Classification

Multi-labelled Data and Data Streams

Jesse Read

29 November, 2016

Classification

$\mathbf{x} =$ 

Binary classification

$$y \in \{\texttt{sunset}, \texttt{non\_sunset}\}$$

Classification

$\mathbf{x} =$ 

Multi-*class* classification

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

Classification

$\mathbf{x} =$ 

Multi-label classification

$$\mathbf{y} \subseteq \{\texttt{sunset}, \texttt{people}, \texttt{foliage}, \texttt{beach}, \texttt{urban}, \texttt{field}\}$$
$$\in \{0,1\}^6 \qquad = [1,0,1,0,0,0]$$

i.e., multiple labels per instance instead of a single label.

Introduction

	$K = 2$	$K > 2$
$L = 1$ (single-label)	**binary**	**multi-class**
$L > 1$ (multi-label)	**multi-label**	**multi-output**†

† also known as multi-label, multi-target, multi-dimensional.

Figure: For L labels (target variables), each of K values.

Example Dataset

X_1	X_2	X_3	X_4	X_5	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	?

$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ • dataset

$\mathbf{x}_i = [x_1^{(i)}, \ldots, x_D^{(i)}] \in \mathbb{R}^D$ • i-th instance, $\mathcal{X} = \mathbb{R}^D$

$y_i \in \{0, 1\}$ • class, $\mathcal{Y} = \{0, 1\}$

$h : \mathcal{X} \to \mathcal{Y}$ • model

$\hat{y} = h(\tilde{\mathbf{x}})$ • classification

$\epsilon = E(\hat{y}, y)$ • evaluation

Single-label vs. Multi-label

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

X_1	X_2	X_3	X_4	X_5	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	?

Table: Multi-label $Y \subseteq \{\lambda_1, \ldots, \lambda_L\}$

X_1	X_2	X_3	X_4	X_5	Y
1	0.1	3	1	0	$\{\lambda_2, \lambda_3\}$
0	0.9	1	0	1	$\{\lambda_1\}$
0	0.0	1	1	0	$\{\lambda_2\}$
1	0.8	2	0	1	$\{\lambda_1, \lambda_4\}$
1	0.0	2	0	1	$\{\lambda_4\}$
0	0.0	3	1	1	?

Single-label vs. Multi-label

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

X_1	X_2	X_3	X_4	X_5	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	?

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

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2	Y_3	Y_4
1	0.1	3	1	0	0	1	1	0
0	0.9	1	0	1	1	0	0	0
0	0.0	1	1	0	0	1	0	0
1	0.8	2	0	1	1	0	0	1
1	0.0	2	0	1	0	0	0	1
0	0.0	3	1	1	?	?	?	?

Multi-label Learning

The task of building a model to map D inputs to L outputs:

$\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ • dataset

$\mathbf{x}_i = [x_1^{(i)}, \ldots, x_D^{(i)}] \in \mathbb{R}^D$ • i-th instance, $\mathcal{X} = \mathbb{R}^D$

$\mathbf{y}_i = [y_1^{(i)}, \ldots, y_L^{(i)}]$ • label assignment $y_j \in \{0, 1\}$, $\mathcal{Y} \in \{0, 1\}^2$

$h : \mathcal{X} \to \mathcal{Y}$ • multi-label model

$\hat{\mathbf{y}} = h(\tilde{\mathbf{x}})$ • multi-label classification

$\epsilon = E(\hat{\mathbf{y}}, \mathbf{y})$ • multi-label evaluation

Multi-label Learning

$$\mathcal{L} = \{\texttt{sunset}, \texttt{people}, \texttt{foliage}, \texttt{beach}, \texttt{urban}, \texttt{field}\} \quad (L = 6)$$

$$\mathbf{x}_i = \quad \text{  }$$

$$\begin{aligned}
\hat{\mathbf{y}}_i &= h(\mathbf{x}_i) \\
&= [1, 0, 1, 0, 0, 0] \Leftrightarrow \{\texttt{sunset}, \texttt{foliage}\} \\
&\in \{0, 1\}^6 \Leftrightarrow \hat{Y}_i \subseteq \mathcal{L}
\end{aligned}$$

i.e., multiple labels per instance instead of a single label.

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

Text Categorization / Recommendation

For example, the IMDb dataset: Textual movie plot summaries associated with genres (labels),

Text Categorization / Recommendation

For example, the IMDb dataset: Textual movie plot summaries associated with genres (labels),

i	*abandoned* X_1	*accident* X_2	$\dots$	*violent* X_{1000}	*wedding* X_{1001}	horror Y_1	romance Y_2	$\dots$	comedy Y_{27}	action Y_{28}
1	1	0	$\dots$	0	1	0	1	$\dots$	0	0
2	0	1	$\dots$	1	0	1	0	$\dots$	0	0
3	0	0	$\dots$	0	1	0	1	$\dots$	0	0
4	1	1	$\dots$	0	1	1	0	$\dots$	0	1
5	1	1	$\dots$	0	1	0	1	$\dots$	0	1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
120919	1	1	$\dots$	0	0	0	0	$\dots$	0	1

Labelling E-mails

Enron, e-mails messages made public from the Enron corporation.

- For example, the *Enron* e-mails multi-labelled to 53 categories by the *UC Berkeley Enron Email Analysis Project*

Company Business, Strategy, etc.
Purely Personal
Empty Message
Forwarded email(s)
...
company image – current
...
Jokes, humor (related to business)
...
Emotional tone: worry / anxiety
Emotional tone: sarcasm
...
Emotional tone: shame
Company Business, Strategy, etc.

Labelling Images

Images are labelled to indicate

- multiple concepts
- multiple objects
- multiple people

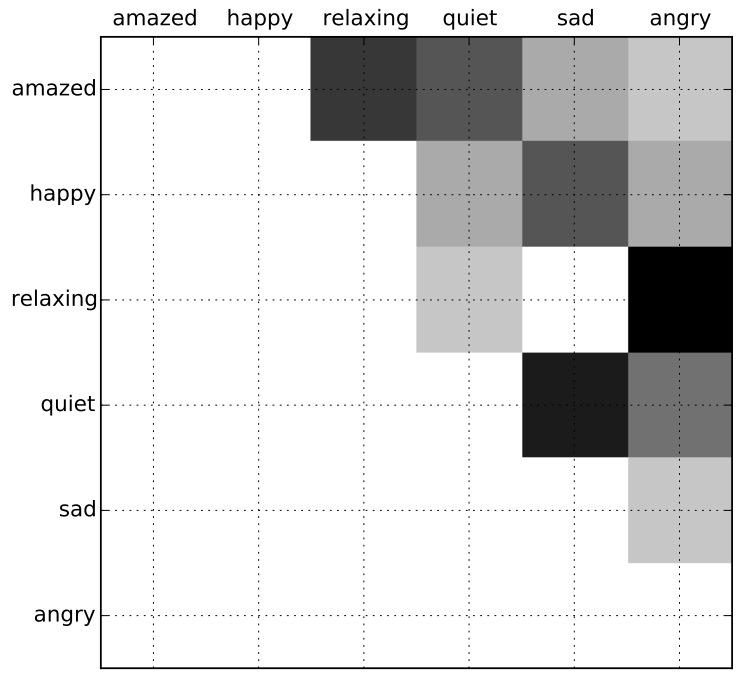
e.g., Associating Scenes with concepts

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

Labelling Audio

For example, labelling music with emotions, concepts, etc.

- amazed-surprised
- happy-pleased
- relaxing-calm
- quiet-still
- sad-lonely
- angry-aggressive



Medical

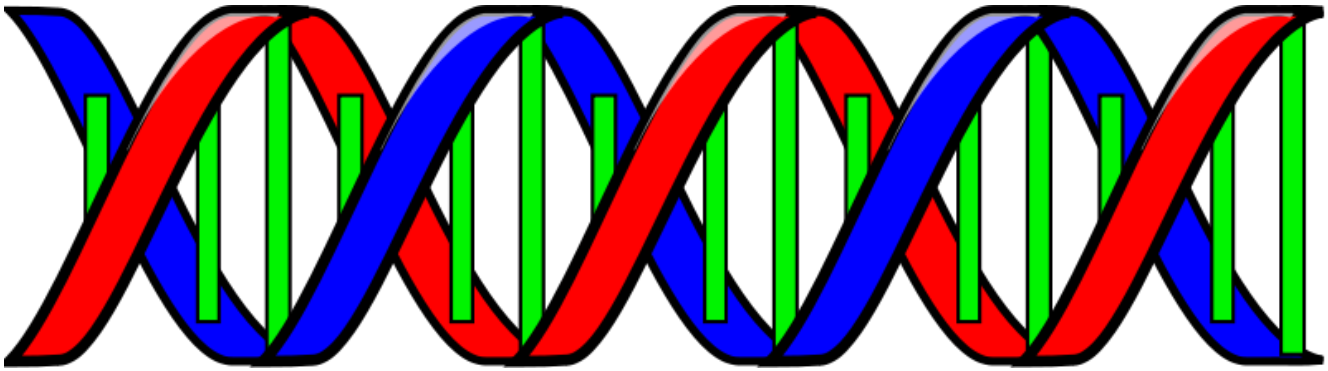
Medical Diagnosis

- medical history, symptoms $\rightarrow$ diseases / ailments

e.g., Medical dataset,

- clinical free text reports by radiologists
- label assignment out of 45 ICD-9-CM codes

Bioinformatics

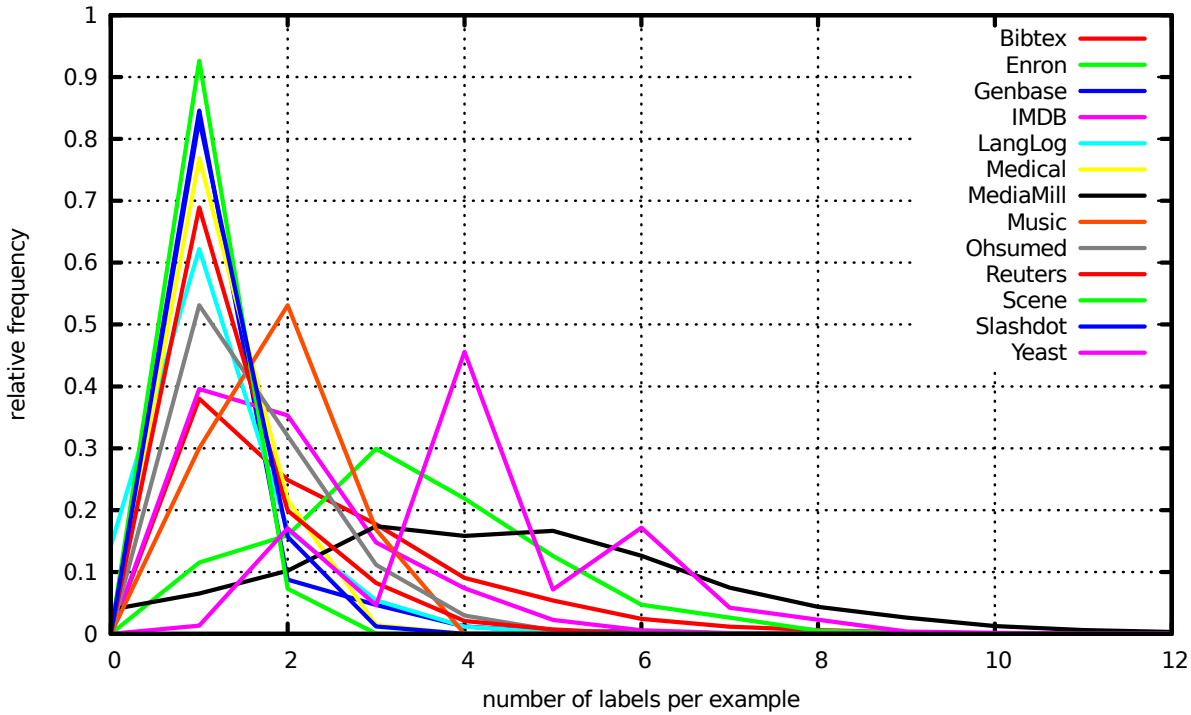


- Genes are associated with biological functions (not in a one-one mapping).
- E.g. the Yeast dataset: $2,417$ genes, described by 103 attributes, labeled into 14 groups of the FunCAt functional catalogue.

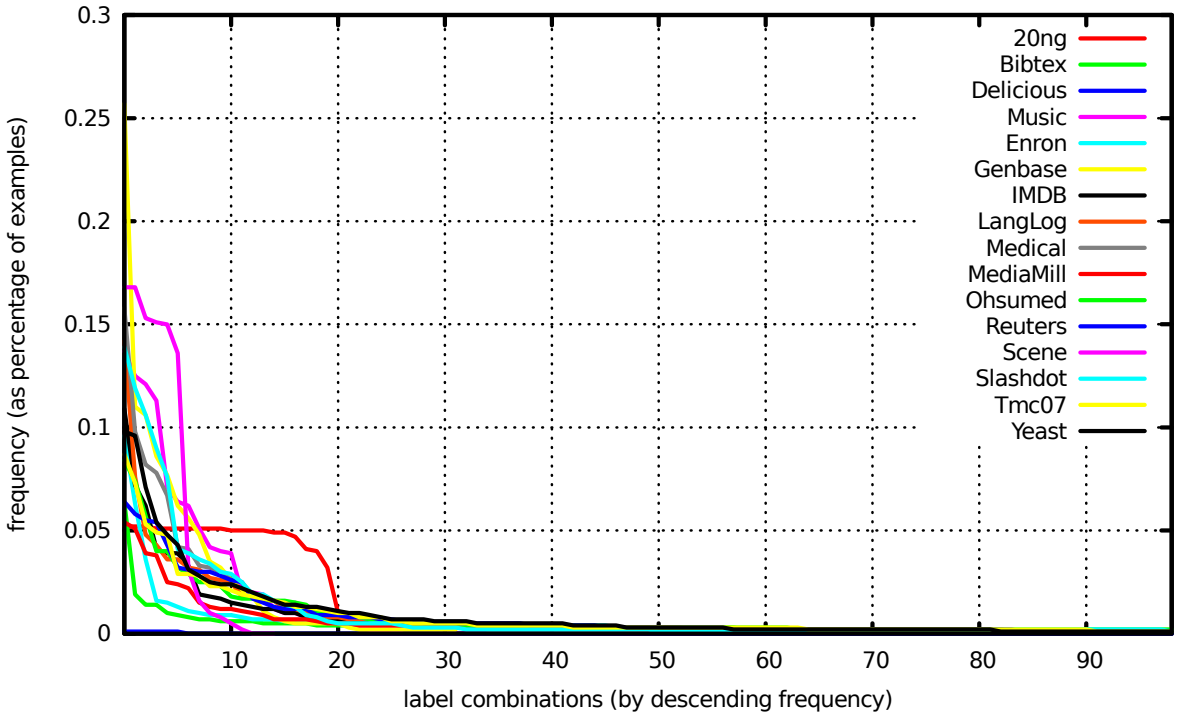
Multi-label Datasets

	N	L	D	L.Card.	Type
Music	593	6	72	1.87	audio
Scene	2407	6	294	1.07	image
Yeast	2417	14	103	4.24	biology
Genbase	661	27	1185	1.25	biology
Medical	978	45	1449	1.25	medical/text
Slashdot	3782	22	1079	1.18	text
Enron	1702	53	1001	3.38	text
LangLog	1460	75	1004	1.18	text
Reuters	6000	103	500	1.46	text
OHSUMED	13929	23	1002	1.66	text
TMC2007	28596	22	500	2.16	text
IMDB	120919	28	1001	2.00	text
Bibtex	7395	159	1836	2.40	text
MediaMill	43907	101	120	4.38	video
Delicious	16105	983	500	19.02	text

Distribution of labelling



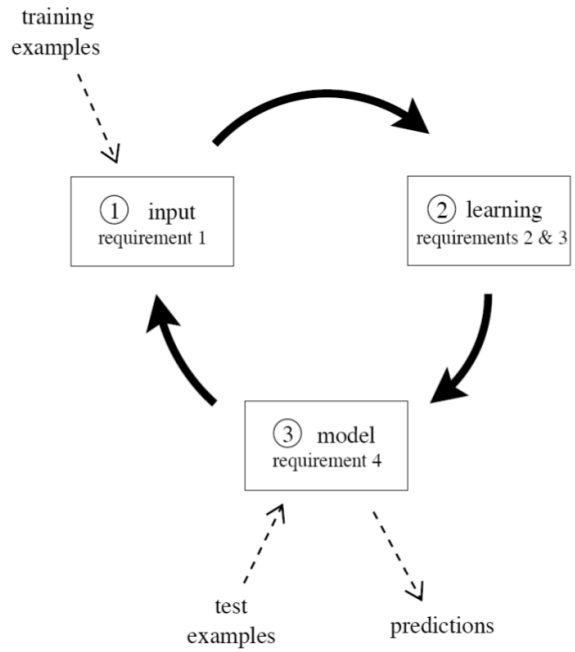
Distribution of label *combinations*



Learning in Data Streams

Setting:

- sequence is potentially infinite
- high speed of arrival
- stream is one-way



Examples,

- Document classification
- Demand prediction
- Intrusion detection
- Pollution and environmental monitoring
- …

Demand Prediction

Outputs (labels) represent the demand at multiple points.

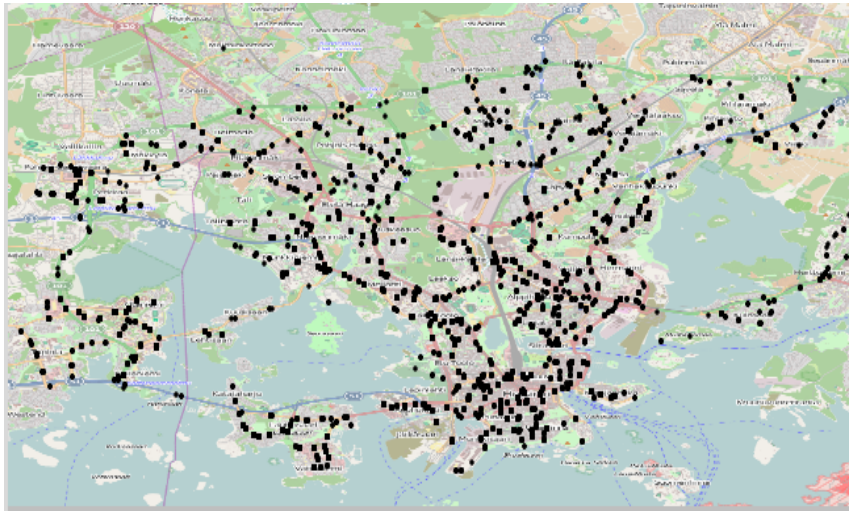


Figure: Bus stops in the greater Helsinki region. The *Kutsuplus* taxi service could be called to any of these.

Each stop is a label! We have L stops, wish to predict stops that will experience high demand in the next 15 minutes, given time of day, day of the week, etc.

Localization and Tracking

Outputs represent points in space which may contain an object ($y_j = 1$) or not ($y_j = 0$). Sensor readings $\mathbf{x}$.

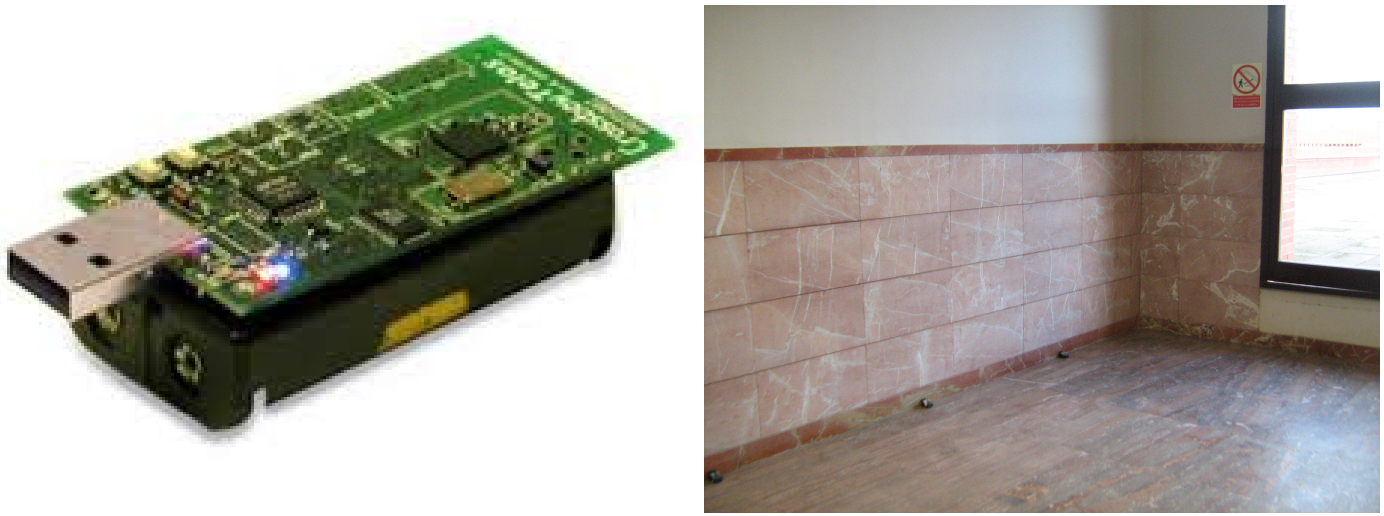


Figure: Modelled on a real-world scenario; a room with a single light source and a number of light sensors.

We are interested in predicting, for each label $[y_1, \ldots, y_L]$,

$$y_j = 1 \quad \bullet \text{ if } j\text{-th tile occupied}$$

Localization and Tracking

Outputs represent points in space which may contain an object ($y_j = 1$) or not ($y_j = 0$). Sensor readings $\mathbf{x}$.

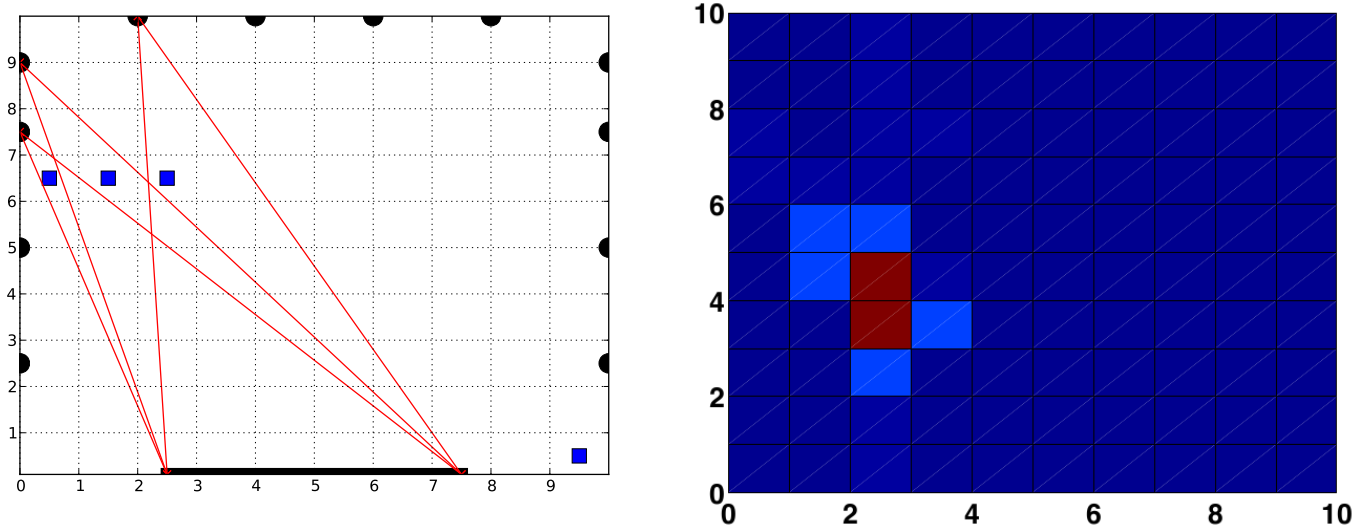


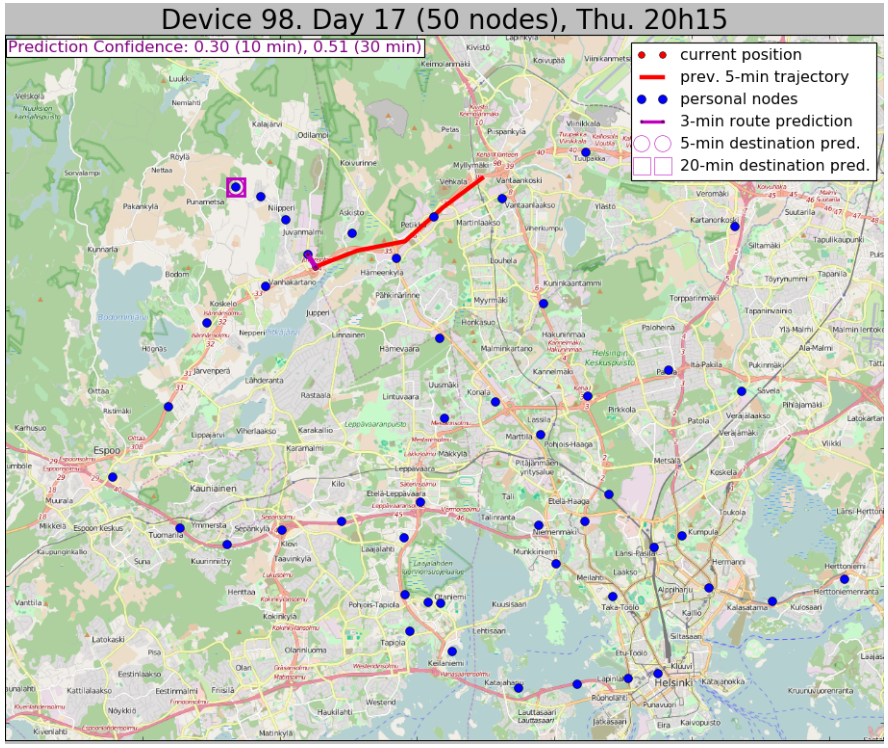
Figure: Modelled on a real-world scenario; a room with a single light source and a number of light sensors.

We are interested in predicting, for each label $[y_1, \ldots, y_L]$,

$y_j = 1$ • if j-th tile occupied

Route/Destination Forecasting

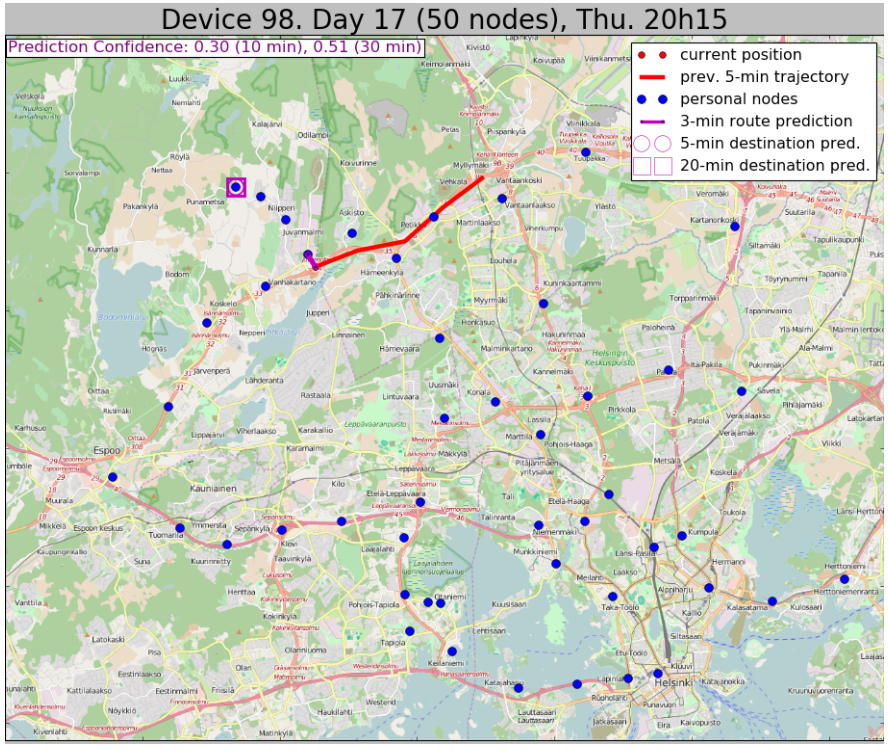
Personal nodes of a traveller and predicted trajectory



- L • number of geographic points of interest
- $\mathbf{x}$ • observed data (e.g., GPS, sensor activity, time of day)
- $p(y_j = 1|\mathbf{x})$ • probability an object is present at the j-th node
- $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$ • training data

Route/Destination Forecasting

Personal nodes of a traveller and predicted trajectory

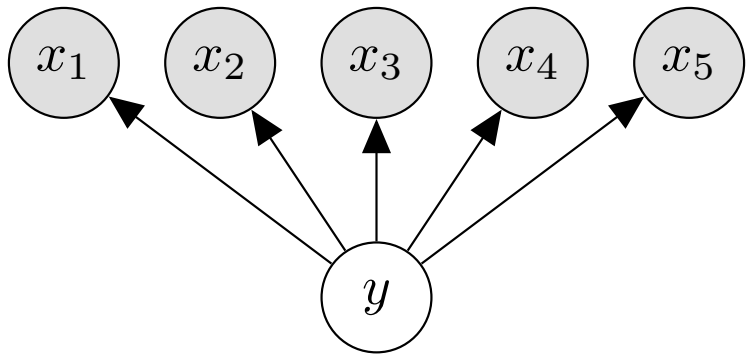


- Many gigabytes of information per traveller,
- Arrives in stream form
- Model frequently needs to be updated

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

(Single-label) Naive Bayes

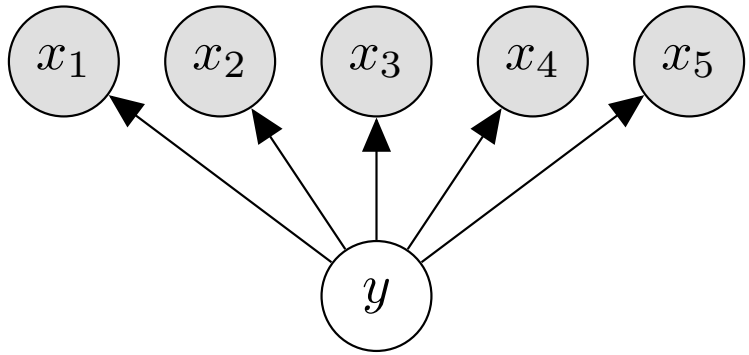


$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(\mathbf{x})}$$ • Bayes rule

where likelihood

$$P(\mathbf{x}|y) = \prod_{d=1}^{D} P(x_d|y)$$

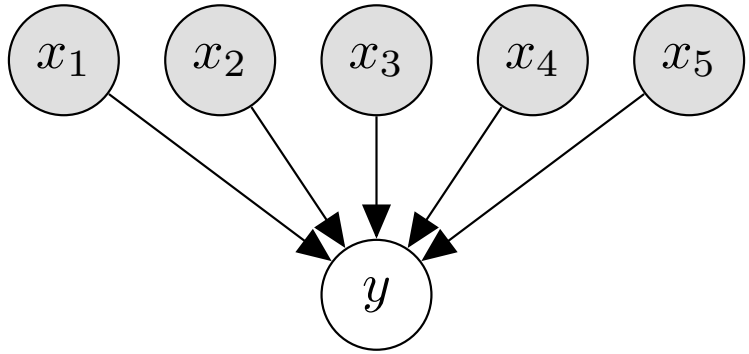
(Single-label) Naive Bayes



$$\hat{y} = h(\mathbf{x}) = \underset{y \in \{0,1\}}{\operatorname{argmax}}\, p(y)p(\mathbf{x}|y) \quad \bullet\ p(y|\mathbf{x}) \propto p(\mathbf{x}|y)p(y) \quad \bullet\ \text{classification}$$

$$= \underset{y \in \{0,1\}}{\operatorname{argmax}}\, p(y) \prod_{d=1}^{D} p(x_d|y)$$

Logistic Regression



$$p(y = 1|\mathbf{x}) \approx p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x})}$$

We take the gradient of the (log) error

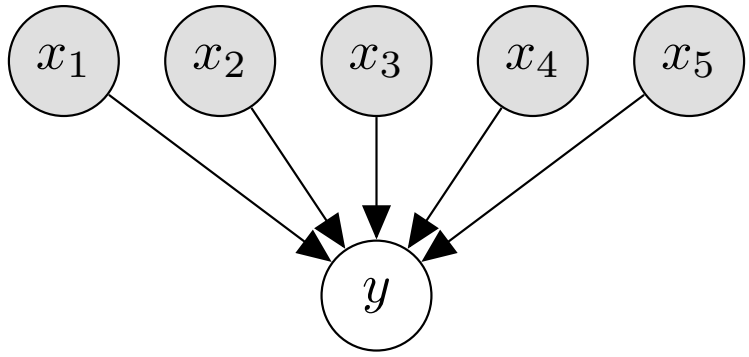
$$E(\boldsymbol{\theta}) = -\log \prod_{i=1}^{N} \left[(p_i)^{y_i} \cdot (1 - p_i)^{1-y_i} \right] = \sum_{i=1}^{N} E_i(\boldsymbol{\theta})$$

and we get a weight-update, to *descend* the error gradient

$$\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \lambda \nabla E_i(\boldsymbol{\theta})$$

i.e., (stochastic/*incremental*) gradient descent.

Logistic Regression



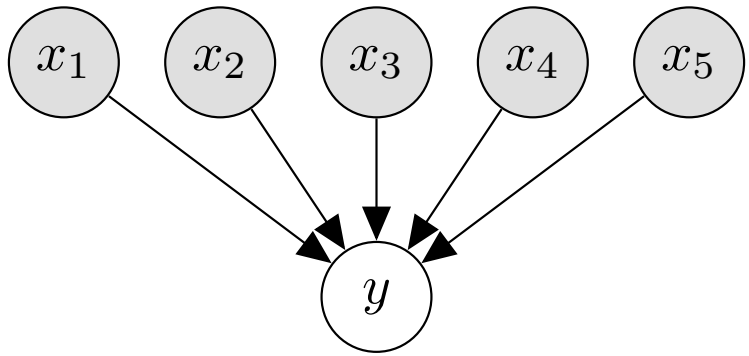
$$\hat{y} = h(\mathbf{x}) = \underset{y \in \{0,1\}}{\operatorname{argmax}}\, p(y|\mathbf{x}) \quad \bullet \text{ classification}$$

$$p(y = 1|\mathbf{x}) = p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x})}$$

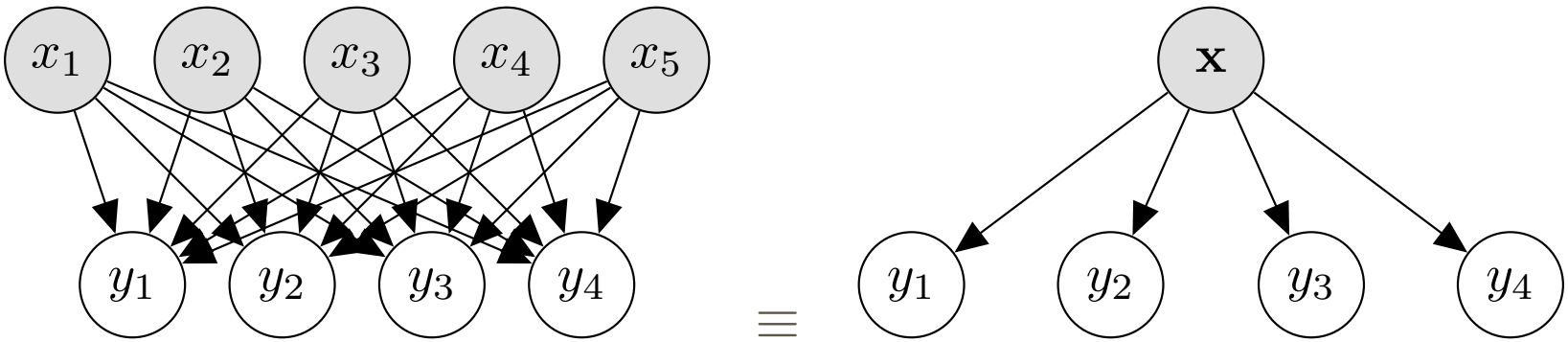
$$p(y = 0|\mathbf{x}) = 1 - p(\mathbf{x}; \boldsymbol{\theta})$$

Moving focus to the labels . . .

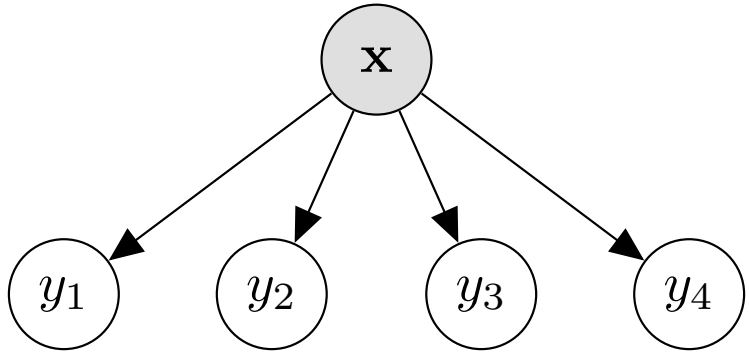
From single label. . .



to multi-label . . .



The Binary Relevance approach



$$\hat{y}_j = h_j(\mathbf{x}) = \underset{y_j \in \{0,1\}}{\operatorname{argmax}}\, p(y_j|\mathbf{x}) \quad \bullet \text{ for index, } j = 1, \ldots, L$$

and then,

$$\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{h}(\mathbf{x}) = [\hat{y}_1, \ldots, \hat{y}_4] \\
&= \left[\underset{y_1 \in \{0,1\}}{\operatorname{argmax}}\, p(y_1|\mathbf{x}), \cdots, \underset{y_4 \in \{0,1\}}{\operatorname{argmax}}\, p(y_4|\mathbf{x}) \right] \\
&= \left[h_1(\mathbf{x}), \cdots, h_4(\mathbf{x}) \right]
\end{aligned}$$

BR *Transformation*

1. Transform dataset ...

$\mathbf{X}$	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

...into L separate binary problems (one for each label)

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

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

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

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

2. and train with any off-the-shelf binary base classifier.

Why Not Binary Relevance?

BR ignores label dependence, i.e.,

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

which may not always hold!

Example (Film Genre Classification)

$$p(y_{\texttt{romance}}|\mathbf{x}) \neq p(y_{\texttt{romance}}|\mathbf{x}, y_{\texttt{horror}})$$

Why Not Binary Relevance?

BR ignores label dependence, i.e.,

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

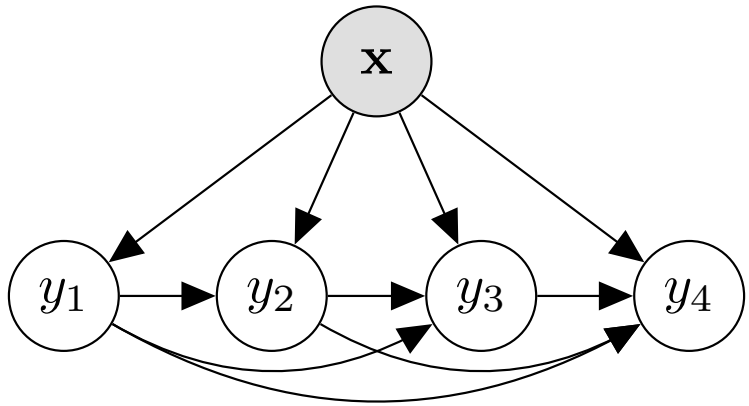
which may not always hold!

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

	L	BR	MCC
Music	6	0.30	**0.37**
Scene	6	0.54	**0.68**
Yeast	14	0.14	**0.23**
Genbase	27	0.94	**0.96**
Medical	45	0.58	**0.62**
Enron	53	0.07	**0.09**
Reuters	101	0.29	**0.37**

Classifier Chains (CC)

Modelling label dependence,



- The prediction of first label is used as input (i.e., as an additional feature) to predict the second label
- The prediction of first and second labels are used as input

...

i.e., labels are predicted together.

CC Transformation

Similar to BR: make L binary problems, but include previous predictions as feature attributes,

$\mathbf{X}$	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0
$\tilde{\mathbf{x}}$	$\hat{y}_1$

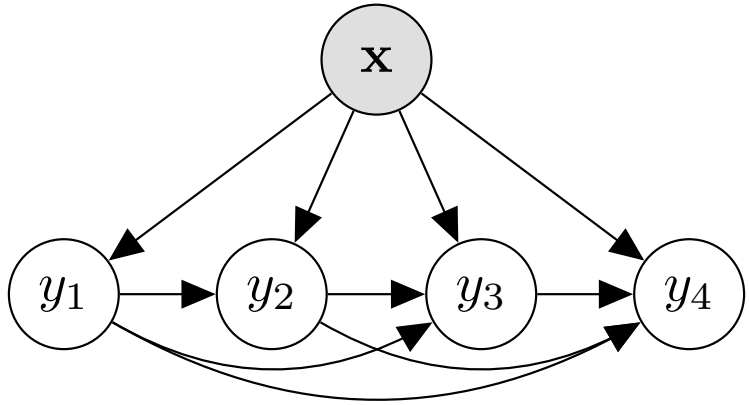
$\mathbf{X}$	Y_1	Y_2
$\mathbf{x}^{(1)}$	0	1
$\mathbf{x}^{(2)}$	1	0
$\mathbf{x}^{(3)}$	0	1
$\mathbf{x}^{(4)}$	1	0
$\mathbf{x}^{(5)}$	0	0
$\tilde{\mathbf{x}}$	$\hat{y}_1$	$\hat{y}_2$

$\mathbf{X}$	Y_1	Y_2	Y_3
$\mathbf{x}^{(1)}$	0	1	1
$\mathbf{x}^{(2)}$	1	0	0
$\mathbf{x}^{(3)}$	0	1	0
$\mathbf{x}^{(4)}$	1	0	0
$\mathbf{x}^{(5)}$	0	0	0
$\tilde{\mathbf{x}}$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$

$\mathbf{X}$	Y_1	Y_3	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
$\tilde{\mathbf{x}}$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$

and, like binary relevance, apply a base classifier for each label.

Greedy CC



L classifiers for L labels. For test instance $\tilde{\mathbf{x}}$, classify

1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}})$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1)$
3. $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2)$
4. $\hat{y}_4 = h_4(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2, \hat{y}_3)$

and return

$$\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_L]$$

Probabilistic Classifier Chains (PCC)

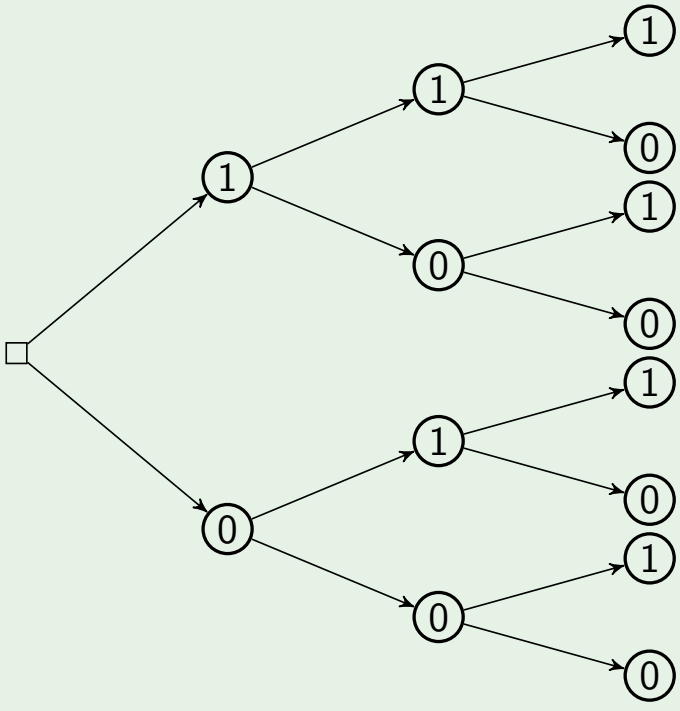
$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \{0,1\}^L}{\operatorname{argmax}} \, p(\mathbf{y}|\mathbf{x})$$

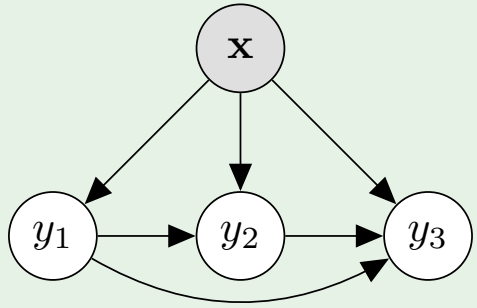
where (for example, $L = 2$),

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= p(y_1, y_2|\mathbf{x}) \\ &= \frac{p(y_1, \mathbf{x})}{p(\mathbf{x})} \frac{p(y_1, y_2, \mathbf{x})}{p(y_1, \mathbf{x})} \\ &= p(y_1|\mathbf{x}) p(y_2|y_1, \mathbf{x}) \\ \text{(generally, for } L > 2) &= p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1}) \end{aligned}$$

where p must be approximated/learned from training data.

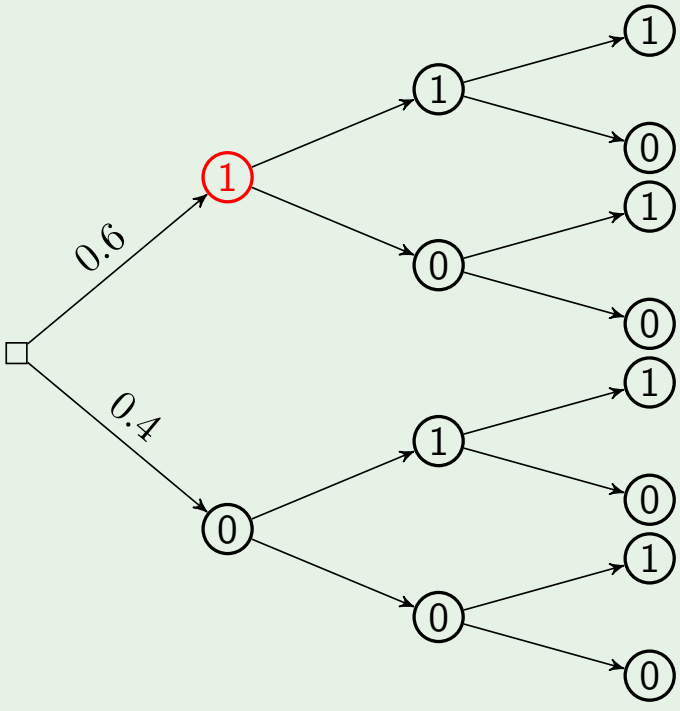
Example (Greedy Inference)

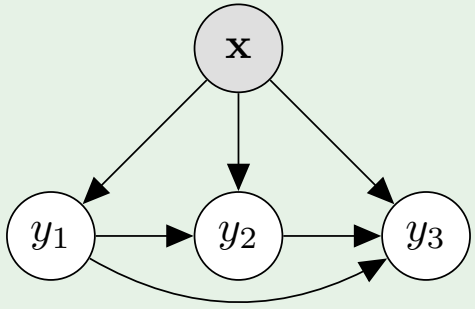




$$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [?, ?, ?]$$

Example (Greedy Inference)

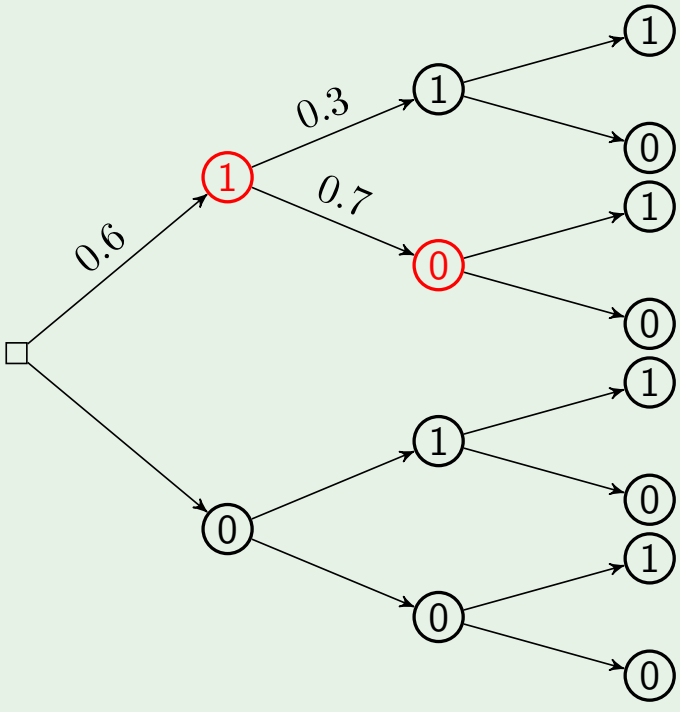


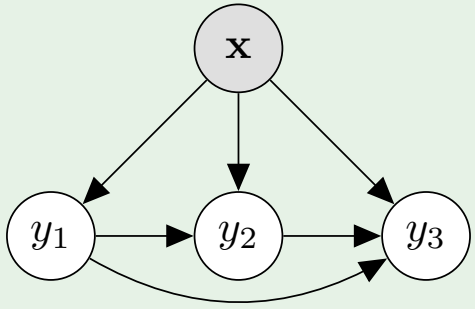


1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = \text{argmax}_{y_1}\, p(y_1|\tilde{\mathbf{x}}) = 1$

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

Example (Greedy Inference)

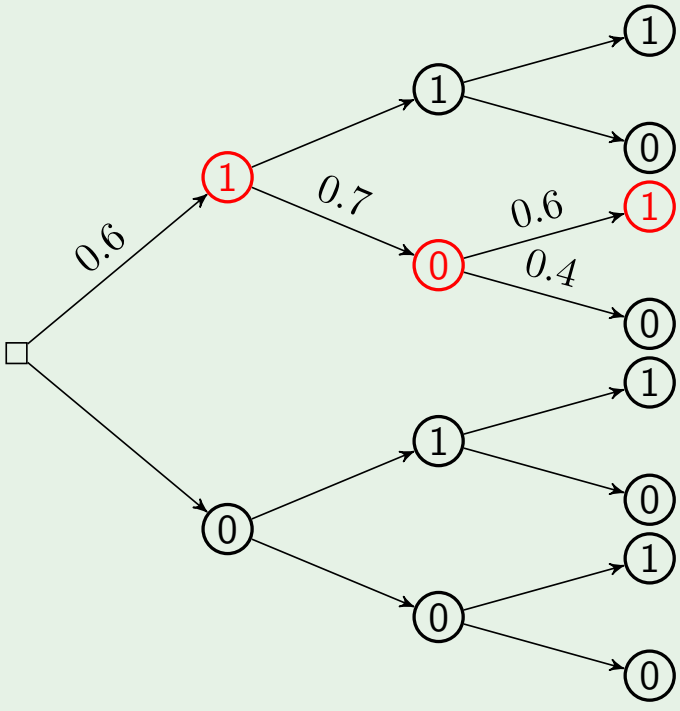




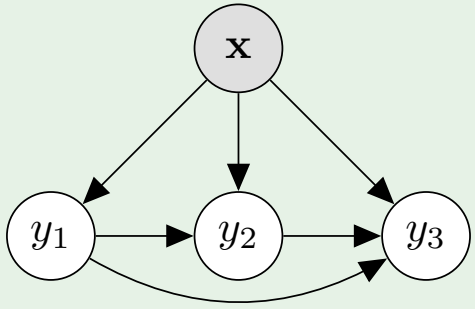
1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = \text{argmax}_{y_1}\, p(y_1|\tilde{\mathbf{x}}) = 1$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1) = \ldots = 0$

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

Example (Greedy Inference)

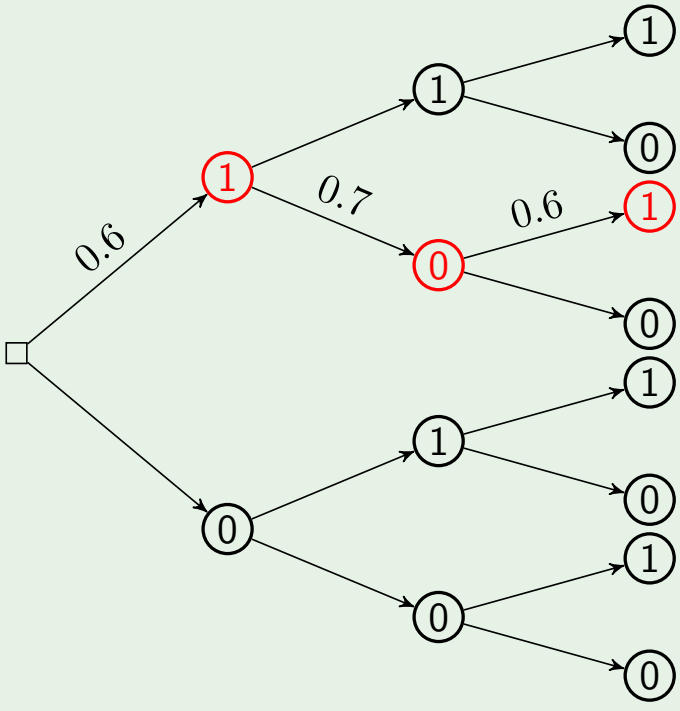


$\hat{\mathbf{y}} = \mathbf{h}(\tilde{\mathbf{x}}) = [1, 0, \mathbf{1}]$

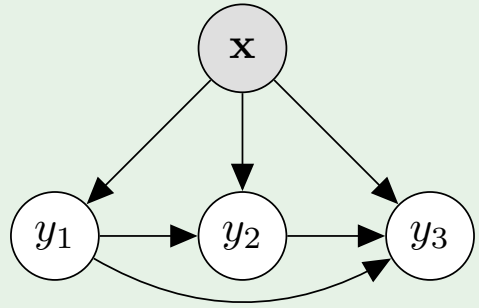


1. $\hat{y}_1 = h_1(\tilde{\mathbf{x}}) = \text{argmax}_{y_1}\, p(y_1|\tilde{\mathbf{x}}) = 1$
2. $\hat{y}_2 = h_2(\tilde{\mathbf{x}}, \hat{y}_1) = \ldots = 0$
3. $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2) = \ldots = 1$

Example (Greedy Inference)



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



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. $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2) = \ldots = 1$

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

Bayes Optimal PCC

Recall ...

$$\hat{\mathbf{y}} = \underset{\mathbf{y}\in\{0,1\}^L}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}) \neq \Big[\underbrace{\underset{y_1\in\{0,1\}}{\operatorname{argmax}}\, p(y_1|\mathbf{x})}_{\hat{y}_1}, \cdots, \underbrace{\underset{y_4\in\{0,1\}}{\operatorname{argmax}}\, p(y_4|\hat{y}_1,\ldots,\hat{y}_3,\mathbf{x})}_{\hat{y}_4} \Big]$$

For all $[y_1, y_2, y_3, y_4] \in \{0,1\}^L$:

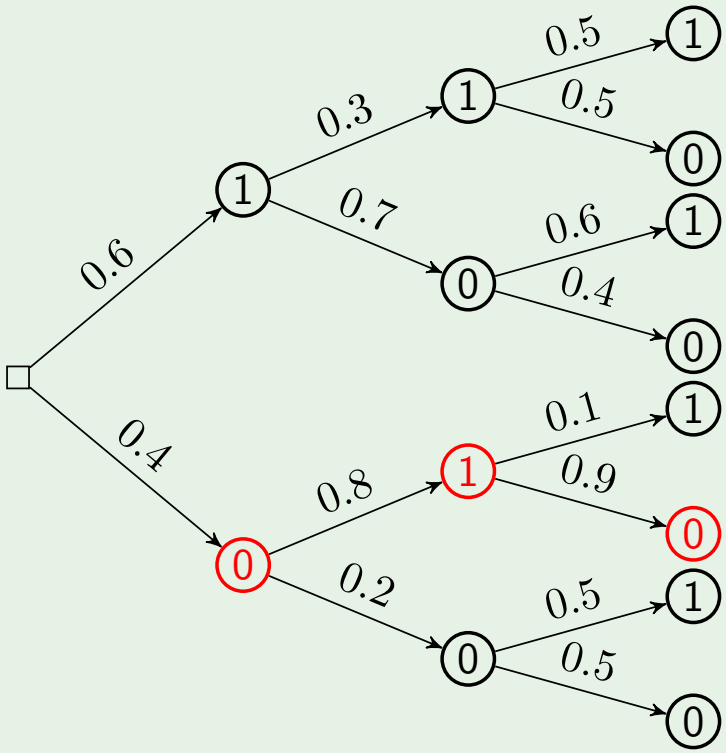
1. $p_1(y_1 = \mathbf{x})$
2. $p_2(y_2 = \mathbf{x}|y_1)$
3. $p_3(y_3 = \mathbf{x}|y_1, y_2)$
4. $p_4(y_4 = \mathbf{x}|y_1, y_2, y_3)$

and then

$$\text{return} \underset{\mathbf{y}\in\{0,1\}^L}{\operatorname{argmax}} \prod_{j=1}^{4} p_j$$

Bayes Optimal PCC

Example (Bayes-Optimal Inference)

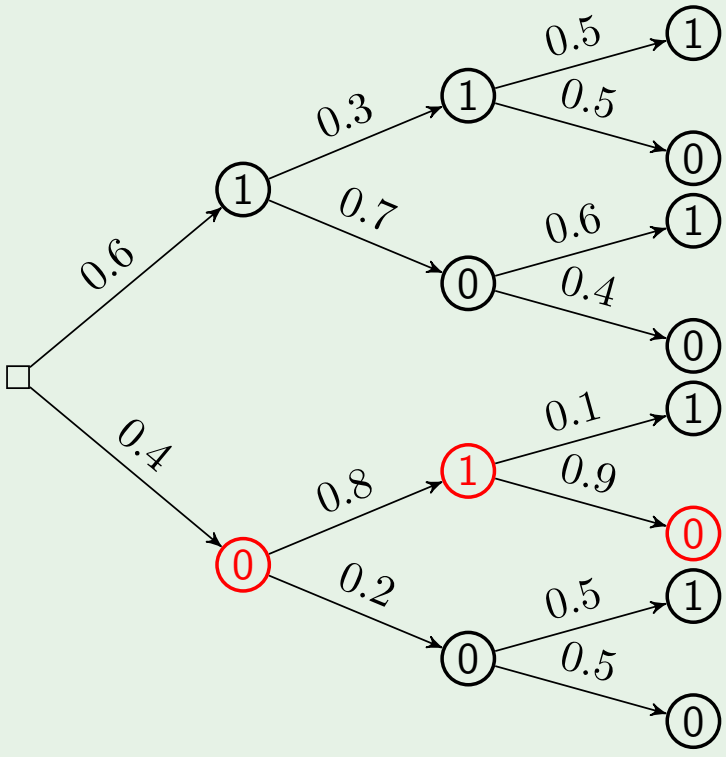


1. $p(\mathbf{y} = [0,0,0]) = 0.040$
2. $p(\mathbf{y} = [0,0,1]) = 0.040$
3. $p(\mathbf{y} = [0,1,0]) = 0.288$
4. $\ldots$
6. $p(\mathbf{y} = [1,0,1]) = 0.252$
7. $\ldots$
8. $p(\mathbf{y} = [1,1,1]) = 0.090$

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

Bayes Optimal PCC

Example (Bayes-Optimal Inference)



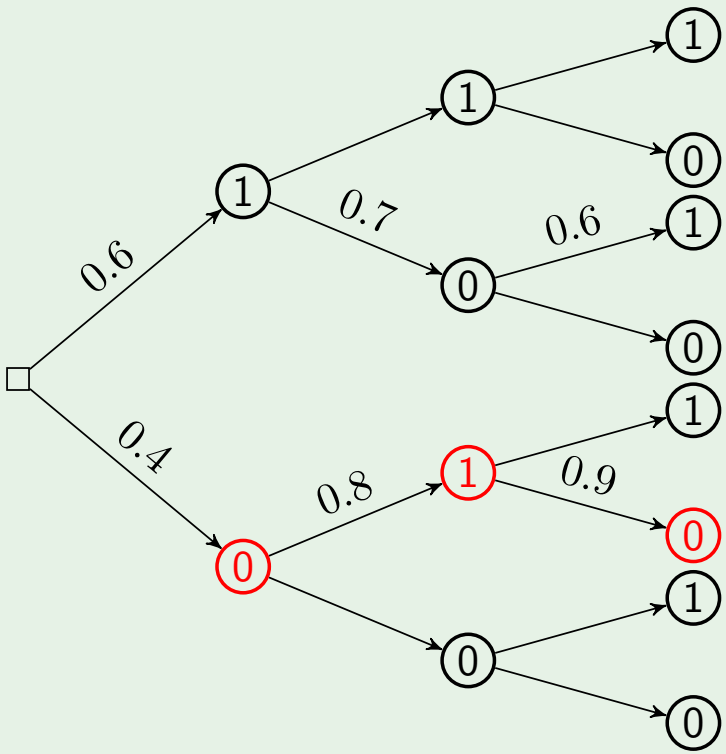
1. $p(\mathbf{y} = [0,0,0]) = 0.040$
2. $p(\mathbf{y} = [0,0,1]) = 0.040$
3. $p(\mathbf{y} = [0,1,0]) = 0.288$
4. $\ldots$
6. $p(\mathbf{y} = [1,0,1]) = 0.252$
7. $\ldots$
8. $p(\mathbf{y} = [1,1,1]) = 0.090$

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

- Search space of $\{0,1\}^L$ paths!

Monte-Carlo search CC (MCC)

Example (Monte-Carlo Inference)



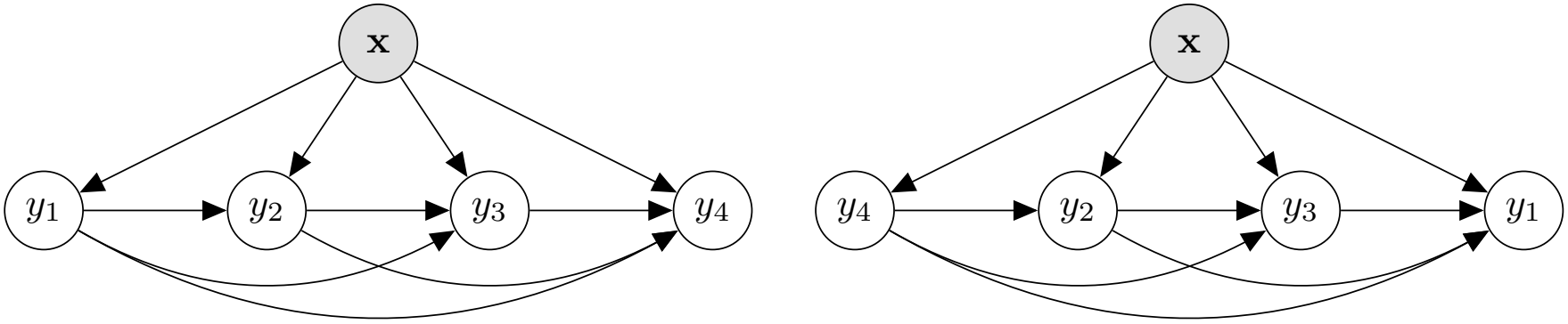
Sample T times ...

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

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

- Tractable, with similar accuracy to (Bayes Optimal) PCC
- Can use other search algorithms, e.g., beam search

Does Label-*order* Matter?



vs

In theory, models are equivalent, since

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

- **but** we are estimating p from finite and noisy data; thus

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

- and in the greedy case,

$$\hat{p}(y_2|y_1,\mathbf{x}) \approx \hat{p}(y_2|\hat{y}_1,\mathbf{x}) = \hat{p}(y_2|y_1 = \underset{y_1}{\operatorname{argmax}}\, \hat{p}(y_1|\mathbf{x})|\mathbf{x})$$

Does Label-*order* Matter?

In theory, models are equivalent, since

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

- **but** we are estimating p from finite and noisy data; thus

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

- and in the greedy case,

$$\hat{p}(y_2|y_1,\mathbf{x}) \approx \hat{p}(y_2|\hat{y}_1,\mathbf{x}) = \hat{p}(y_2|y_1 = \operatorname*{argmax}_{y_1} \hat{p}(y_1|\mathbf{x})|\mathbf{x})$$

The approximations cause high variance on account of error propagation. We can

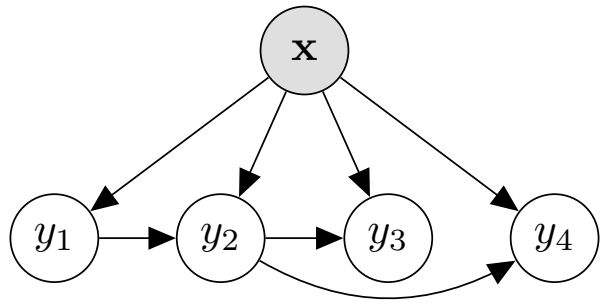
1. use an ensemble of (randomly ordered) classifier chains
2. search space of chain orders

Structure also matters!

We can formulate any structure,

$$\hat{y}_j = \underset{y_j \in \{0,1\}}{\operatorname{argmax}}\, p(y_j | \mathbf{x}, \mathsf{pa}(y_j))$$

where $\mathsf{pa}(y_j)$ = parents of node j.

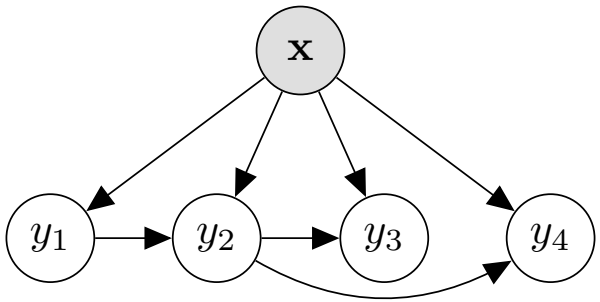


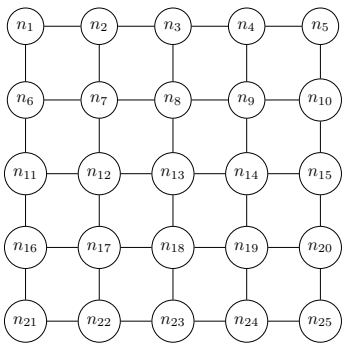
Structure also matters!

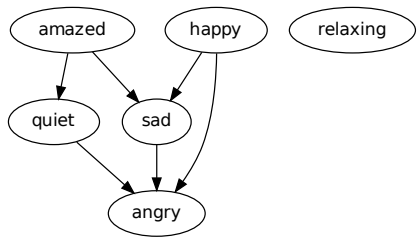
We can formulate any structure,

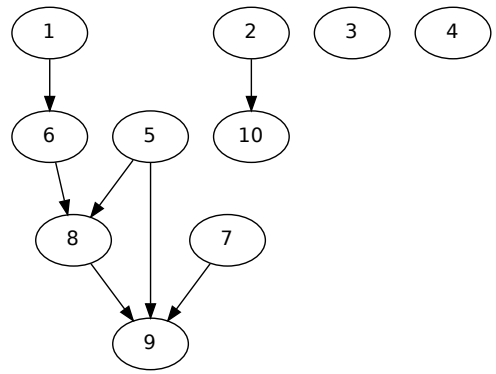
$$\hat{y}_j = \underset{y_j \in \{0,1\}}{\operatorname{argmax}}\, p(y_j | \mathbf{x}, \mathsf{pa}(y_j))$$

where $\mathsf{pa}(y_j)$ = parents of node j.



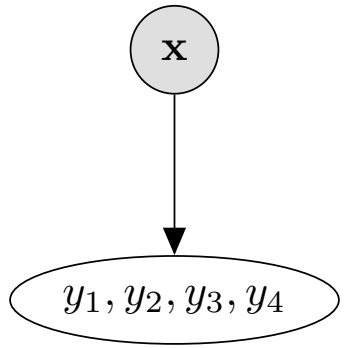






Label Powerset Method (LP)

One multi-class problem (taking many values),

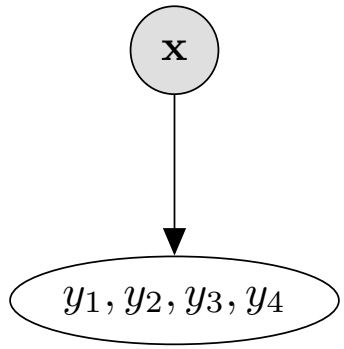


$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}) \quad \bullet \text{ where } |\mathcal{Y}| \leq \{0,1\}^L$$

- Each value is a label vector, 2^L in total, but
- typically, $\mathcal{Y} =$ distinct combinations in training set.
- (thus, in practice, $|\mathcal{Y}| \leq$size of training set, and $|\mathcal{Y}| \ll 2^L$)

Label Powerset Method (LP)

One multi-class problem (taking many values),



Example

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

- Each value is a label vector, 2^L in total, but
- typically, $\mathcal{Y}$ = distinct combinations in training set.
- (thus, in practice, $|\mathcal{Y}| \leq$size of training set, and $|\mathcal{Y}| \ll 2^L$)

Label Powerset Method (LP)

1. Transform dataset ...

$\mathbf{X}$	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	1	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

...into a multi-*class* problem, taking 2^L possible values:

$\mathbf{X}$	$Y \in 2^L$
$\mathbf{x}^{(1)}$	0110
$\mathbf{x}^{(2)}$	1000
$\mathbf{x}^{(3)}$	0110
$\mathbf{x}^{(4)}$	1001
$\mathbf{x}^{(5)}$	0001

2. ...and train any off-the-shelf multi-*class* classifier.

Issues with LP

- complexity (up to 2^L combinations)
- imbalance: few examples per class label
- overfitting: how to predict new value?

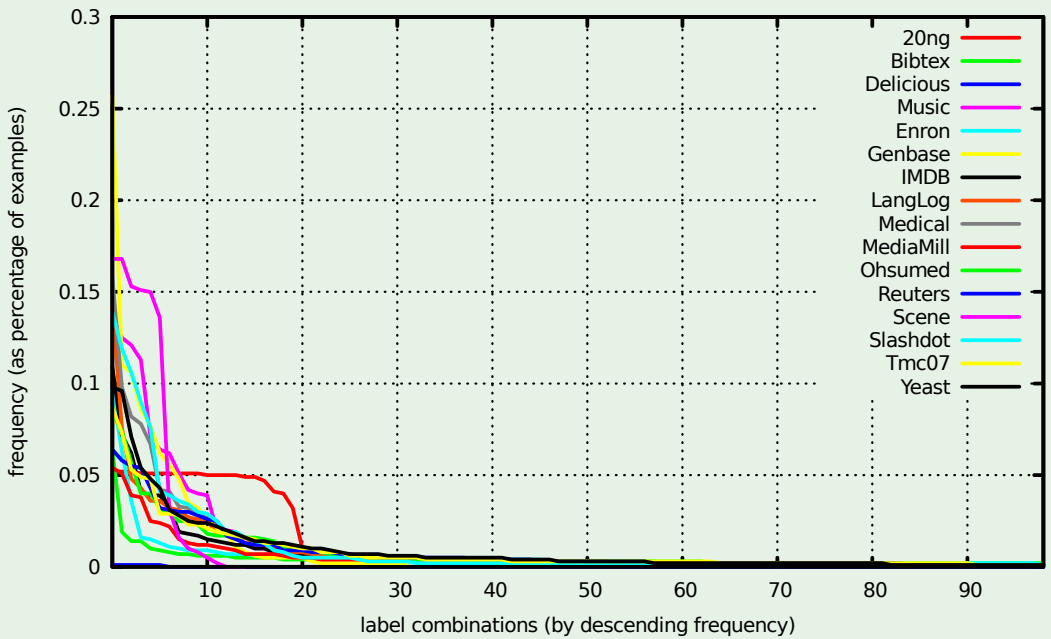
Example

In the Enron dataset, 44% of labelsets are unique (to a single training example or test instance). In del.icio.us dataset, 98% are unique.

Issues with LP

- complexity (up to 2^L combinations)
- imbalance: few examples per class label
- overfitting: how to predict new value?

Example



Note: horizontal axis has been truncated

Meta Labels

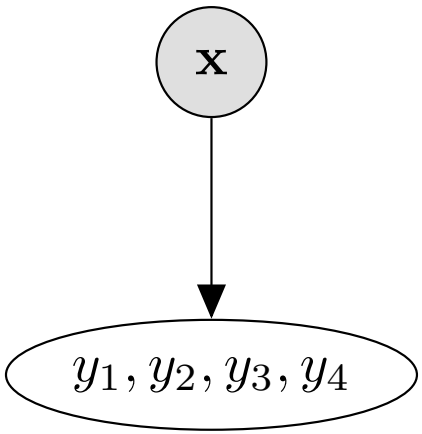
Decompose the labelset into M sets of k labels.

	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$
$h^1(\tilde{\mathbf{x}})$	1		1	
$h^2(\tilde{\mathbf{x}})$		0		0
$\hat{\mathbf{y}}$	1	0	1	0

complexity reduces from $O(2^L)$ to $O(M \cdot 2^k)$, predictive performance usually improves.

Summary: Meta labels vs Connected labels

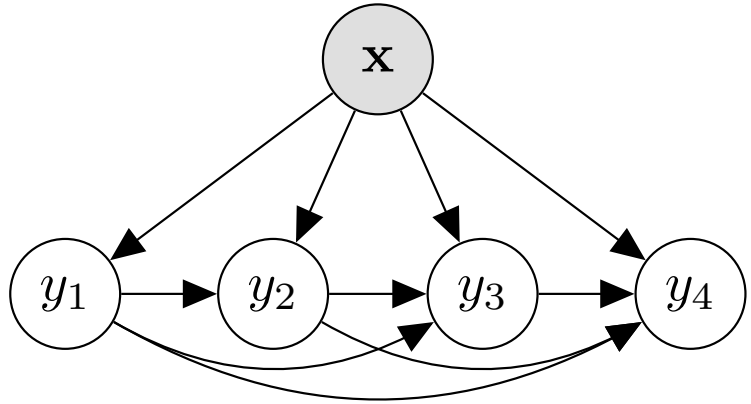
Meta Labels



$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x})$$

- goal: reduce size of $\mathcal{Y}$ (i.e., distinct combinations)

Classifier Chains

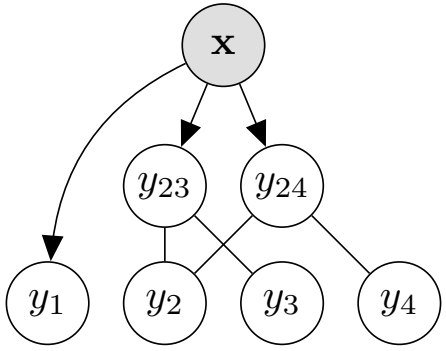


$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \{0,1\}^L}{\operatorname{argmax}}\, \underbrace{p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})}_{\text{chain rule}}$$

- goal: reduce connectivity among $Y_1, \ldots, Y_L$

Summary: Meta labels vs Connected labels

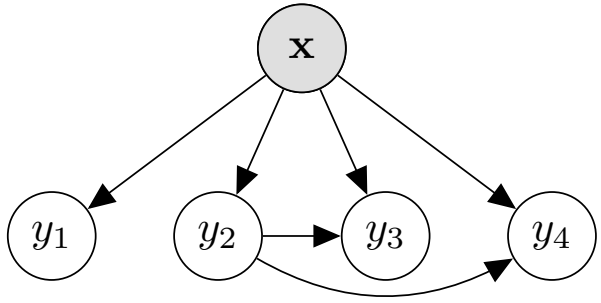
Meta Labels



$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x})$$

- goal: reduce size of $\mathcal{Y}$ (i.e., distinct combinations)

Classifier Chains



$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \{0,1\}^L}{\operatorname{argmax}}\, \underbrace{p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})}_{\text{chain rule}}$$

- goal: reduce connectivity among $Y_1, \ldots, Y_L$

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

Label Dependence

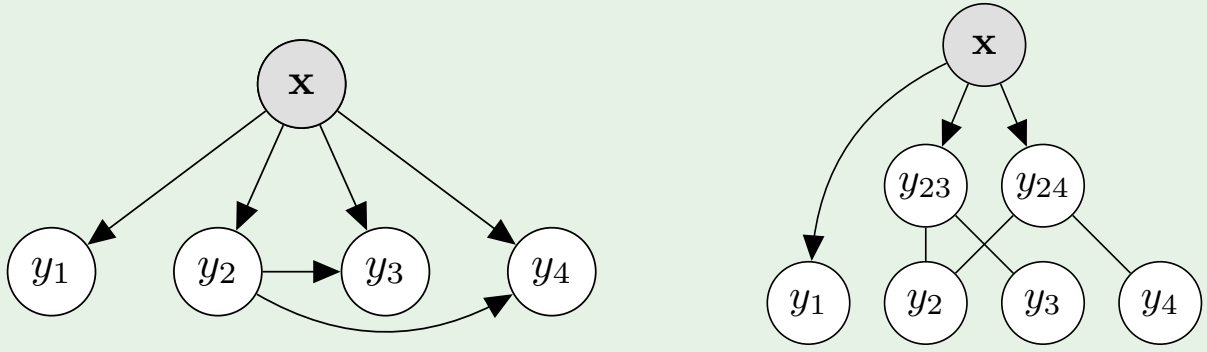
Common approach: Present methods to

1. measure label dependence
2. find a structure that best represents this

and then apply classifiers, compare results to BR.

Example

Dependence detected between y_2 and y_3 – and – y_3 and y_4;



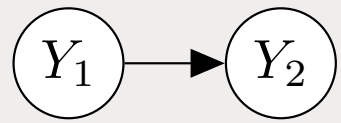
- Link labels (nodes) together in a chained structure; or
- Form label subsets

Marginal label dependence

Marginal dependence

When the joint is **not** the product of the marginals, i.e.,

$$P(y_2) \neq P(y_2|y_1)$$
$$P(y_1)P(y_2) \neq P(y_1, y_2)$$

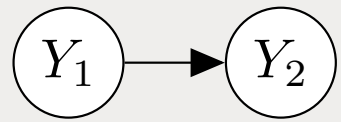


Marginal label dependence

Marginal dependence

When the joint is **not** the product of the marginals, i.e.,

$$P(y_2) \neq P(y_2|y_1)$$
$$P(y_1)P(y_2) \neq P(y_1, y_2)$$



We can estimate from pairwise co-occurrence frequencies in training data, correlation matrix, etc.

Marginal label dependence

Example

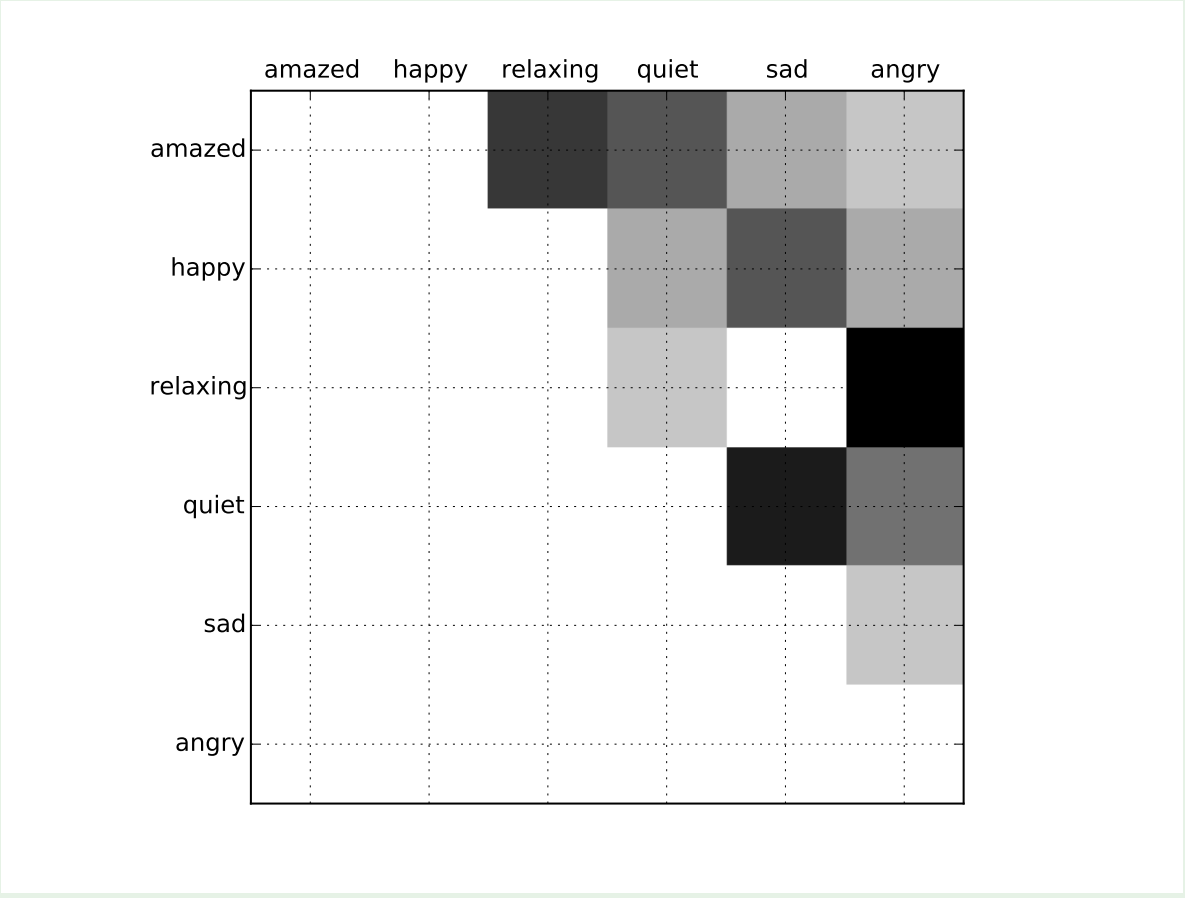


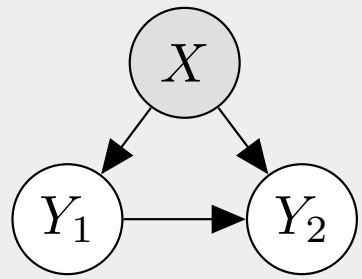
Figure: Music dataset - Mutual Information

Conditional label dependence

Conditional dependence

. . . conditioned on input observation $\mathbf{x}$.

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

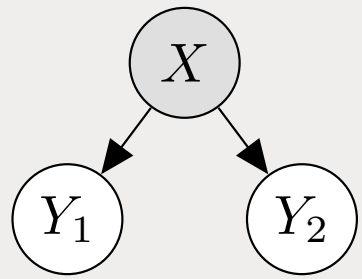


Conditional label dependence

Conditional *in*dependence

. . . conditioned on input observation $\mathbf{x}$.

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



Conditional label dependence

But how to measure $P(y_1|\mathbf{x}, y_2)$? Recall,

$$\hat{y}_1 = h_1(\mathbf{x}) = \underset{y_1 \in \{0,1\}}{\operatorname{argmax}}\, p(y_1|\mathbf{x})$$

where

$$p(y_2 = 1|\mathbf{x}) \approx f_1^{\mathsf{br}}(\mathbf{x})$$
$$p(y_2 = 1|\mathbf{x}, y_1) \approx f_2^{\mathsf{cc}}(\mathbf{x}, y_1)$$

To measure, we have to build and measure models (take into account input). Indication of conditional dependence if

- the performance of LP/CC methods exceed that of BR

$$\propto \left| h_j^{\mathsf{br}}(\mathbf{x}) - h_j^{\mathsf{cc}}(\mathbf{x}, y_k) \right|$$

- errors among the binary models are correlated

$$y_j = h_j(\mathbf{x}) + \epsilon_j$$
$$y_k = h_k(\mathbf{x}) + \epsilon_k$$

Marginal dependence $\neq$ Conditional dependence

x_1	y_1	y_2	$p(x_1, y_1, y_2)$
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

(example from Krzysztof Dembczyński et al. *On label dependence and loss minimization in multi-label classification.* MLJ, 2012).

The marginals are (found by summing out other variables),

$$P(y_1 = 1, y_2 = 1) = \sum_{x' \in \{0,1\}} p(y_1, y_2 | x') = 0.25$$

etc., and since $p(y_1, y_2) = p(y_1)p(y_2)$, independence!

Marginal dependence $\neq$ Conditional dependence

x_1	y_1	y_2	$p(x_1, y_1, y_2)$
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

(example from Krzysztof Dembczyński et al. *On label dependence and loss minimization in multi-label classification.* MLJ, 2012).

But,

$$p(y_1 = 1, y_2 = 1 | x = 1) = 0$$

etc., where there *is* conditional dependence.

Dependence is dependent . . .

Conditional label dependence and the choice of base model are inseparable.

$$y_j = h_j(\mathbf{x}) + \epsilon_j$$
$$y_k = h_k(\mathbf{x}) + \epsilon_k$$

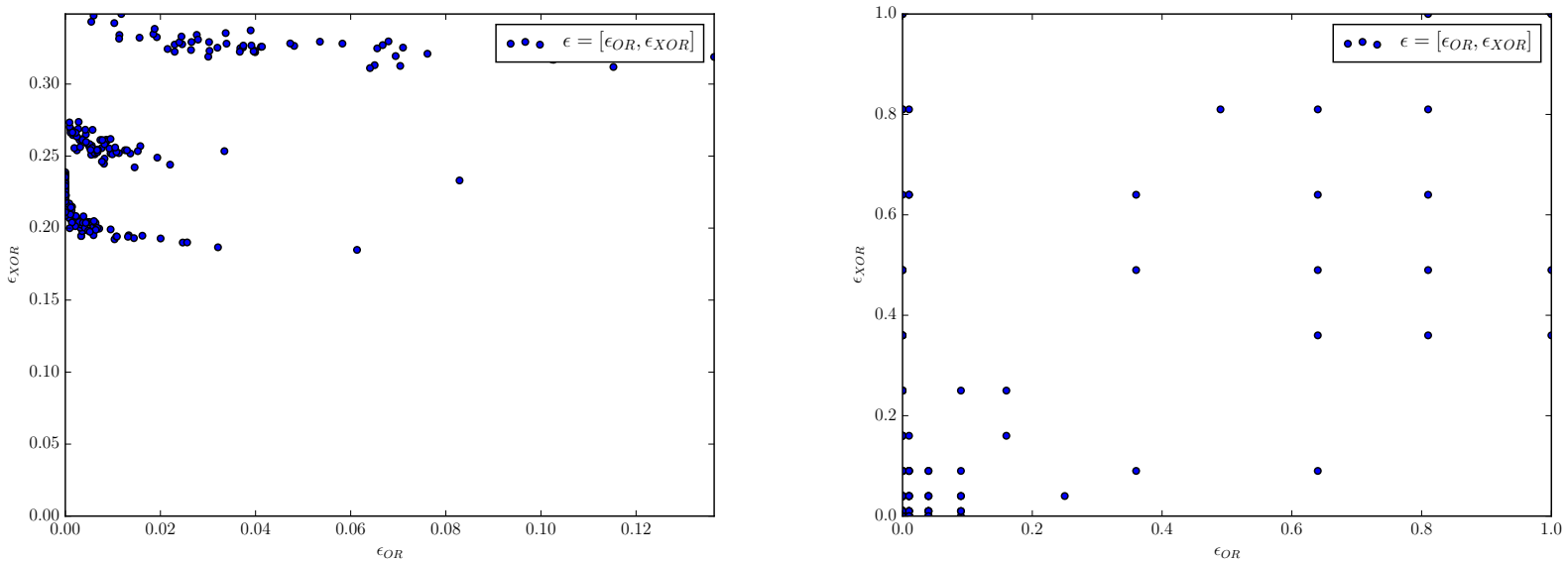


Figure: Errors from logistic regression (left) and decision tree (right).

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

Large scale text classification

Kaggle Competition

The challenge is based on a large dataset created from Wikipedia. The dataset is multi-class, multi-label and hierarchical. The number of categories is roughly 325,000 and number of the documents is 2,400,000.

Scaling Up

LSHTC4: Large Scale Hierarchal Text Classification

A wikipedia-scale problem

- $325,056$ labels
- $2.4M$ examples

- Even with only $1,000$ features, have to learn over $300M$ parameters with BR (linear models)
- ...plus $52,831M$ more with CC
- ...plus ensembles ($\times 10$, $\times 50$?)
- LP transformation generates around $1.47M$ classes

Multi-Label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages

Rahul Agrawal
Microsoft AdCenter
rahulagr@microsoft.com

Archit Gupta
IIT Delhi
archit.gupta@gmail.com

Yashoteja Prabhu
Microsoft Research
yashoteja.prabhu@gmail.com

Manik Varma
Microsoft Research
manik@microsoft.com

ABSTRACT

Recommending phrases from web pages for advertisers to bid on against search engine queries is an important research problem with direct commercial impact. Most approaches have found it infeasible to determine the relevance of all possible queries to a given ad landing page and have focussed on making recommendations from a small set of phrases extracted (and expanded) from the page using NLP and ranking based techniques. In this paper, we eschew this paradigm, and demonstrate that it is possible to efficiently predict the relevant subset of queries from a large set of monetizable ones by posing the problem as a multi-label learning task with each query being represented by a separate label.

We develop Multi-label Random Forests to tackle problems with millions of labels. Our proposed classifier has prediction costs that are logarithmic in the number of labels and can make predictions in a few milliseconds using 10 Gb of RAM. We demonstrate that it is possible to generate training data for our classifier automatically from click logs without any human annotation or intervention. We train our

ad landing page. The advertiser can choose to bid on specific recommendations so as to maximize the ad's chances of getting triggered in response to a relevant search engine query. This is an important research problem from a scientific as well as a commercial perspective as it facilitates the automation of large scale campaign creation and enables both premium and small advertisers to make more informed choices about their bid phrases.

A natural way of formulating the problem, from a machine learning perspective, would be to pose it as a supervised multi-label learning task with each query that could potentially be asked at a search engine corresponding to a separate label. The objective would be to learn a multi-label classifier which could predict the most relevant *set* of labels (queries) in response to a given ad landing page. However, such an approach has traditionally been considered to be infeasible due to four primary reasons. First, supervised learning techniques typically deal with problems involving hundreds to thousands of labels and it is not clear how to formulate problems involving an infinite number of labels. Sec-

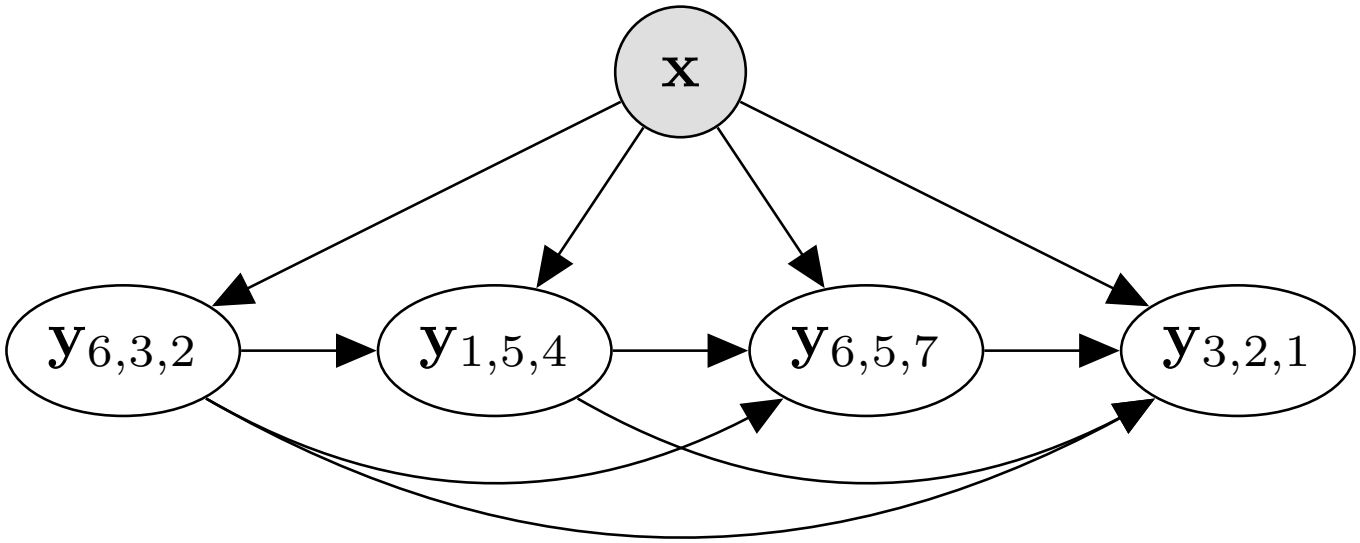
Case study: Winning solution of the LSHTC4 challenge

(reminder: 325, 056 labels, $2.4M$ examples)

1. Ignore the predefined hierarchy
2. Create meta labels
3. . . . from random samples of training data
4. . . . heavily pruned
5. . . . chained together
6. Mix of base classifiers (centroid, decision trees, SVMs)
7. . . . randomization on: splits, pruning, chain links, base classifier parameters
8. Train models across 20 machines, combine votes with meta model.

Chained Meta Labels

Using meta labels makes structure formation simpler (fewer nodes),



Combine many instantiations in large ensemble of models.

Hierarchy and dependence

1 Coarse genre

1.1 Company Business, Strategy, etc. 1.2 Purely Personal 1.3 Personal but in professional context (e.g., it was good working with you) 1.4 Logistic Arrangements (meeting scheduling, technical support, etc) 1.5 Employment arrangements (job seeking, hiring, recommendations, etc) 1.6 Document editing/checking (collaboration) 1.7 Empty message (due to missing attachment) 1.8 Empty message

1.1 Company Business, Strategy, etc.

1.1.1 regulations and regulators (includes price caps) 1.1.2 internal projects – progress and strategy 1.1.3 company image – current 1.1.4 company image – changing / influencing 1.1.5 political influence / contributions / contacts 1.1.6 california energy crisis / california politics 1.1.7 internal company policy 1.1.8 internal company operations 1.1.9 alliances / partnerships 1.1.10 legal advice 1.1.11 talking points 1.1.12 meeting minutes 1.1.13 trip reports

2 Included/forwarded information

2.1 Includes new text in addition to forwarded material 2.2 Forwarded email(s) including replies 2.3 Business letter(s) / document(s) 2.4 News article(s) 2.5 Government / academic report(s) 2.6 Government action(s) (such as results of a hearing, etc) 2.7 Press release(s) 2.8 Legal documents (complaints, lawsuits, advice) 2.9 Pointers to url(s) 2.10 Newsletters 2.11 Jokes, humor (related to business) 2.12 Jokes, humor (unrelated to business) 2.13 Attachment(s) (assumed missing)

3 Emotional tone (if not neutral)

3.1 jubilation 3.2 hope / anticipation 3.3 humor 3.3 camaraderie 3.5 admiration 3.6 gratitude 3.7 friendship / affection 3.8 sympathy / support 3.9 sarcasm 3.10 secrecy / confidentiality 3.11 worry / anxiety 3.12 concern 3.13 competitiveness / aggressiveness 3.14 triumph / gloating 3.15 pride 3.16 anger / agitation 3.17 sadness / despair 3.18 shame 3.19 dislike / scorn

Figure: UC Berkeley's Hierarchy for Enron Email Analysis Project

Hierarchy and dependence

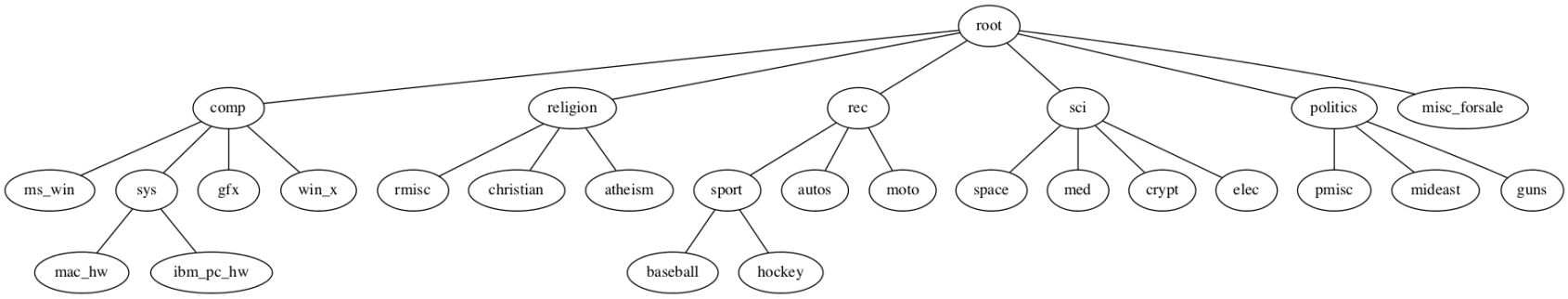


Figure: 20 newsgroups hierarchy

Is a dataset hierarchy a representation of label dependence?

Hierarchy and dependence

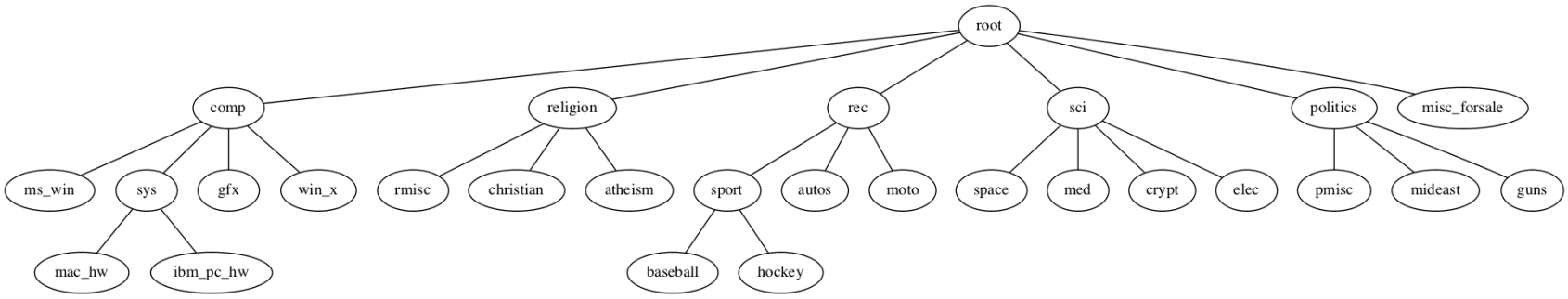


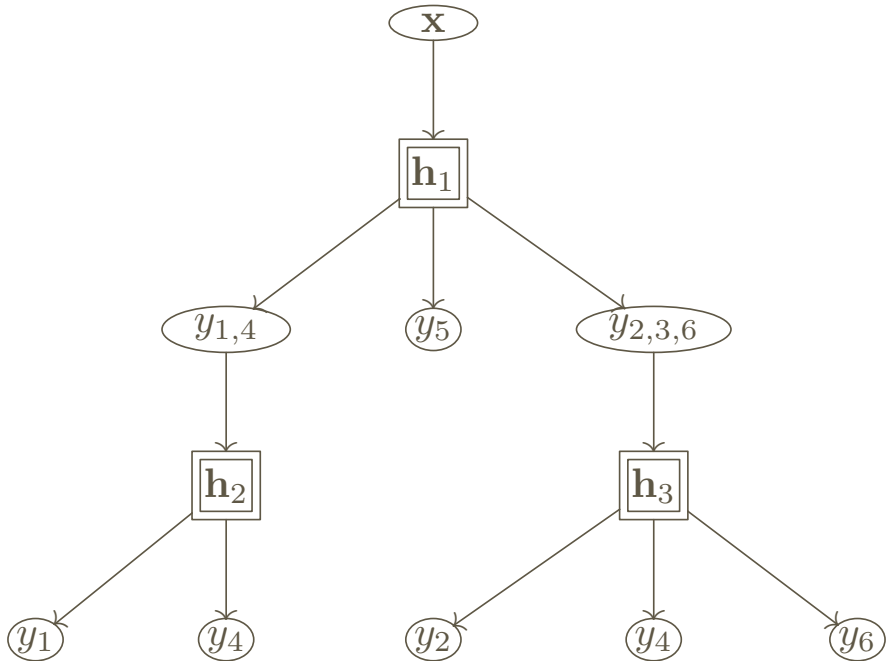
Figure: 20 newsgroups hierarchy

Is a dataset hierarchy a representation of label dependence?

- Yes, but *not necessarily the best one*!
- Hierarchies normally only represent similarities, but exclusivity is also dependence.
- A random hierarchy may perform as well

Hierarchy of Meta Labels

The HOMER method:



1. Cluster labels (e.g., randomly, hierarchical k-means, or use pre-defined hierarchy)
2. Apply problem transformation / base classifiers
3. Significant speed improvement, easier to distribute

Implications of Data Streams

Usual implications:

- Prediction must be made immediately
- Learn efficiently – limited memory (on a potentially infinite stream)
- Expect concept drift to occur

Multi-label Implications

- Class imbalance (exacerbated)
- Overfitting (exacerbated)
- Multi-dimensional concept drift
- Labelled examples difficult to obtain
- Dynamic label set
- Dynamic relationships among labels
- Time dependence

Multi-label Learning in Data Streams

To perform better than baseline binary relevance method,

1. ~~Analyze label dependence~~ – No time! Data still arriving! May become invalidated over time
2. ~~Search for a good structure~~ – No time! Data still arriving! And, structure is more difficult to distribute over many nodes.
3. ...

Multi-label Learning in Data Streams

To perform better than baseline binary relevance method,

1. ~~Analyze label dependence~~ – No time! Data still arriving! May become invalidated over time
2. ~~Search for a good structure~~ – No time! Data still arriving! And, structure is more difficult to distribute over many nodes.
3. ...

Still, we can use

- Random and/or sparse structure
- Improved base classifier
- Shared representation to create independence among labels

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

Multi-label Streams Methods

1. Batch-incremental Ensemble
2. Problem transformation with an incremental base learner
3. Multi-label *k*NN
4. Decision Rules
5. Multi-label incremental decision trees
6. Neural networks

Batch-Incremental Ensemble

Build regular multi-label models on batches/windows of instances (typically in a [weighted] ensemble).

$$\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_1, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_2, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_3, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_4}_{\mathbf{h}_1}, \underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_5, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_6, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_7, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_8}_{\mathbf{h}_2}, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_9, \begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{y}}\end{bmatrix}_{10}$$

- build model $\mathbf{h}_1$ on $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_w, \mathbf{y}_w)\}$
- build model $\mathbf{h}_2$ on $\{(\mathbf{x}_{w+1}, \mathbf{y}_{w+1}), \ldots, (\mathbf{x}_{2w}, \mathbf{y}_{2w})\}$
- build model $\mathbf{h}_3$ on $\{(\mathbf{x}_{2w+1}, \mathbf{y}_{2w+1}), \ldots, (\mathbf{x}_{3w}, \mathbf{y}_{3w})\}$
- ...
- ensemble of M models: $\mathbf{h} := \{\mathbf{h}_{t/w-1}, \ldots, \mathbf{h}_{t/w}\}$
- predict via vote :

$$\hat{\mathbf{y}} = \sum_{m=1}^{M} \mathbf{h}_m(\tilde{\mathbf{x}})$$

Batch-Incremental Ensemble

Build regular multi-label models on batches/windows of instances (typically in a [weighted] ensemble).

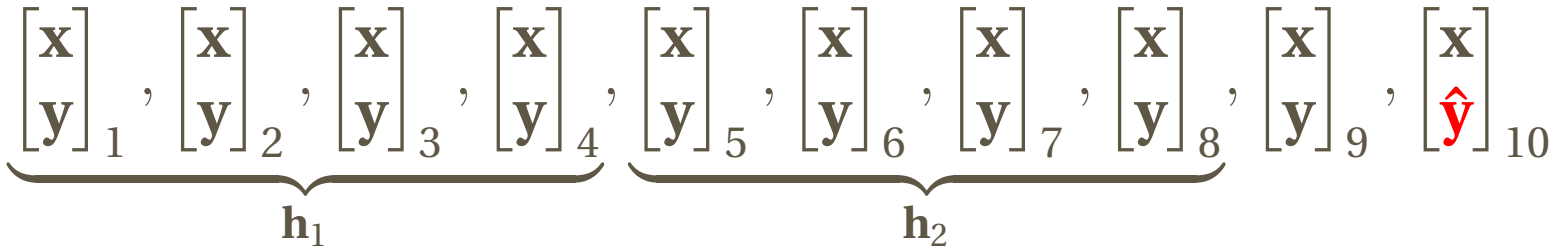
$$\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_1, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_2, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_3, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_4}_{\mathbf{h}_1}, \underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_5, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_6, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_7, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_8}_{\mathbf{h}_2}, \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}_9, \begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{y}}\end{bmatrix}_{10}$$

- build model $\mathbf{h}_1$ on $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_w, \mathbf{y}_w)\}$
- build model $\mathbf{h}_2$ on $\{(\mathbf{x}_{w+1}, \mathbf{y}_{w+1}), \ldots, (\mathbf{x}_{2w}, \mathbf{y}_{2w})\}$
- build model $\mathbf{h}_3$ on $\{(\mathbf{x}_{2w+1}, \mathbf{y}_{2w+1}), \ldots, (\mathbf{x}_{3w}, \mathbf{y}_{3w})\}$
- ...
- ensemble of M models: $\mathbf{h} := \{\mathbf{h}_{t/w-1}, \ldots, \mathbf{h}_{t/w}\}$
- predict via weighted vote (more recent = more relevant):

$$\hat{\mathbf{y}} = \sum_{m=1}^{M} \omega_m \cdot \mathbf{h}_m(\tilde{\mathbf{x}}) \quad \bullet \text{ where } \omega_m \propto m$$

Batch-Incremental Ensemble

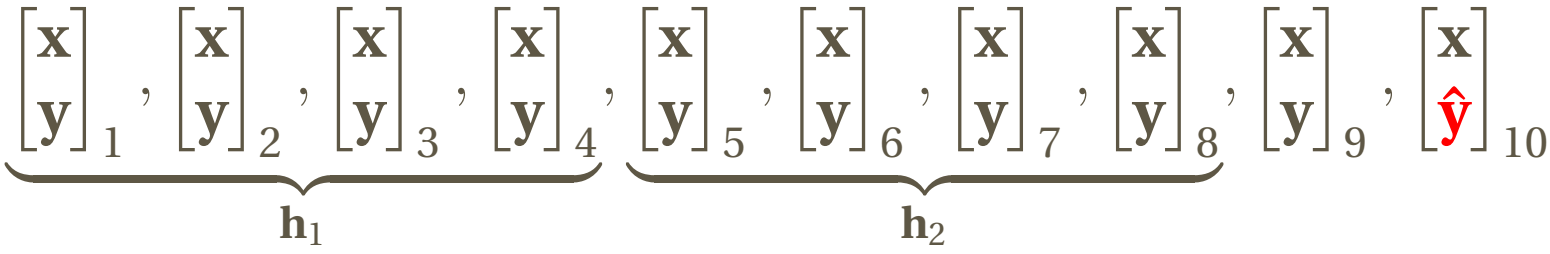
Build regular multi-label models on batches/windows of instances (typically in a [weighted] ensemble).



- A common approach, can be surprisingly effective
- Open choice of base classifier (e.g., C4.5, SVM)
- But what batch size to use?
  - Too small = models are insufficient
  - Too large = slow to adapt (missing new instances!)

Batch-Incremental Ensemble

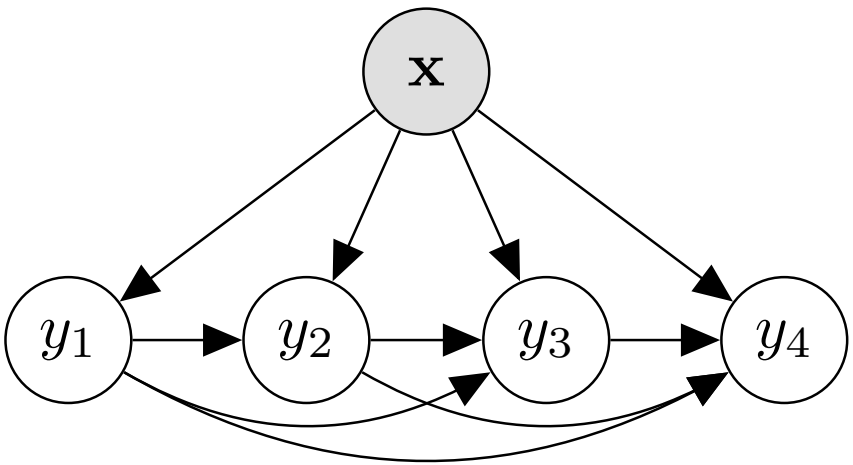
Build regular multi-label models on batches/windows of instances (typically in a [weighted] ensemble).



- A common approach, can be surprisingly effective
- Open choice of base classifier (e.g., C4.5, SVM)
- But what batch size to use?
  - Too small = models are insufficient
  - Too large = slow to adapt (missing new instances!)
- Can have sliding (as opposed to tumbling) windows
  - Extreme case: model built every instance
  - But too many batches = too slow

Problem Transformation with Incremental Base Learner

Use an incremental learner (Naive Bayes, SGD, Hoeffding trees) with any problem transformation method (BR, LP, CC, . . .)



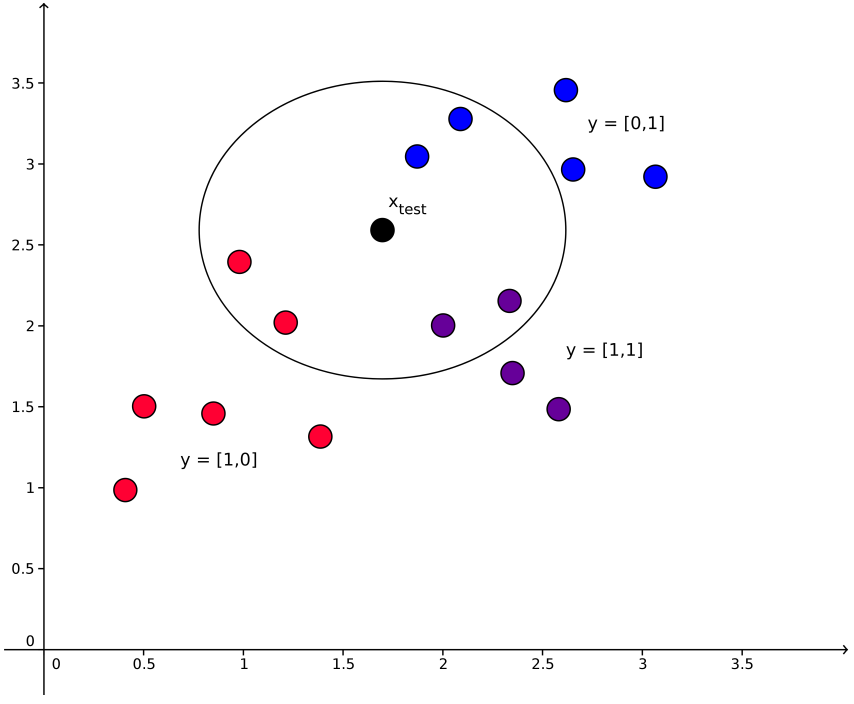
- Simple implementation,
- Risk of overfitting (e.g., with classifier chains)
- Concept drift may invalidate structure
- Limited choice of base learner
  (must be incremental)

Multi-label kNN

Maintain a dynamic buffer of instances, compare each test instance $\tilde{\mathbf{x}}$ to the k neighbouring instances,

$$\hat{y}_j = \begin{cases} 1 & \left(\frac{1}{k} \sum_{t|\mathbf{x}^{(t)} \in \mathsf{Ne}(\tilde{\mathbf{x}})} y_j^{(t)} > 0.5 \right) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathsf{Ne}(\tilde{\mathbf{x}})$ contains the k *closest* points to $\tilde{\mathbf{x}}$ (e.g., Euclidean distance), in our buffer of instances: $\{\mathbf{x}_{t-100}, \mathbf{x}_{t-99}, \ldots, \mathbf{x}_{t-1}\}$



- efficient wrt L
- . . . but not wrt D
- limited buffer size,
- not suitable for all problems / high dimensionality

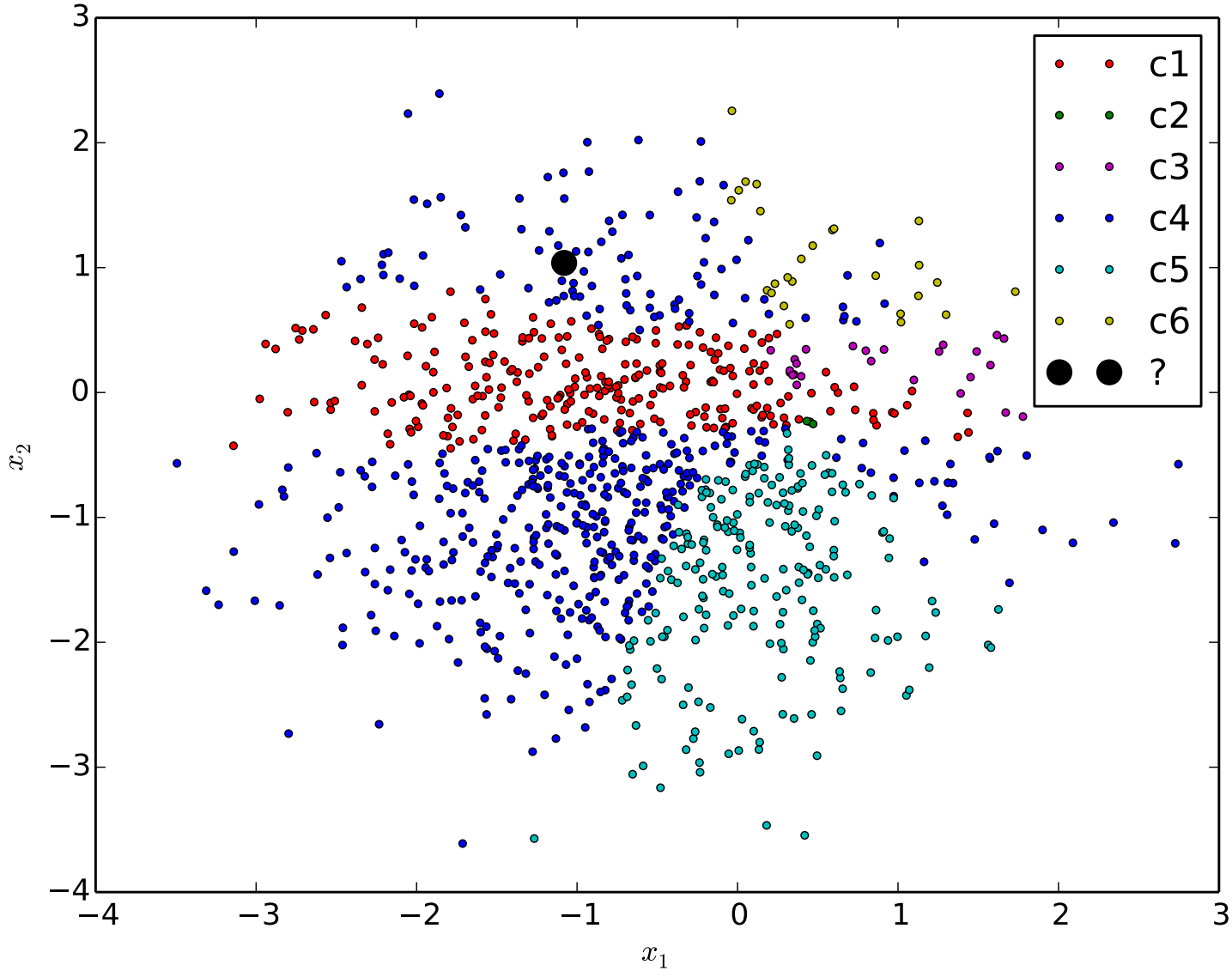


Figure: *Single*-label *k*NN

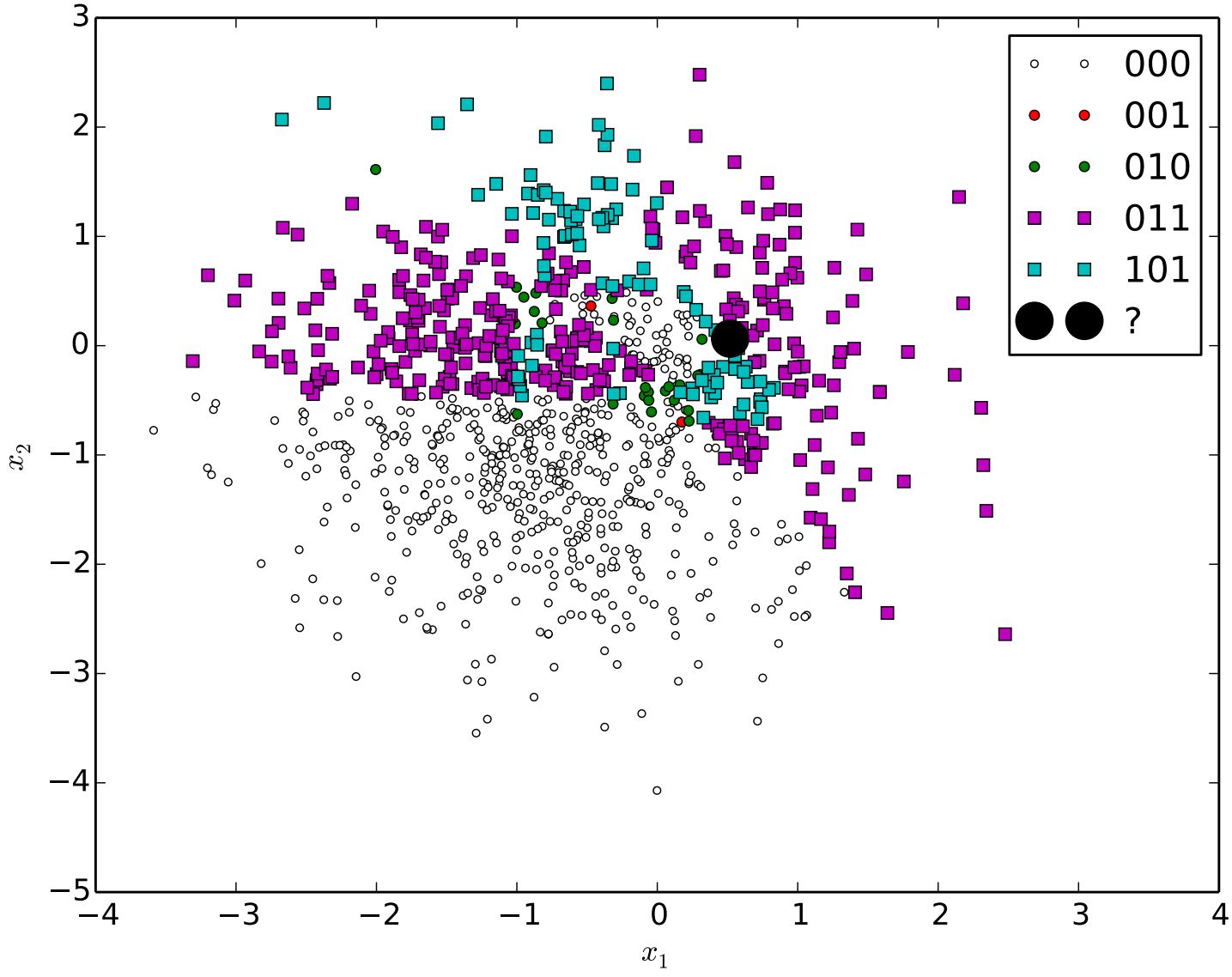


Figure: Multi-label kNN

Rules

For example,

$$\underbrace{X_4 > 1, X_2 \leq 0}_{\text{rule}} \mapsto \underbrace{[1, 0, 1, 0, 0]}_{\mathbf{y}}$$

Can expand rules with the Hoeffding bound: for a variable $x \in R$, the Hoeffding bound states that with probability $1 - \delta$, the true mean of X is at least $\bar{x} - \epsilon$, where

$$\epsilon = \sqrt{\frac{R^2 \ln\left(1/\delta\right)}{2t}}$$

over t examples.

Rules

For example,

$$\underbrace{X_4 > 1, X_2 \leq 0}_{\text{rule}} \mapsto \underbrace{[1, 0, 1, 0, 0]}_{\mathbf{y}}$$

Can expand rules with the Hoeffding bound: for a variable $x \in R$, the Hoeffding bound states that with probability $1 - \delta$, the true mean of X is at least $\bar{x} - \epsilon$, where

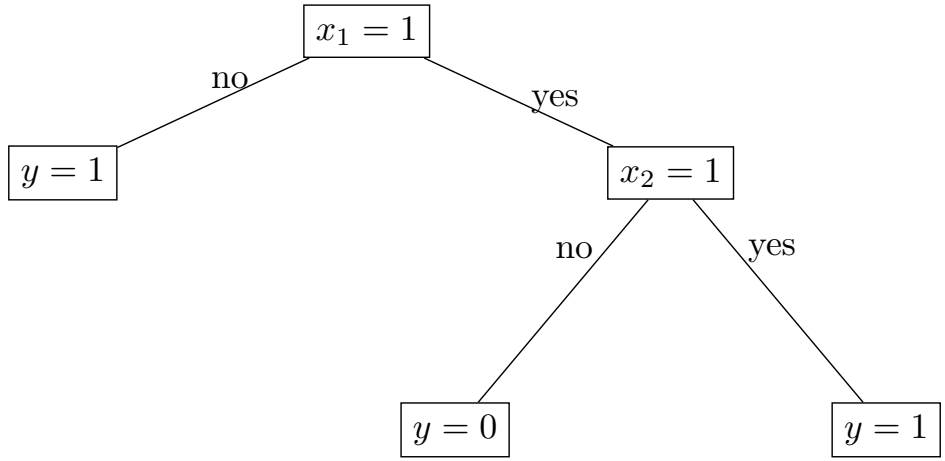
$$\epsilon = \sqrt{\frac{R^2 \ln\left(1/\delta\right)}{2t}}$$

over t examples. This gives us an indication of when to expand a rule.

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

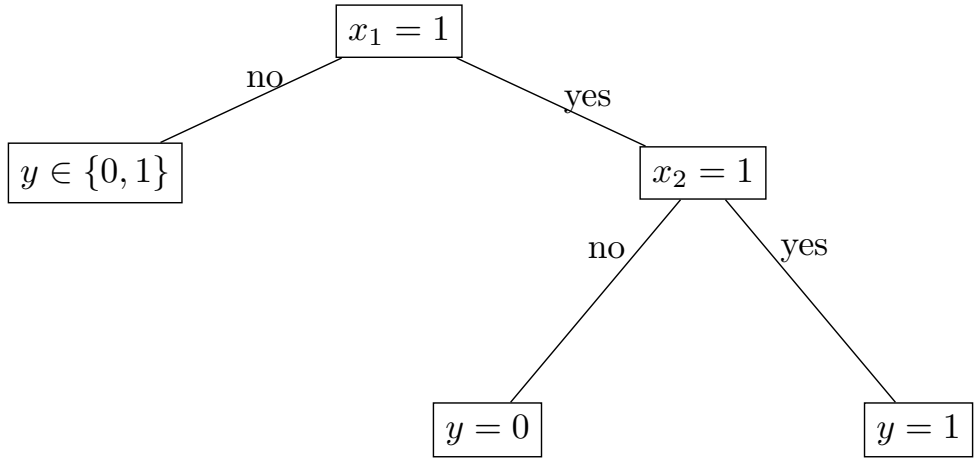
Decision Trees



Induce(S) returns Tree T:

1. if stop(S), return Leaf(S)
2. best $= \text{argmax}_{j \in \{1,\ldots,D\}} G(S, X_j)$
3. Create decision node that tests x_{best}
4. For $v \in \{0, 1\}$:
   - $S_v = \{(\mathbf{x}_{\neg\text{best}}, y) | x_{\text{best}} = v\}$
     - (Induce sub-datasets S_v based on $(x_{\text{best}} = v)$)
   - $T_v = \text{Induce}(S_v)$
   - attach T_v to the corresponding branch
5. return Tree (root x_{best}, branches on T_0, T_1)

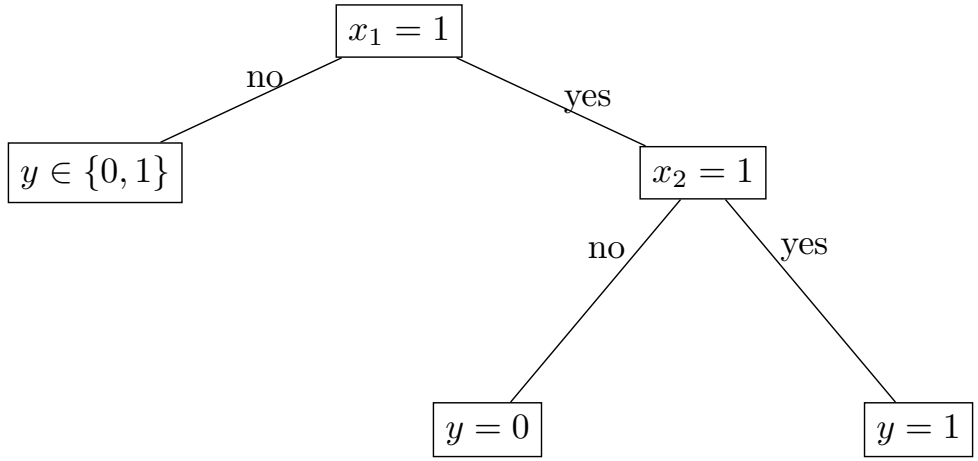
Decision Trees



Induce(S) returns Tree T:

1. if stop(S), return Leaf(S)
2. best $= \text{argmax}_{j \in \{1,\ldots,D\}} G(S, X_j)$
3. Create decision node that tests x_{best}
4. For $v \in \{0, 1\}$:
   - $S_v = \{(\mathbf{x}_{\neg\text{best}}, y) | x_{\text{best}} = v\}$
     - (Induce sub-datasets S_v based on $(x_{\text{best}} = v)$)
   - $T_v = \text{Induce}(S_v)$
   - attach T_v to the corresponding branch
5. return Tree (root x_{best}, branches on T_0, T_1)

Decision Trees



$$H(S) = -\sum_{c \in \{0,1\}} \Big(P(y = c) \log_2 P(y = c) \Big) \quad \bullet \text{ entropy}$$

$$G(S, X) = H(S) - \sum_{v \in \mathsf{vals}(X)} \frac{|S_v|}{|S|} \cdot H(S_v) \quad \bullet \text{ information gain}$$

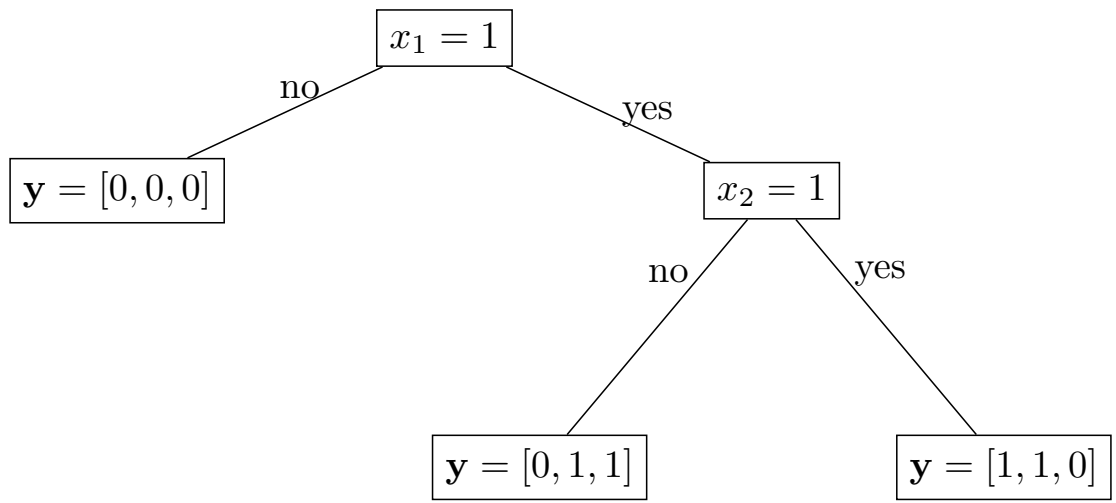
$$= H(S) - \sum_{v \in \mathsf{vals}(X)} \frac{|\{(\mathbf{x}, y) \in S | x = v\}|}{|S|} \cdot H(\{(\mathbf{x}, y) \in S | x = v\})$$

Multi-label Decision Trees

Using multi-label entropy,

$$H_{\mathsf{ML}}(S) = -\sum_{j=1}^{L} \sum_{c \in \{0,1\}} P(y_j = c) \log_2 P(y_j = c)$$

$$= -\sum_{j=1}^{L} P(y_j) \Big[\log_2 P(y_j) + [1 - P(y_j)] \log_2 [1 - P(y_j)] \Big]$$



Note that now, $(\mathbf{y}, \mathbf{x}) \in S$, i.e., multi-label vector assignments $\mathbf{y}$ are filtering down to the leaves.

Incremental Decision Trees

HOEFFDING TREE INDUCTION ALGORITHM:

1. Sort $(\mathbf{x}_t, y_t)$ into leaf ℓ
2. Update *sufficient statistics* (necessary to calculate $\bar{G}$) in ℓ
3. Increment n_ℓ (number of examples seen at ℓ)
4. If $n_\ell \mod n_{\min} = 0$:
   1. Compute $\bar{G}(X_j)$ for all attributes j
   2. X_a is the attribute with highest $\bar{G}$ and X_b second highest
   3. Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2t}}$
   4. if $\left[\bar{G}(X_a) - \bar{G}(X_b)\right] > \epsilon$, then (knowing that $\Delta G \geq \Delta\bar{G} - \epsilon > 0$ with probability $1 - \delta$):
      1. turn leaf ℓ into node splitting on X_a
      2. create leaves $\ell_1, \ldots, \ell_k$ for all values of $X_a \in \{v_1, \ldots, v_k\}$, with initialized sufficient statistics

Incremental Decision Trees

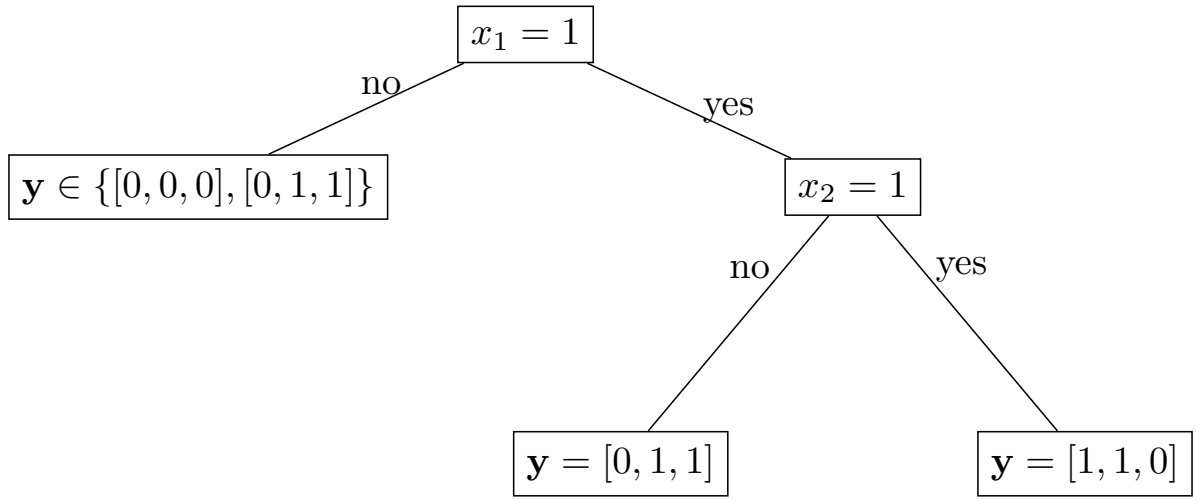
HOEFFDING TREE INDUCTION ALGORITHM:

1. Sort $(\mathbf{x}_t, y_t)$ into leaf ℓ
2. Update *sufficient statistics* (necessary to calculate $\bar{G}$) in ℓ
3. Increment n_ℓ (number of examples seen at ℓ)
4. If $n_\ell \mod n_{\min} = 0$:
   1. Compute $\bar{G}(X_j)$ for all attributes j
   2. X_a is the attribute with highest $\bar{G}$ and X_b second highest
   3. Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2t}}$
   4. if $\left[\bar{G}(X_a) - \bar{G}(X_b)\right] > \epsilon$, then (knowing that $\Delta G \geq \Delta\bar{G} - \epsilon > 0$ with probability $1 - \delta$):
      1. turn leaf ℓ into node splitting on X_a
      2. create leaves $\ell_1, \ldots, \ell_k$ for all values of $X_a \in \{v_1, \ldots, v_k\}$, with initialized sufficient statistics

With high probability, constructs an identical model that a traditional (greedy) method would learn.

ML Incremental Decision Trees

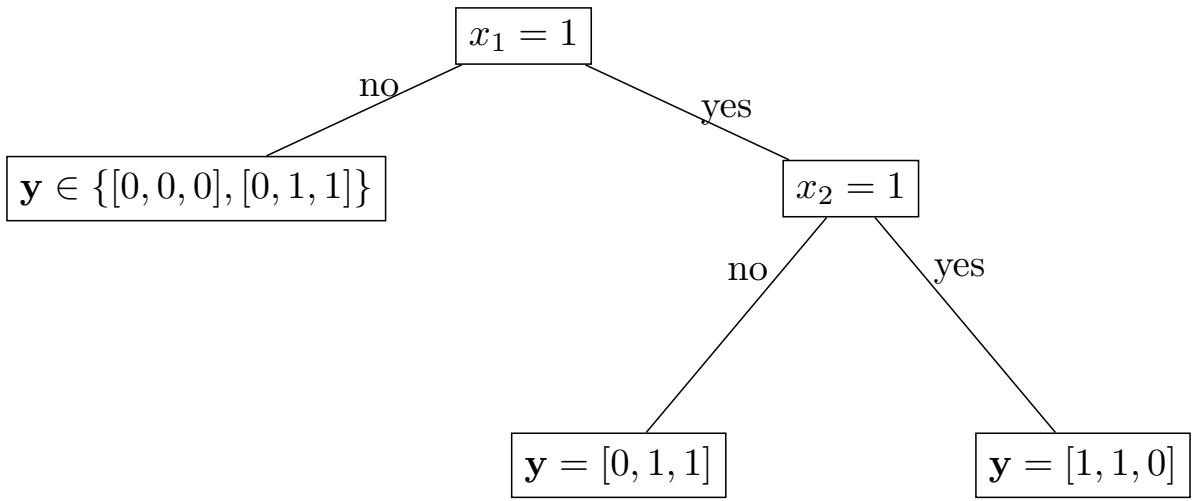
- Use Hoeffding tree algorithm with multi-label entropy
- Examples with *multiple labels* collect at the leaves (we can take the majority labelset, or merge . . .)



If $\{(\mathbf{x}_\ell, [1,1,0]), (\mathbf{x}_\ell, [1,0,0]), (\mathbf{x}_\ell, [0,1,0])\}$ at leaf ℓ, could return score $[0.67, 0.67, 0.0]$

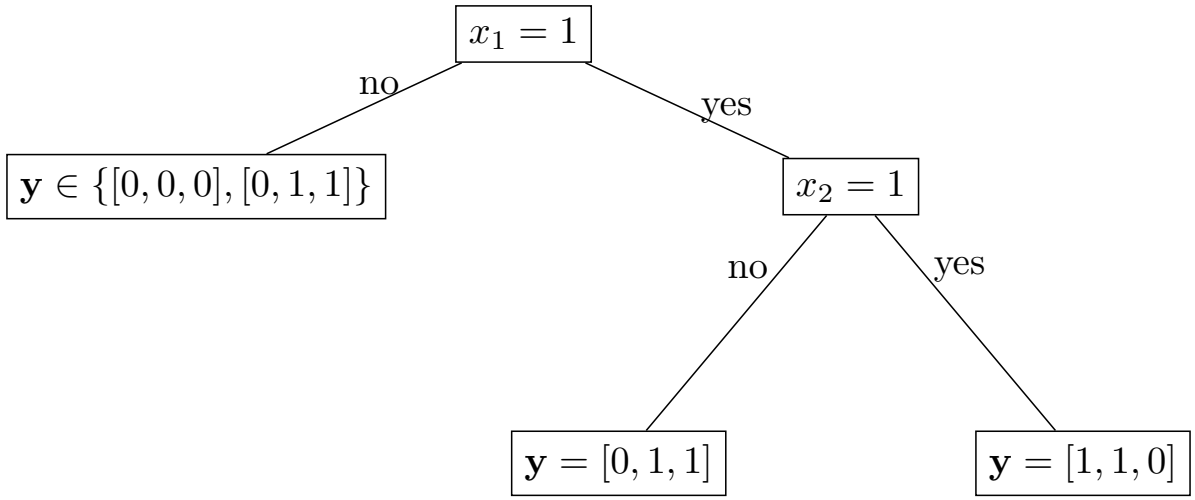
ML Incremental Decision Trees

- A multi-label *incremental* model
- Very *fast* (*single* model for all labels), and usually very *competitive*,



ML Incremental Decision Trees

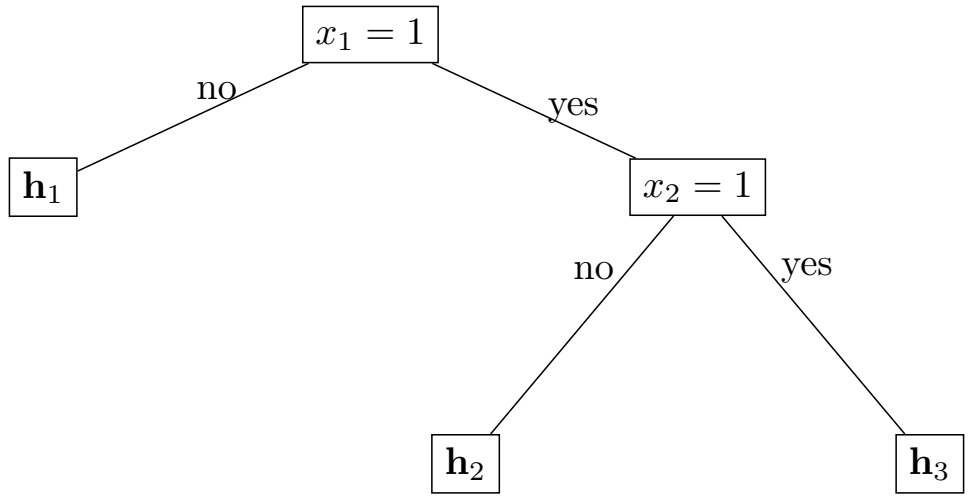
- A multi-label *incremental* model
- Very *fast* (*single* model for all labels), and usually very *competitive*,



- But Hoeffding bound is conservative, tree reluctant to grow, after many examples can remain as root node / stump / majority class classifier, ...
- And when it does grow, it may soon become outdated by concept drift

Multi-label Models at the Leaves

- Place multi-label classifiers at the leaves of the tree
- and use it in an ensemble.



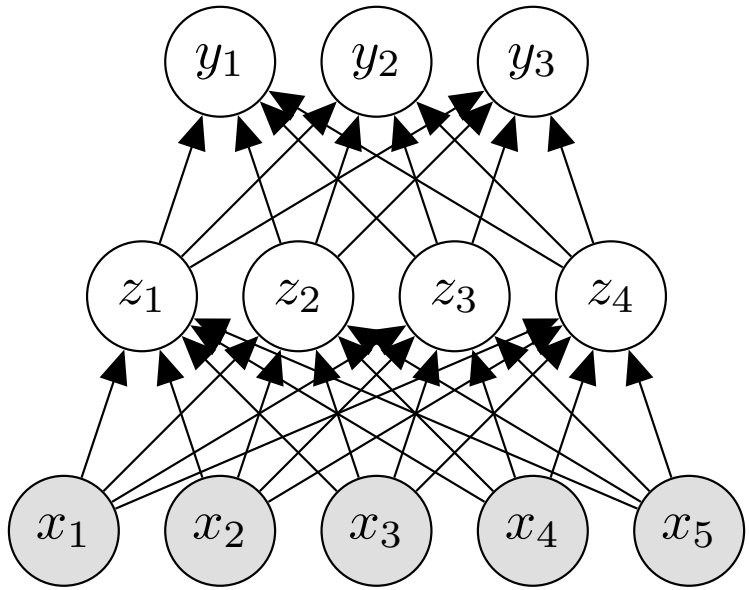
where each $\mathbf{h}_\ell : \mathcal{X}^{D^\ell} \to \mathcal{Y}^{L^\ell}$, i.e., a multi-label classifier dealing with a subset of the input space and a subset of the label space at each leaf ℓ (note: $D^\ell \leq D$, $L^\ell \leq L$).

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

Multiple-layer Models

We can instead use nodes $\mathbf{z} = [z_1, \ldots, z_k]$, derived from either the label space or input space (or both), to obtain ...

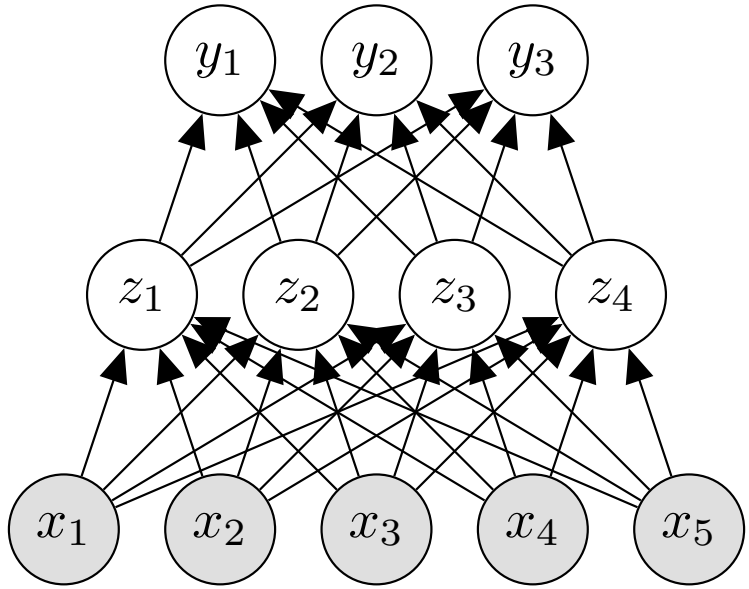


such that we aim to obtain

$$p(y_j|\mathbf{z}) \approx p(y_j|\mathbf{z}, y_k)$$

Multi-layer Perceptrons

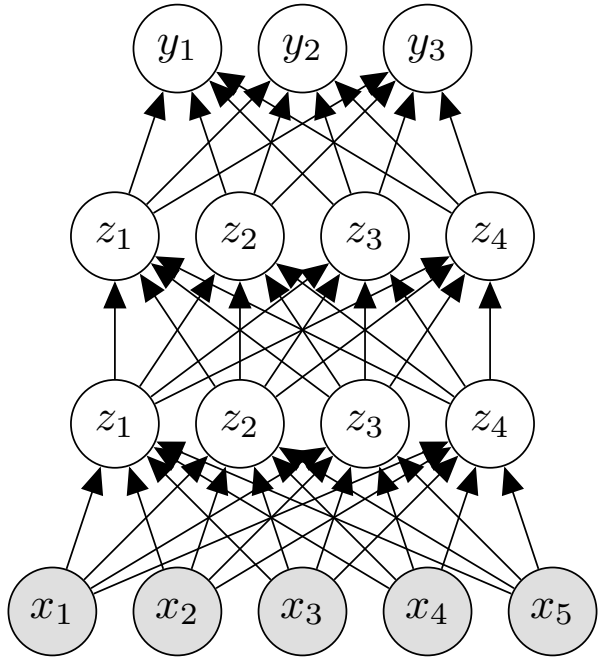
- Just include an output node for each label.
- train with, e.g., gradient descent + error back-propagation



- Native application (no 'transformation')
- Using stochastic gradient descent (SGD) – incremental

Deep Learning

- Independence is removed by multiple layers



- . . . but how many nodes, layers (deep learning), learning rate, etc? (hyper-parameter tuning can be tedious!)
- Empirical data-stream comparisons show that SGD/MLPs struggle against HTs (in predictive performance).

Removing Dependence

We can just run PCA on the label space, make labels independent!

$\mathbf{Z} = \mathbf{W}\mathbf{Y}$ • PCA, $\mathbf{W}$ has top k components, $k < L$

$\mathbf{h} : \mathcal{X} \to \mathcal{Z}$ • model, trained on examples $\{\mathbf{x}_i, \mathbf{z}_i\}_{i=1}^N$

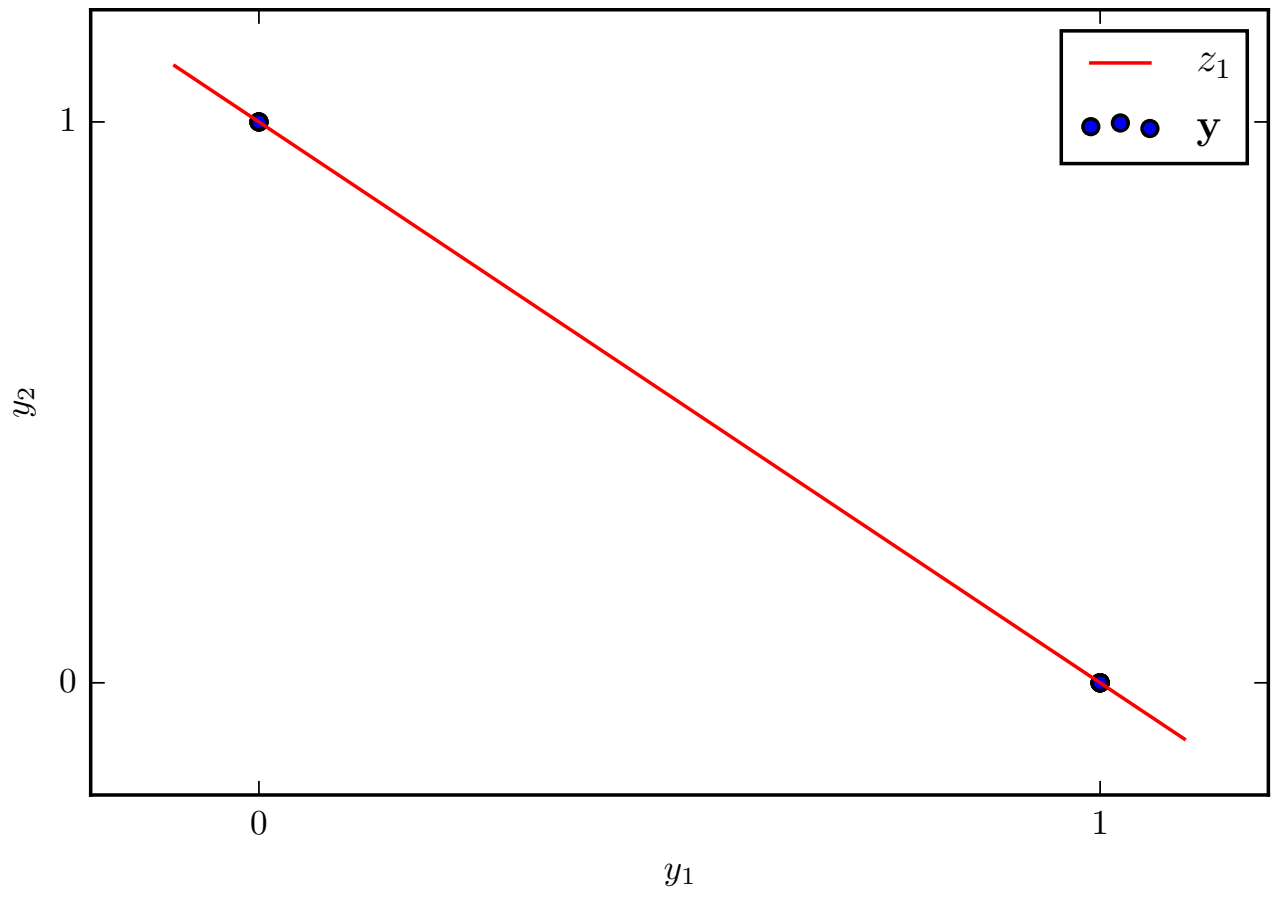
and then, with input $\mathbf{x}$,

$\hat{\mathbf{z}} = \mathbf{h}(\mathbf{x})$ • prediction

$\hat{\mathbf{y}} = \mathbf{W}^{-1}\hat{\mathbf{z}}$ • transform

PCA on the Label Space

Example, where $L = 2, \ldots$

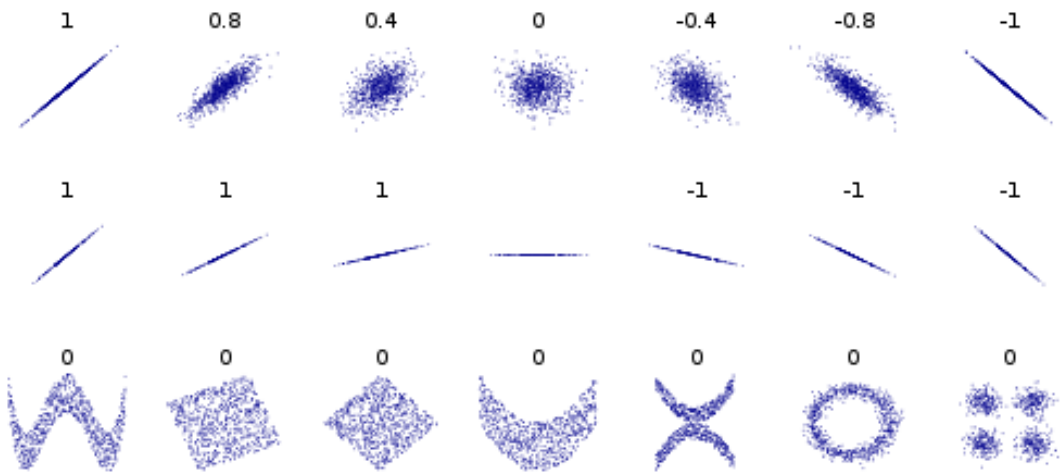


PCA on the Label Space

- Removes correlation, relatively scalable,
- compresses label space (faster learning for BR, etc.)

but . . .

- only linear correlation
- doesn't take into account input space



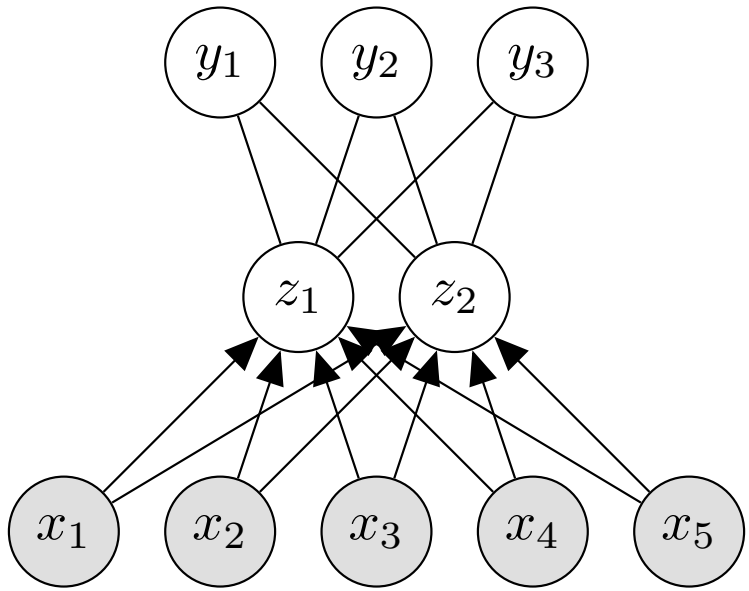
source: Wikipedia

- can use kernels (kernel PCA . . .),
- can use Canonical Correlation Analysis (CCA)

An Independent Label Space

General approach:

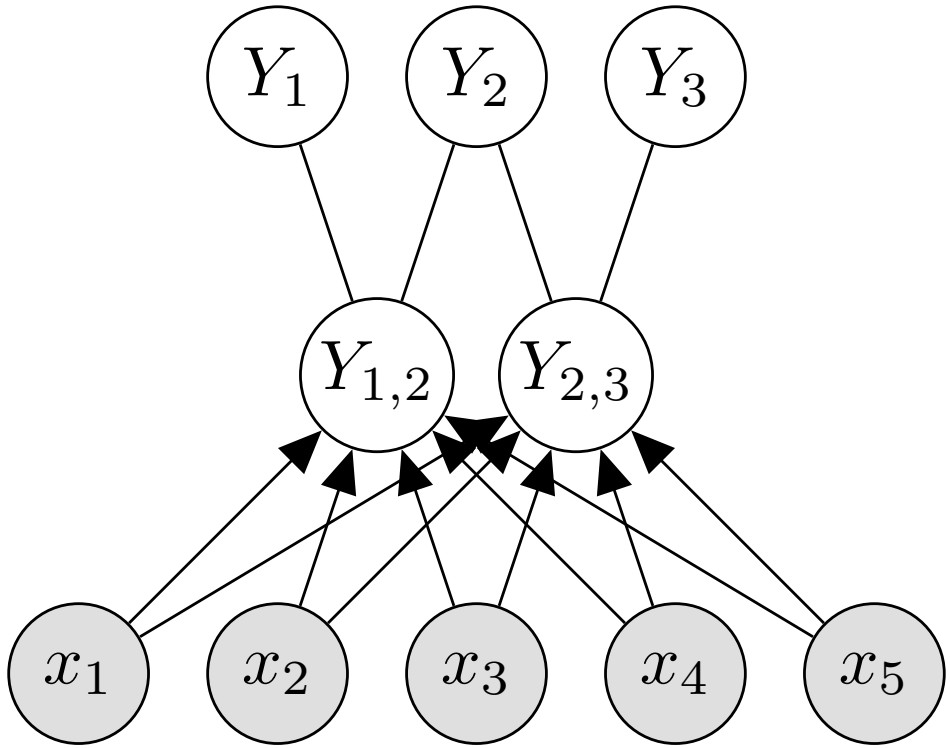
$$\mathbf{y} = f^{-1}(\mathbf{z})$$
$$\mathbf{z} = f(\mathbf{y})$$



- Apply binary relevance, on fewer and independent labels!
- Many methods exist (PCA, CCA, factor analysis, deep learning, MLPs, cluster analysis, ...)

Revisiting Meta Labels

Meta labels form an inner layer, $k = 2$ meta labels . . . :



where, e.g., $Y_{1,2} \in \{[0,0],[1,0]\}$ and $Y_{2,3} \in \{[1,0],[0,1]\}$.

- We can achieve $k < L$, and $k < D$,
- without any iterative learning!

Voting Using Meta Labels

From meta-labels to label confidence, e.g., for $Y'_{1,2} \in \{[1,0],[0,1]\}$

$\mathbf{y}_{1,2}$	P_1	P_2	$p(Y'_{1,2} = \mathbf{y}_{1,2}\|\tilde{\mathbf{x}})$
$[1,0]$	0.1	0	0.0
$[0,1]$	0	1	0.9
$P_{1,2}$	0.1	0.9	

From (meta-)label confidence to label votes

	P_1	P_2	P_3
$P_{1,2}$	0.1	0.9	
$P_{2,3}$		0.7	0.3
$\sum$	0.1	0.85	0.3
$\hat{\mathbf{y}}$ (> 0.5)	0	1	0

Outline

1. Introduction
2. Applications
3. Multi-label Classification
   - Connected Labels
   - Meta Labels
4. Label Dependence
5. Big Multi-Labelled Data Streams
6. Adaptation to data streams
7. Multi-label Hoeffding Trees and Adaptations
8. Inner Layer Models
9. Summary

Summary

Multi-label/multi-output classification

- Many real-world applications
- Related to many areas of machine learning
- Model labels together (for predictive and time performance) because of label dependence
- Label dependence is fundamental, but should be understood correctly
- 'Problem transformation' methods: connected labels and meta labels
- 'Algorithm adaptation' methods: k-nearest-neighbours, incremental decision trees, inner-layer models (neural-networks, etc.)