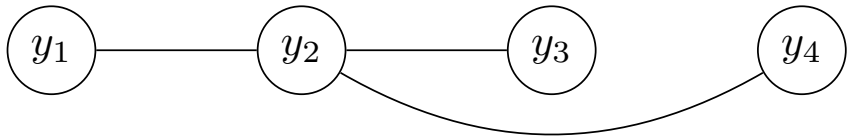
2. Find a **maximum spanning tree** (MST)
3. Choose some directionality (a root node)
4. Employ any classifier, e.g., CC with Naive Bayes

[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[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)



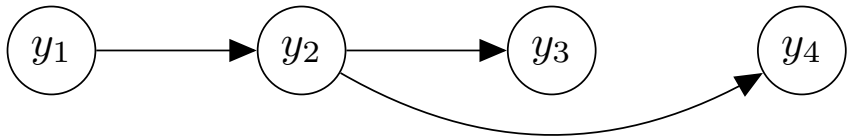
3. Choose some directionality (a root node)
4. Employ any classifier, e.g., CC with Naive Bayes

[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[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)



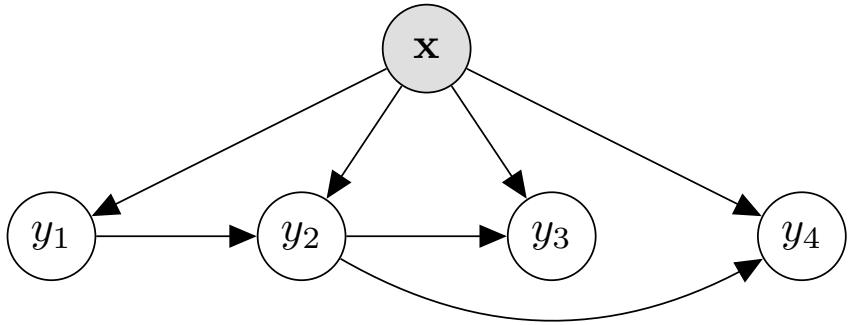
4. Employ any classifier, e.g., CC with Naive Bayes

[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[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

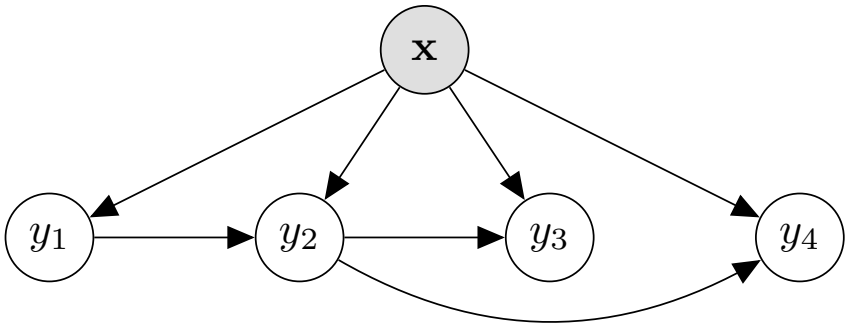


[11] [Zaragoza et al., 2011], IJCAI 11; and related [Alessandro et al., 2013], 'Ensemble of Bayes Nets' for MLC, IJCAI 13 using standard message passing for inference – complexity permitting

Bayesian Chain Classifiers[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.

[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' ($\approx$ Bayesian Network)

LEAD[12] uses an efficient method to measure conditional label dependence:

Proposition

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

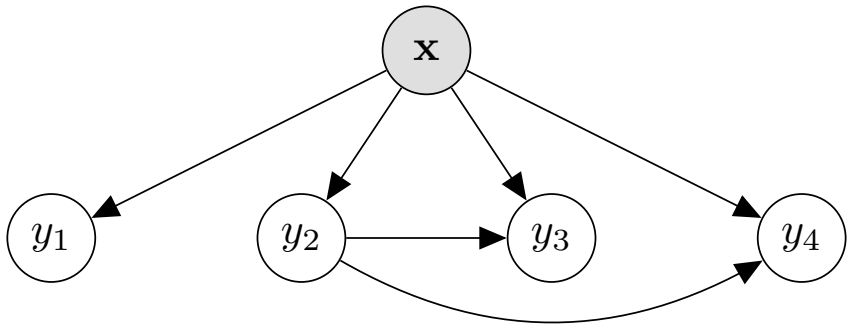
$$y_1 = h_1(\mathbf{x}) + e_1 \qquad \text{and} \qquad y_2 = h_2(\mathbf{x}) + e_2$$

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

Classifier 'Graphs' ($\approx$ Bayesian Network)

LEAD[12] uses an efficient method to measure conditional label dependence:

1. train BR, $h_1, \ldots, h_L$
2. measure dependence among *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.

[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 peformance) to

- model label dependence?
- . . . conditional dependence?
- . . . 'complete' dependence?

Table : Average JACCARD INDEX : rank, under 5×CV [$L = 6$].

	BR	BCC-R	BCC-M	BCC-C	ECC	$\mathrm{P_s}$CC
Music	0.517	0.545 (5)	0.567 (4)	0.582 (3)	0.588 (2)	0.594 (1)
Scene	0.595	0.646 (3)	0.646 (3)	0.643 (5)	0.647 (2)	0.705 (1)

- BR: independent classifiers
- BCC with *R*andom structure / based on *M*arginal and *C*onditional dependence
- ECC: Ensemble of random CC (complete random dependence)
- $\mathrm{P_s}$CC: best of **all** ($6! = 720$) possible chain orders; Bayes-optimal (2^6) inference.

Getting a good Classifier 'Graph'

Is it necessary (for best peformance) 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 [$L = 6$].

	BR	BCC-*R*	BCC-*M*	BCC-*C*	ECC	P_sCC
Music	0.517	0.545 (5)	0.567 (4)	0.582 (3)	0.588 (2)	0.594 (1)
Scene	0.595	0.646 (3)	0.646 (3)	0.643 (5)	0.647 (2)	0.705 (1)

- BR: independent classifiers
- BCC with *R*andom structure / based on *M*arginal and *C*onditional dependence
- ECC: Ensemble of random CC (complete random dependence)
- P_sCC: best of **all** ($6! = 720$) possible chain orders; Bayes-optimal (2^6) inference.

So far . . .

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 $\mathtt{M_s CC}$. Can either

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

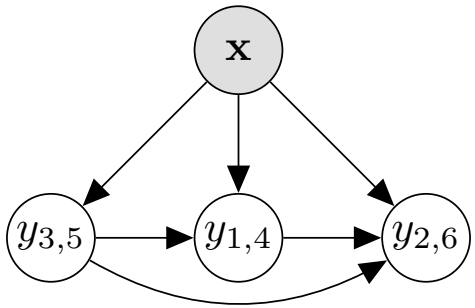
Although 'randomly dependent' models can perform quite well,

- "quite well" $\neq$ very well,
- they are not so interpretable, and
- **are not necessarily the most efficient.**

Super-Label Classifier

A super label is just a class with > 2 possible values, e.g.:

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



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

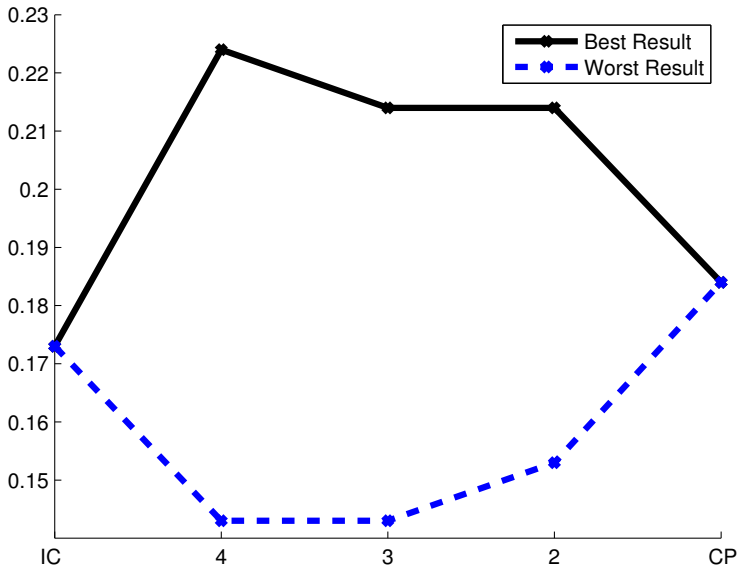


Figure : Performance (Parkinson's data) for $L, L-1, \ldots, 2, 1$ classes, i.e., from BR to LP.

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

	BR	SCC
Exact Match	0.121	**0.169**
Ham. Loss	0.057	**0.054**
Time (s)	43.67	**9.02**
Num. Labels	53	6
Values/label	2	4.5

A model based on label dependence can perform more accurately *and* much faster (including the time to measure dependence).

Some Alternative Approaches

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., [Fürnkranz 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

Summary and Future Work

- 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.

Summary and Future Work

- 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.

End

Thank you!

Questions?