Classification

Jesse Read

[D&K] IoT Stream Data Mining 2017-2018

December 13, 2017

Outline

Classification

$\mathbf{x} =$ 

Binary classification

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

Classification

$\mathbf{x} =$ 

Multi-class classification

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

Classification

$\mathbf{x} =$ 

Multi-label classification

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

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

Single-label vs. Multi-label

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

[†] aka multi-objective, multi-target, multi-dimensional, multi-label.

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

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	?	?	?	?

Notation

Given a set of L labels, e.g.,

$$\mathcal{L} = \{\texttt{sunset}, \texttt{people}, \texttt{foliage}, \texttt{beach}, \texttt{urban}, \texttt{field}\}$$

($|\mathcal{L}| = L = 6$ in this example) we wish to obtain a model h, that can take some input,

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

and produce predictions,

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

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

Outline

Text Categorization and Tag Recommendation

For example, the IMDb dataset: Textual movie plot summaries associated with genres (labels). Also: Bookmarks, Bibtex, del.icio.us datasets.

Text Categorization and Tag Recommendation

For example, the IMDb dataset: Textual movie plot summaries associated with genres (labels). Also: Bookmarks, Bibtex, del.icio.us datasets.

	abandoned	*accident*	...	*violent*	*wedding*	horror	romance	...	comedy	action
i	X_1	X_2	...	X_{1000}	X_{1001}	Y_1	Y_2	...	Y_{27}	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

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

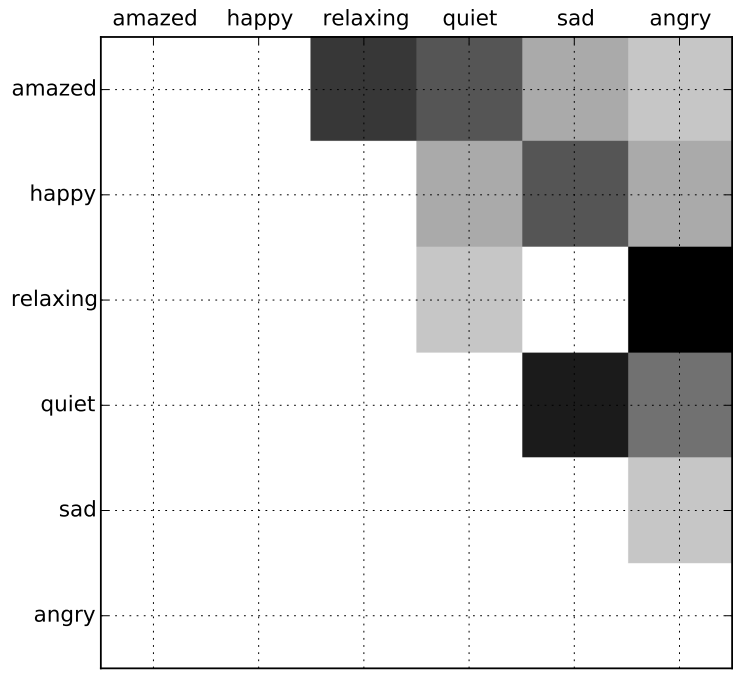
e.g., Associating Scenes with concepts

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

Labelling Audio

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

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



Medicine

Medical Diagnosis

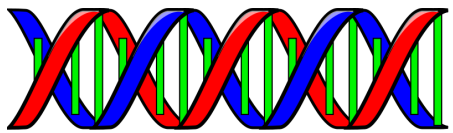
- medical history, symptoms $\rightarrow$ diseases / ailments

The 'Medical' dataset (a benchmark multi-label dataset),

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

Bioinformatics

A gene or a protein may be associated with biological functions.



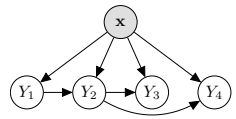
For example, the well-known Yeast dataset: $2,417$ genes, described by 103 attributes, labeled into 14 groups of the FunCat functional catalogue.
Useful applications such as anti-biotics[1]:

> *Activity prediction of AMPs from their amino acid sequences is of great therapeutic importance but imposes challenges on prediction methods due to label interactions. [. . .] we propose a novel multi-label learning model to address this problem.*

[1] e.g., Wang et al. *Multi-label Learning for Predicting the Activities of Antimicrobial Peptides*. Nature 2017

Multi-label Classification in Artificial Intelligence

- For knowledge representation (multi-label methods are often based on probabilistic graphical models)



- For decision making: multiple agents, agents taking multiple actions
- Techniques can be applied (almost) as-is to sequential data
- A stepping stone to structured output prediction,

$$h : \mathcal{X} \mapsto \mathcal{Y}$$

which encompasses tasks mapping (multiple) perception to (multiple) actions.

Localization and Tracking

We can consider a pixel/tile as a label, where $y_j = 1$ if the j-th tile contains an object, otherwise $y_j = 0$. The instance $\mathbf{x}$ contains some observation, e.g., sensor readings,

Figure: A room with a single light source and a number of light sensors.

Given some training data, we train a multi-label model for this problem

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

(some domain assumptions may help).

Localization and Tracking

We can consider a pixel/tile as a label, where $y_j = 1$ if the j-th tile contains an object, otherwise $y_j = 0$. The instance $\mathbf{x}$ contains some observation, e.g., sensor readings,

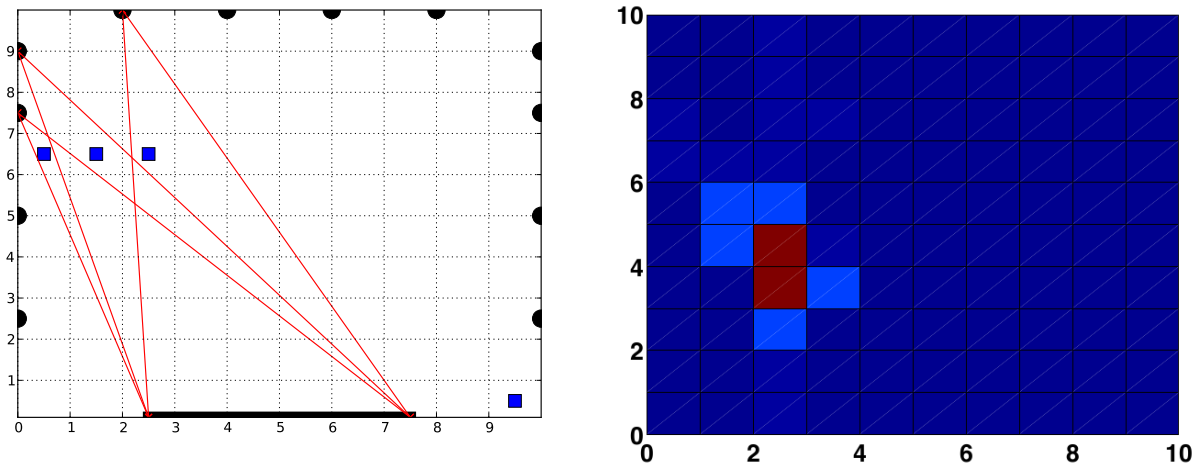


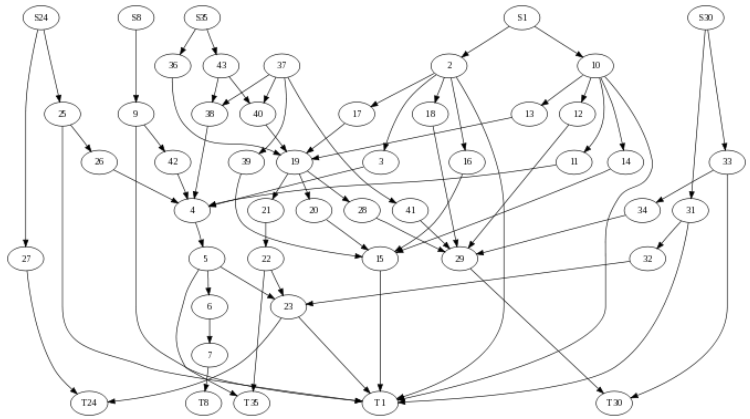
Figure: A room with a single light source and a number of light sensors.

Given some training data, we train a multi-label model for this problem

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

(some domain assumptions may help).

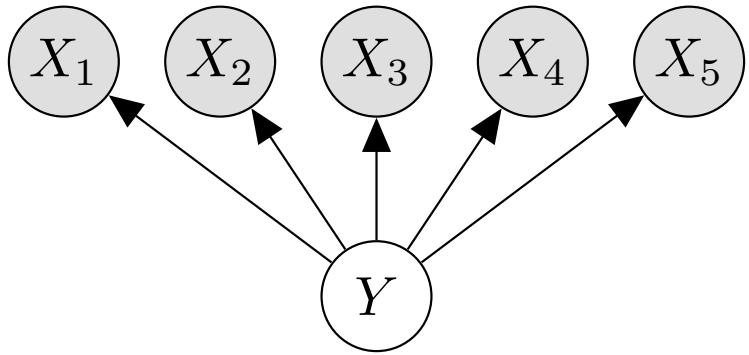
Internet of Things (IoT)



A network of interconnected *things* (software, actuators, sensors, vehicles, . . .). Many nodes require a multi-input multi-output decision-making framework.

Outline

(Single-label) Naive Bayes



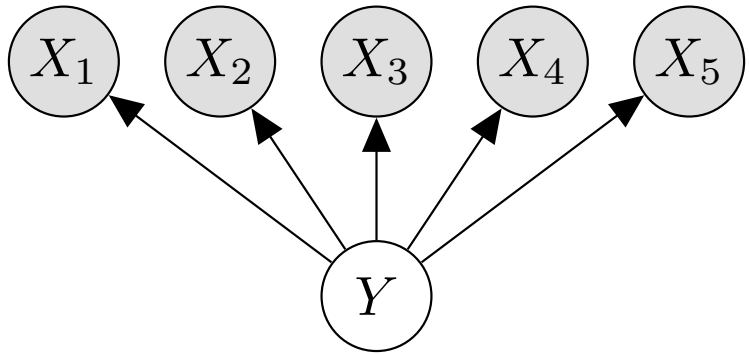
$$P(Y|\mathbf{X}) = \frac{P(Y)P(\mathbf{X}|Y)}{P(\mathbf{X})} \quad \triangleright \text{ Bayes rule}$$

$$P(\mathbf{X}, Y) = P(Y)P(\mathbf{X}|Y) \quad \triangleright \text{ factorize from PGM}$$

where likelihood (under independence assumption)

$$P(\mathbf{X}|Y) = \prod_{d=1}^{D} P(X_d|Y)$$

(Single-label) Naive Bayes

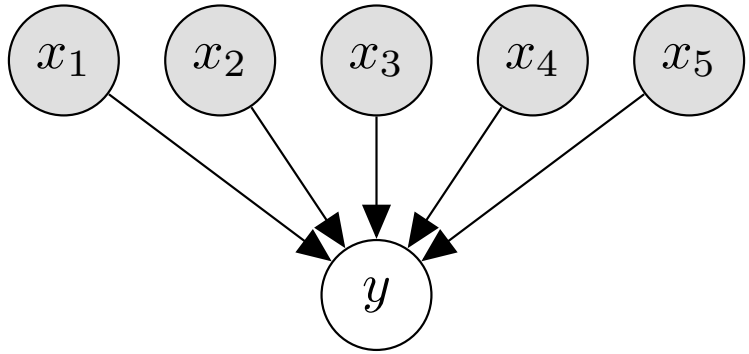


$$\mathsf{h}(\mathbf{x}) = \operatorname*{argmax}_{y \in \{0,1\}} P(y) P(\mathbf{x}|y) \quad \triangleright \text{ decision rule}$$

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

i.e., our **classifier**, with with we may make classifications $\hat{y} = h(\mathbf{x})$.

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

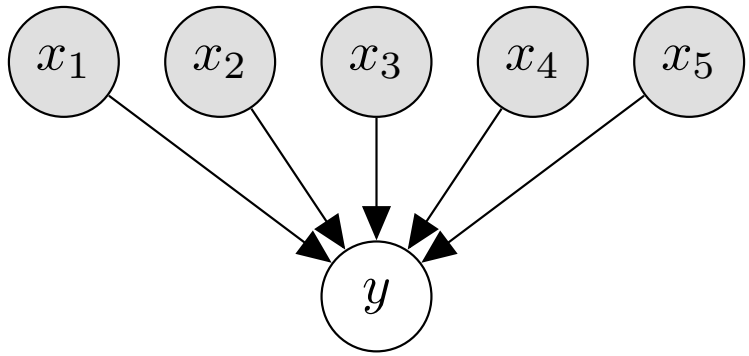
$$E(\boldsymbol{\theta}) = -\log \prod_{i=1}^{N} \left[(p_i)^{y_i} \cdot (1 - p_i)^{1-y_i} \right] = \sum_{i=1}^{N} E_i(\boldsymbol{\theta})$$

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



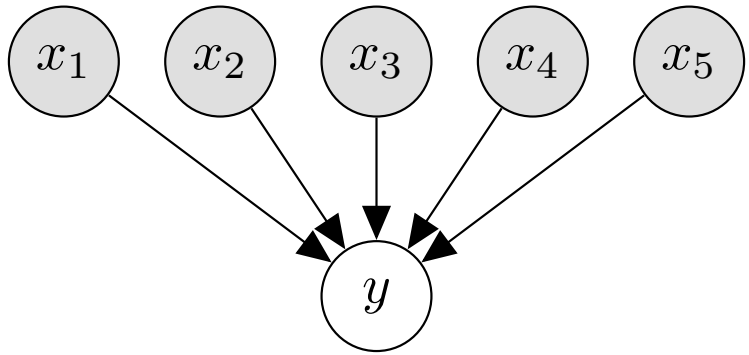
$$\hat{y} = h(\mathbf{x}) = \underset{y \in \{0,1\}}{\text{argmax}}\, p(y|\mathbf{x})$$

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

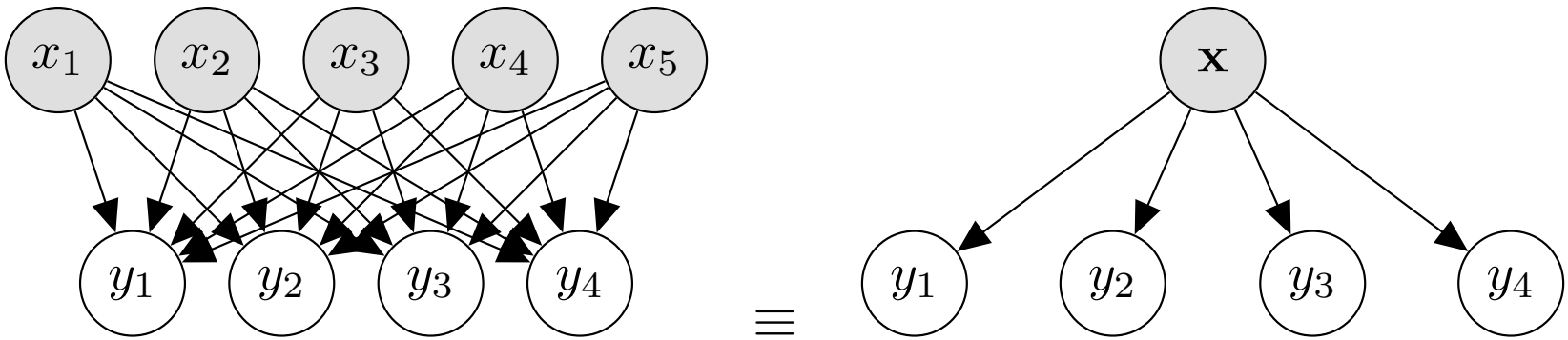
$$p(y = 0|\mathbf{x}) = 1 - p(\mathbf{x}; \boldsymbol{\theta})$$

A Note on Graphical Notation

We move from single label representation...

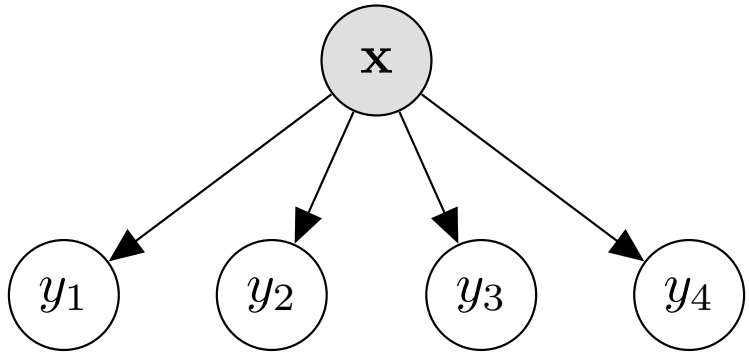


to multi-label representation...



(the shaded nodes represent the observed instance).

The Binary Relevance Approach



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

and then,

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

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

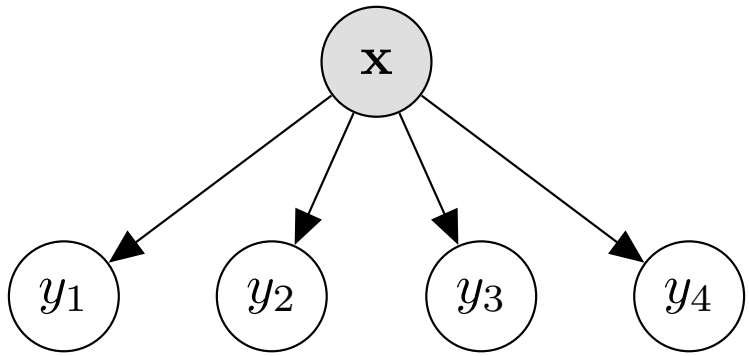
$\mathbf{X}$	Y_3
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	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:

$$h_j : \mathcal{X} \to \{0, 1\}$$

Why Not Binary Relevance?

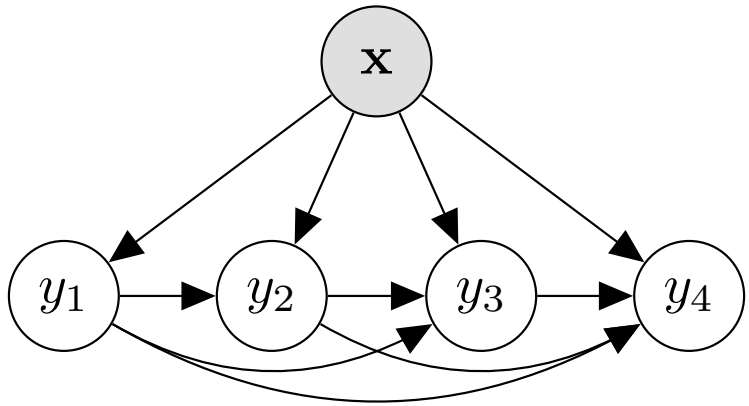


The independence assumption is not a good representation of the knowledge we have of the world;

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

(in most cases).

A Probabilistic Cascade[2]



Given a query instance $\mathbf{x}$,

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

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

[2] In the literature, known as *Probabilistic Classifier Chains*

Multi-label Learning

From the formulation

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

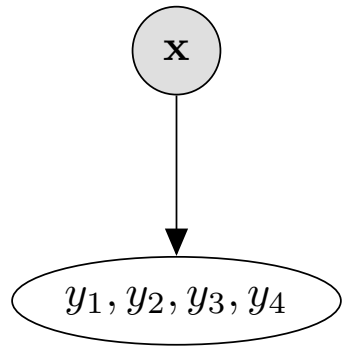
we get our classifier h, such that $\hat{\mathbf{y}} = h(\mathbf{x})$. But where does P come from? As in single-label learning, P must be *estimated*/learned from training data. We may either

- learn $P(\mathbf{y}|\mathbf{x})$ directly (e.g., label-powerset approach), or
- learn the $P(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})$ individually (binary relevance approach)

How to do this learning? It depends what loss function we are interested in.

Each Label Combination as a Class Value

We may formulate one multi-class problem (taking many values),



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

where the space $\mathcal{Y}$ contains all possible combinations:

$$\mathcal{Y} = \{[0,0,0,0],[0,0,0,1],\ldots,[1,1,1,1]\}$$

i.e., the powerset[3] of $\mathcal{L} = \{1,\ldots,L\}$; of size $|\mathcal{Y}| = \mathcal{P}(\mathcal{L}) = 2^{|\mathcal{L}|}$. Although, typically, $\mathcal{Y} = \{\mathbf{y}^{(i)},\ldots,\mathbf{y}^{(N)}\}$, i.e., the distinct combinations in the training set; thus $|\mathcal{Y}| \leq \min(N, 2^L)$.

[3] Known in the literature as the label powerset method (LP)

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

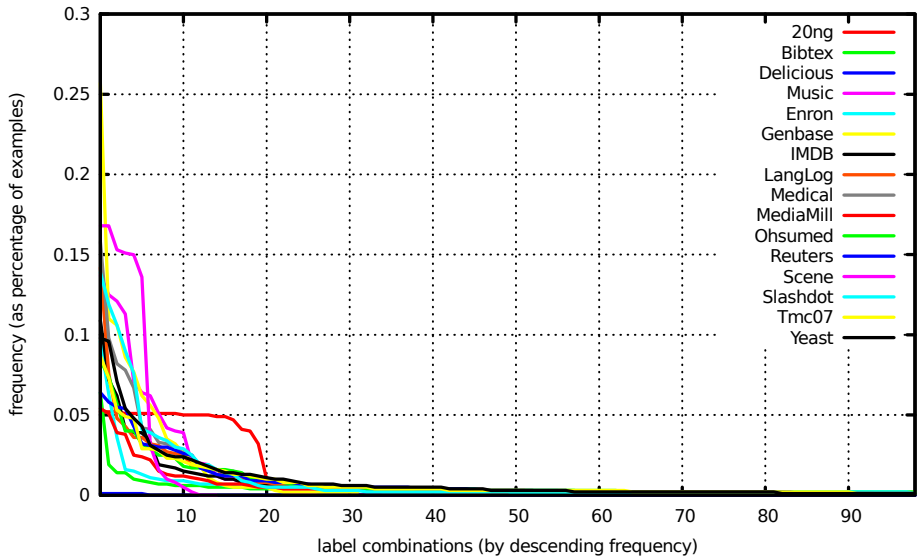
. . . into a multi-*class* dataset:

$\mathbf{X}$	$Y \in 2^L$
$\mathbf{x}^{(1)}$	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 the Label-powerset Method

- Complexity: (up to 2^L combinations)
- Imbalance: few examples per class label
- Overfitting: how to predict new value?



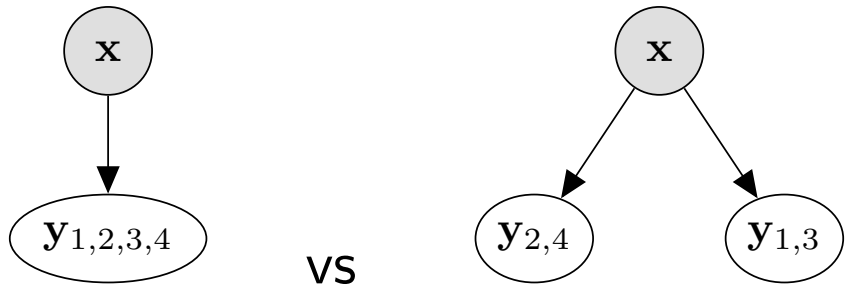
Note: horizontal axis has been truncated

Solutions involve:

- Pruning
- Dealing with subsets

Meta Labels

Decompose the labelset into M sets of k labels.

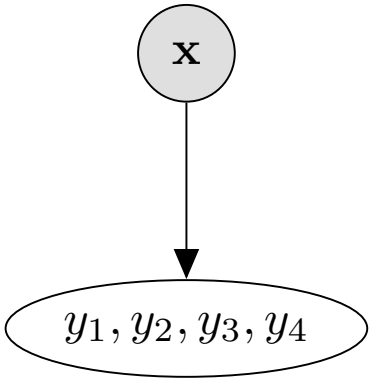


	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$	$\hat{y}_4$
$h^1(\mathbf{x})$		0		0
$h^2(\mathbf{x})$	1		1	
$\hat{\mathbf{y}}$	1	0	1	0

- Apply LP to the label subsets
- Combine later via voting
- Complexity reduction from $O(2^L)$ to $O(M \cdot 2^k)$, predictive performance usually improves too.

Same Goal, Different Approaches

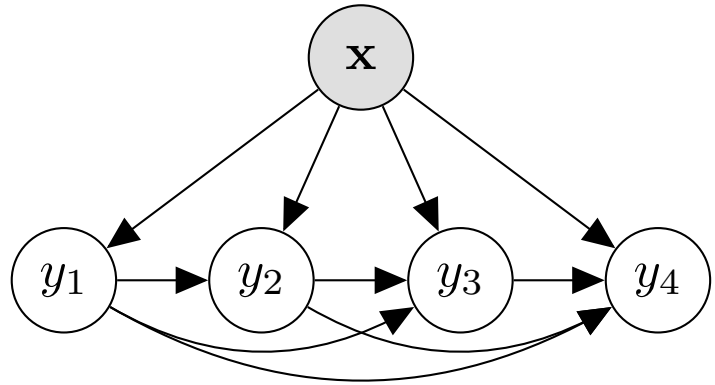
Meta Labels



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

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

Classifier Chains



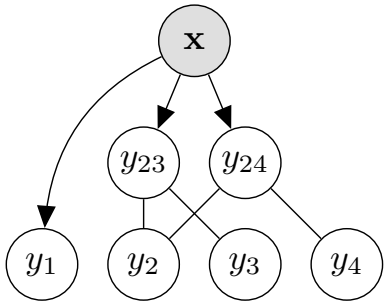
$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x})$$

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

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

Same Goal, Different Approaches

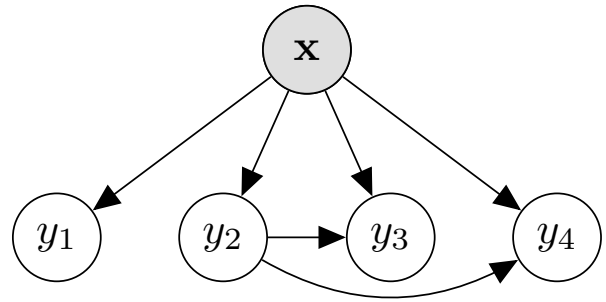
Meta Labels



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

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

Classifier Chains



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

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

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

Both methods: Model label dependence with a minimum of structure, i.e., predictive performance vs computational complexity.

Outline

Multi-label Evaluation

For an instance $\mathbf{x}$, our model provides predictions

$$[\hat{y}_1, \ldots, \hat{y}_L] = \hat{\mathbf{y}} = h(\mathbf{x})$$

$\mathbf{x}^{(i)}$	$\mathbf{y}^{(i)}$	$\hat{\mathbf{y}}^{(i)}$
$\mathbf{x}^{(1)}$	[1 0 1 0]	[1 0 0 1]
$\mathbf{x}^{(2)}$	[0 1 0 1]	[0 1 0 1]
$\mathbf{x}^{(3)}$	[1 0 0 1]	[1 0 0 1]
$\mathbf{x}^{(4)}$	[0 1 1 0]	[0 1 0 0]
$\mathbf{x}^{(5)}$	[1 0 0 0]	[1 0 0 1]

How good is model h?

Multi-label Evaluation Metrics

	$\mathbf{y}^{(i)}$	$\hat{\mathbf{y}}^{(i)}$
$\mathbf{x}^{(1)}$	[1 0 1 0]	[1 0 0 1]
$\mathbf{x}^{(2)}$	[0 1 0 1]	[0 1 0 1]
$\mathbf{x}^{(3)}$	[1 0 0 1]	[1 0 0 1]
$\mathbf{x}^{(4)}$	[0 1 1 0]	[0 1 0 0]
$\mathbf{x}^{(5)}$	[1 0 0 0]	[1 0 0 1]

Hamming loss

$$L_{\text{Hamming}}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{N \cdot L} \sum_{i=1}^{N} \sum_{i=1}^{L} \mathbb{I}[\hat{y}_j^{(i)} \neq y_j^{(i)}]$$

$$= 0.20$$

No consideration of label dependence; easy to get low loss on sparse labels.

Multi-label Evaluation Metrics

	$\mathbf{y}^{(i)}$	$\hat{\mathbf{y}}^{(i)}$
$\mathbf{x}^{(1)}$	[1 0 1 0]	[1 0 0 1]
$\mathbf{x}^{(2)}$	[0 1 0 1]	[0 1 0 1]
$\mathbf{x}^{(3)}$	[1 0 0 1]	[1 0 0 1]
$\mathbf{x}^{(4)}$	[0 1 1 0]	[0 1 0 0]
$\mathbf{x}^{(5)}$	[1 0 0 0]	[1 0 0 1]

0/1 loss

$$L_{0/1}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\hat{\mathbf{y}}^{(i)} \neq \mathbf{y}^{(i)})$$

$$= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\Big(\underbrace{\sum_{i=1}^{L} \mathbb{I}[\hat{y}_j^{(i)} \neq y_j^{(i)}]}_{\text{Hamming Loss}} = 0\Big)$$

$$= 0.60$$

If a single label relevance incorrect, loss of 1 for that instance.

Multi-label Evaluation Metrics

	$\mathbf{y}^{(i)}$	$\hat{\mathbf{y}}^{(i)}$
$\mathbf{x}^{(1)}$	[1 0 1 0]	[1 0 0 1]
$\mathbf{x}^{(2)}$	[0 1 0 1]	[0 1 0 1]
$\mathbf{x}^{(3)}$	[1 0 0 1]	[1 0 0 1]
$\mathbf{x}^{(4)}$	[0 1 1 0]	[0 1 0 0]
$\mathbf{x}^{(5)}$	[1 0 0 0]	[1 0 0 1]

Jaccard Index

$$
\begin{aligned}
J(\mathbf{Y}, \hat{\mathbf{Y}}) &= \frac{1}{N} \sum_{i=1}^{N} \frac{|\hat{\mathbf{y}}^{(i)} \wedge \mathbf{y}^{(i)}|}{|\hat{\mathbf{y}}^{(i)} \vee \mathbf{y}^{(i)}|} \\
&= \frac{1}{5}(\frac{1}{3} + 1 + 1 + \frac{1}{2} + \frac{1}{2}) \\
&= 0.67
\end{aligned}
$$

(Where $\vee$ and $\wedge$ are the logical OR and AND operations, applied vector-wise)

Multi-label Evaluation Metrics

We can evaluate posterior probabilities/confidences directly.

	$\mathbf{y}^{(i)}$	$[P(y_j\|\mathbf{x}^{(i)}])]_{j=1}^{L}$
$\mathbf{x}^{(1)}$	[1 0 1 0]	[0.9 0.0 0.4 0.6]
$\mathbf{x}^{(2)}$	[0 1 0 1]	[0.1 0.8 0.0 0.8]
$\mathbf{x}^{(3)}$	[1 0 0 1]	[0.8 0.0 0.1 0.7]
$\mathbf{x}^{(4)}$	[0 1 1 0]	[0.1 0.7 0.1 0.2]
$\mathbf{x}^{(5)}$	[1 0 0 0]	[1.0 0.0 0.0 1.0]

Log loss

- $y_3 = 1$, $h_3(\mathbf{x}) = 0.4$ incurs loss of $-\log(0.4) = 0.92$
- $y_3 = 1$, $h_3(\mathbf{x}) = 0.1$ incurs loss of $-\log(0.1) = 2.30$

aka cross entropy for multiple classes, $y \in \{1, \ldots, K\}$.

Multi-label Evaluation Metrics

	$\mathbf{y}^{(i)}$	$[P(y_j\|\mathbf{x}^{(i)}])]_{j=1}^{L}$
$\mathbf{x}^{(1)}$	[1 0 1 0]	[0.9 0.0 0.4 0.6]
$\mathbf{x}^{(2)}$	[0 1 0 1]	[0.1 0.8 0.0 0.8]
$\mathbf{x}^{(3)}$	[1 0 0 1]	[0.8 0.0 0.1 0.7]
$\mathbf{x}^{(4)}$	[0 1 1 0]	[0.1 0.7 0.1 0.2]
$\mathbf{x}^{(5)}$	[1 0 0 0]	[1.0 0.0 0.0 1.0]

Ranking loss – to encourage good ranking;
evaluates the average fraction of label pairs miss-ordered for $\mathbf{x}$:

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{(j,k):y_j>y_k}\left(\mathbb{I}[r_i(j) < r_i(k)] + \frac{1}{2}\mathbb{I}[r_i(j) = r_i(k)]\right)$$

where $r_i(j) :=$ ranking of label j for instance $\mathbf{x}^{(i)}$

Multi-label Evaluation Metrics

	$\mathbf{y}^{(i)}$	$[P(y_j\|\mathbf{x}^{(i)}])]_{j=1}^{L}$
$\mathbf{x}^{(1)}$	[1 0 1 0]	[0.9 0.0 0.4 0.6]
$\mathbf{x}^{(2)}$	[0 1 0 1]	[0.1 0.8 0.0 0.8]
$\mathbf{x}^{(3)}$	[1 0 0 1]	[0.8 0.0 0.1 0.7]
$\mathbf{x}^{(4)}$	[0 1 1 0]	[0.1 0.7 0.1 0.2]
$\mathbf{x}^{(5)}$	[1 0 0 0]	[1.0 0.0 0.0 1.0]

Ranking loss – to encourage good ranking;
evaluates the average fraction of label pairs miss-ordered for $\mathbf{x}$:

$$\frac{1}{5}(\frac{1}{4}+\frac{0}{4}+\frac{0}{4}+\frac{1.5}{4}+\frac{1}{4})$$

Multi-label Evaluation Metrics

Other metrics include

- ONE ERROR – if top ranked label is not in set of true labels
- COVERAGE – average "depth" to cover all true labels
- PRECISION
- RECALL
- macro-averaged F-MEASURE (ordinary averaging of a binary measure)
- micro-averaged F-MEASURE (labels as different instances of a 'global' label)
- PRECISION vs. RECALL curves

As generally, a metric should be considered based on the application. We cannot optimize all losses at once!

Minimizer for 0/1 Loss

$$L_{0/1} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{\mathbf{y}}^{(i)} \neq \mathbf{y}^{(i)})$$

We write out the expected loss, and try to minimize it. The single-label case:

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}}[\ell(\hat{y})] &= \sum_{k \in \{0,1\}} \ell_{\mathbf{x}}(y = k, \hat{y}) \cdot P(y = k|\mathbf{x}) \\
&= \sum_{k \in \{0,1\}} (1 - \delta_{y\hat{y}}) \cdot P(y = k|\mathbf{x}) \quad \triangleright \text{ plug in } \ell(\cdot, \cdot) \\
&= 1 - P(y|\mathbf{x}) \\
\operatorname*{argmin}_{\hat{y} \in \{0,1\}} \mathbb{E}_{\mathbf{x}}[\ell(\hat{y})] &= \operatorname*{argmin}_{\hat{y} \in \{0,1\}} \Big[1 - P(y|\mathbf{x})\Big] \quad \triangleright \text{ want to minimize this} \\
\hat{y} &= \operatorname*{argmax}_{y \in \{0,1\}} P(y|\mathbf{x}) \quad \triangleright \ldots, \text{ i.e., MAP estimate!}
\end{aligned}$$

Minimizer for 0/1 Loss

$$L_{0/1} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{\mathbf{y}}^{(i)} \neq \mathbf{y}^{(i)})$$

We write out the expected loss, and try to minimize it. The single-label case:

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}}[\ell(\hat{y})] &= \sum_{k \in \{0,1\}} \ell_{\mathbf{x}}(y = k, \hat{y}) \cdot P(y = k|\mathbf{x}) \\
&= \sum_{k \in \{0,1\}} (1 - \delta_{y\hat{y}}) \cdot P(y = k|\mathbf{x}) \quad \triangleright \text{ plug in } \ell(\cdot, \cdot) \\
&= 1 - P(y|\mathbf{x}) \\
\operatorname*{argmin}_{\hat{y} \in \{0,1\}} \mathbb{E}_{\mathbf{x}}[\ell(\hat{y})] &= \operatorname*{argmin}_{\hat{y} \in \{0,1\}} \Big[1 - P(y|\mathbf{x})\Big] \quad \triangleright \text{ want to minimize this} \\
\hat{y} &= \operatorname*{argmax}_{y \in \{0,1\}} P(y|\mathbf{x}) \quad \triangleright \ldots, \text{ i.e., MAP estimate!}
\end{aligned}$$

Same in the multi-label case, $k \in \{0,1\}^L$ and thus $y \in \{0,1\}^L$.

Hamming Loss Decomposes Across Labels

$$
\begin{aligned}
L_{\text{Hamming}}(\mathbf{Y}, \hat{\mathbf{Y}}) &= \frac{1}{N \cdot L} \sum_{i=1}^{N} \sum_{i=1}^{L} \mathbb{I}[\hat{y}_j^{(i)} \neq y_j^{(i)}] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left(\mathbb{I}[\hat{y}_1^{(i)} \neq y_1^{(i)}] + \cdots + \mathbb{I}[\hat{y}_L^{(i)} \neq y_L^{(i)}] \right) \\
&= \ell_{0/1}(\mathbf{y}_1, \hat{\mathbf{y}}_1) + \cdots + \ell_{0/1}(\mathbf{y}_L, \hat{\mathbf{y}}_L)
\end{aligned}
$$

and for each j-th label,

$$
\begin{aligned}
\hat{y}_j &= \underset{y_j \in \{0,1\}}{\text{argmax}} \sum_{y_k \in \{0,1\}} P(y_j|, y_k, \mathbf{x}) \\
&= \underset{y \in \{0,1\}}{\text{argmax}}\, P_j(y|\mathbf{x})
\end{aligned}
$$

Hamming Loss Decomposes Across Labels

$$
\begin{aligned}
L_{\text{Hamming}}(\mathbf{Y}, \hat{\mathbf{Y}}) &= \frac{1}{N \cdot L} \sum_{i=1}^{N} \sum_{i=1}^{L} \mathbb{I}[\hat{y}_j^{(i)} \neq y_j^{(i)}] \\
&= \frac{1}{N} \sum_{i=1}^{N} \left(\mathbb{I}[\hat{y}_1^{(i)} \neq y_1^{(i)}] + \cdots + \mathbb{I}[\hat{y}_L^{(i)} \neq y_L^{(i)}] \right) \\
&= \ell_{0/1}(\mathbf{y}_1, \hat{\mathbf{y}}_1) + \cdots + \ell_{0/1}(\mathbf{y}_L, \hat{\mathbf{y}}_L)
\end{aligned}
$$

and for each j-th label,

$$
\begin{aligned}
\hat{y}_j &= \underset{y_j \in \{0,1\}}{\text{argmax}} \sum_{y_k \in \{0,1\}} P(y_j|, y_k, \mathbf{x}) \\
&= \underset{y \in \{0,1\}}{\text{argmax}}\, P_j(y|\mathbf{x})
\end{aligned}
$$

In theory, Hamming loss is optimized by naive binary relevance.

Loss Functions in Multi-label Classification

Hamming loss is not very interesting for structured data. We would not optimize an image recognition algorithm for each pixel individually! But for optimizing 0/1 loss we have a daunting search space:

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

Whatever our search method, the space gets smaller if we can identify independence.

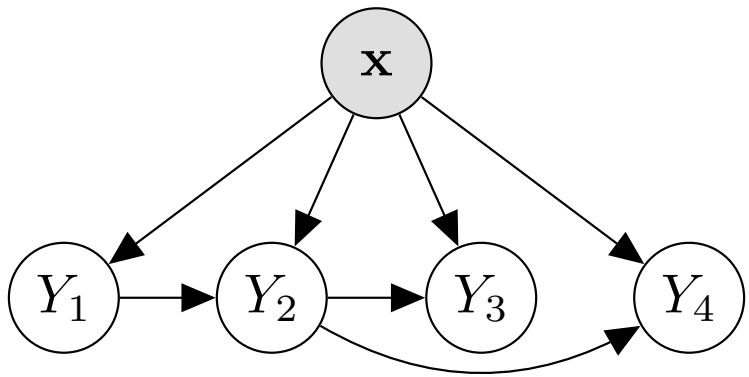
Dependence	Space	Comment
Full	2^L	Should use full cascade or similar
Split	$2^{L/2} + 2^{L/2}$	use something in between
None	$2L$	Maximizing 0/1 and Hamming loss equivalent!

i.e., try to identify label dependence; use it to build a structure.

Outline

Structure for a Multi-label Classifier

We can formulate any structure,

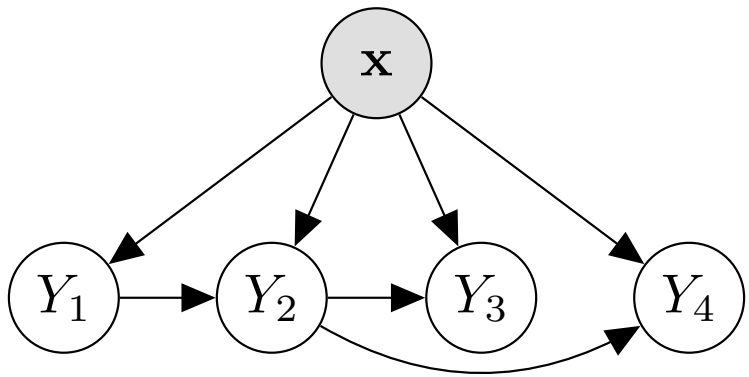


$$\hat{y}_j = \underset{y_j \in \{0,1\}}{\text{argmax}}\, P(y_j | \mathbf{x}, \text{pa}_j)$$

where pa_j = parents of node j. This defines a structure, i.e., graph $\mathcal{G}$.

Structure for a Multi-label Classifier

We can formulate any structure,

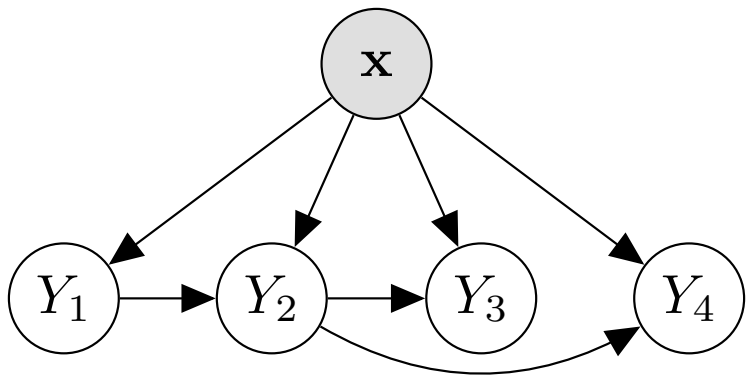


$$\hat{y}_j = \underset{y_j \in \{0,1\}}{\text{argmax}}\, P(y_j | \mathbf{x}, \text{pa}_j)$$

where pa_j = parents of node j. This defines a structure, i.e., graph $\mathcal{G}$. What structure should we use? – The graph $\mathcal{G}^*$ that provides the best score.

Learning Structure

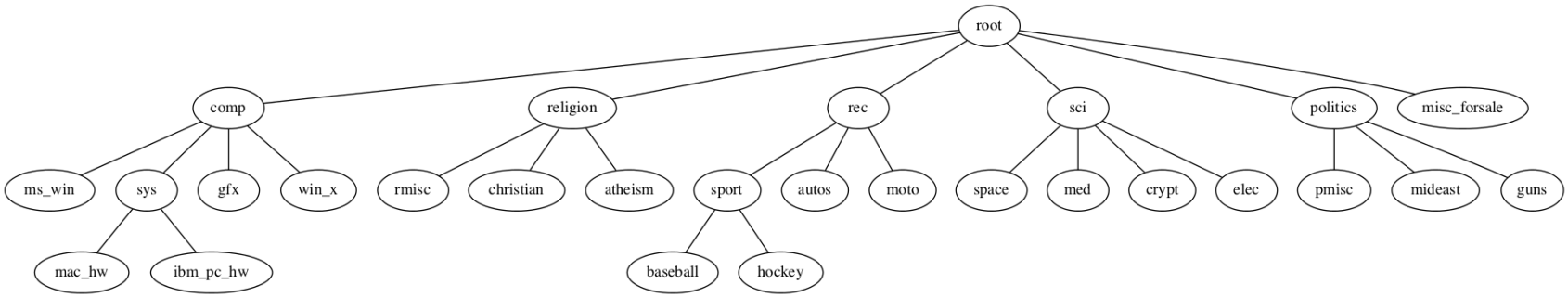
- From a Bayesian network point of view (a probabilistic graphical model), structure represents conditional dependence between variables
- From a neural network point of view (may be considered a special case of a Bayesian network), edges represent functional dependence



i.e., we need to explore the idea of dependence.

Hierarchy and dependence

But wait, (perhaps) I have a structure!



Is a dataset hierarchy a representation of label dependence?

Hierarchy and dependence

But wait, (perhaps) I have a structure!

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

Is a dataset hierarchy a representation of label dependence?

Hierarchy and dependence

But wait, (perhaps) I have a structure!

Is a dataset hierarchy a representation of label dependence?

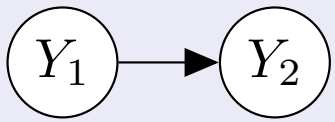
- Yes, but not necessarily the best one!
- Hierarchies normally only represent similarities, but exclusivity is also dependence.
- A random hierarchy/structure may perform as well

Marginal Dependence

Marginal dependence

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

$$P(Y_2) \neq P(Y_2|Y_1)$$
$$P(Y_1)P(Y_2) \neq P(Y_1, Y_2)$$



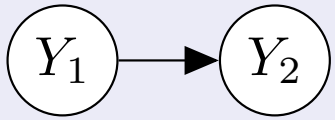
We can estimate from pairwise co-occurrence frequencies in training data, correlation matrix, etc. (the empirical probability P).

Marginal 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. (the empirical probability P).

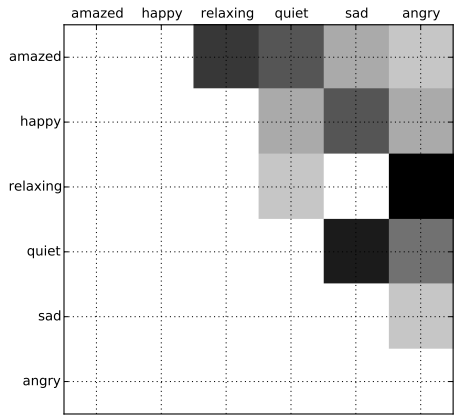


Figure: Music dataset - Mutual Information:
$I(Y_1, Y_2) = \sum_{y_1,y_2} P(y_1, y_2) \log \frac{P(y_1,y_2)}{P(y_1)P(y_2)}$

Toy Problem – Marginal Dependence

X_1	X_2	Y_1	Y_2	$P(\mathbf{x}, \mathbf{y})$
0	0	0	0	0.25
0	1	0	1	0.25
1	0	0	1	0.25
1	1	1	0	0.25
...	...	...	...	0

Is there marginal label dependence between labels Y_1 and Y_2?

Toy Problem – Marginal Dependence

X_1	X_2	Y_1	Y_2	$P(\mathbf{x}, \mathbf{y})$
0	0	0	0	0.25
0	1	0	1	0.25
1	0	0	1	0.25
1	1	1	0	0.25
...	...	...	...	0

Is there marginal label dependence between labels Y_1 and Y_2?

$$P(Y_1, Y_2) = P(Y_1) \cdot P(Y_2) \quad \triangleright \text{ if independent}$$

$$P(Y_1, Y_2 = 1, 1) \neq P(Y_1 = 1) \cdot P(Y_2 = 1)$$

$$0.0 \neq 0.25 \cdot 0.5 = 0.125$$

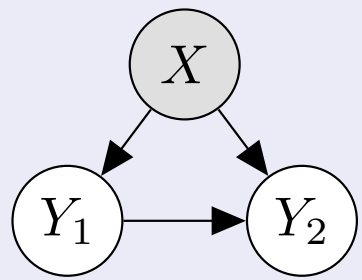
and so on (for $\mathbf{y} \in \{[0,0], [0,1], [1,0], [1,1]\}$). So, yes – there is *marginal* dependence! But what about the input $\mathbf{x}$? We are really interested in $P(\mathbf{y}|\mathbf{x})$ rather than $P(\mathbf{y})$.

Conditional Label Dependence

Conditional dependence

. . . conditioned on input observation **x**.

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

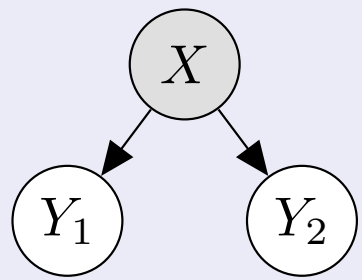


Conditional Label Dependence

Conditional *in*dependence

. . . conditioned on input observation **x**.

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

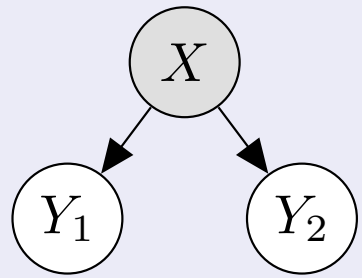


Conditional Label Dependence

Conditional dependence

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

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



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

$$\hat{y}_2 = h_2(\mathbf{x}) = \underset{y_2 \in \{0,1\}}{\text{argmax}}\, P(y_2|\mathbf{x}, y_1)$$

i.e., our models. We need to build and compare models. If the independent model outperforms the joint model, we have conditional dependence!

We can compare two models using a loss function, i.e.,

$$\epsilon_j = \ell\Big(y_j, h_j(\mathbf{x})\Big)$$
$$\epsilon'_j = \ell\Big(y_j, h'(\mathbf{x})_j\Big)$$

where h' models labels together. If $\epsilon'_j < \epsilon_j$, then the labels are conditionally dependent.

We can compare two models using a loss function, i.e.,

$$\epsilon_j = \ell\Big(y_j, h_j(\mathbf{x})\Big)$$
$$\epsilon'_j = \ell\Big(y_j, h'(\mathbf{x})_j\Big)$$

where h' models labels together. If $\epsilon'_j < \epsilon_j$, then the labels are conditionally dependent. Notice that

- h_j is the base model; and therefore,
- conditional dependence is dependent on the base model!

Given two labels, and two binary classifiers (one for each), we can also compare

$$\epsilon_j = \ell\Big(y_j, h_j(\mathbf{x})\Big)$$
$$\epsilon_k = \ell\Big(y_k, h_k(\mathbf{x})\Big)$$

The function ℓ takes into account $\mathbf{x}$. Therefore, we can measure dependence among the errors as a measure of conditional dependence.

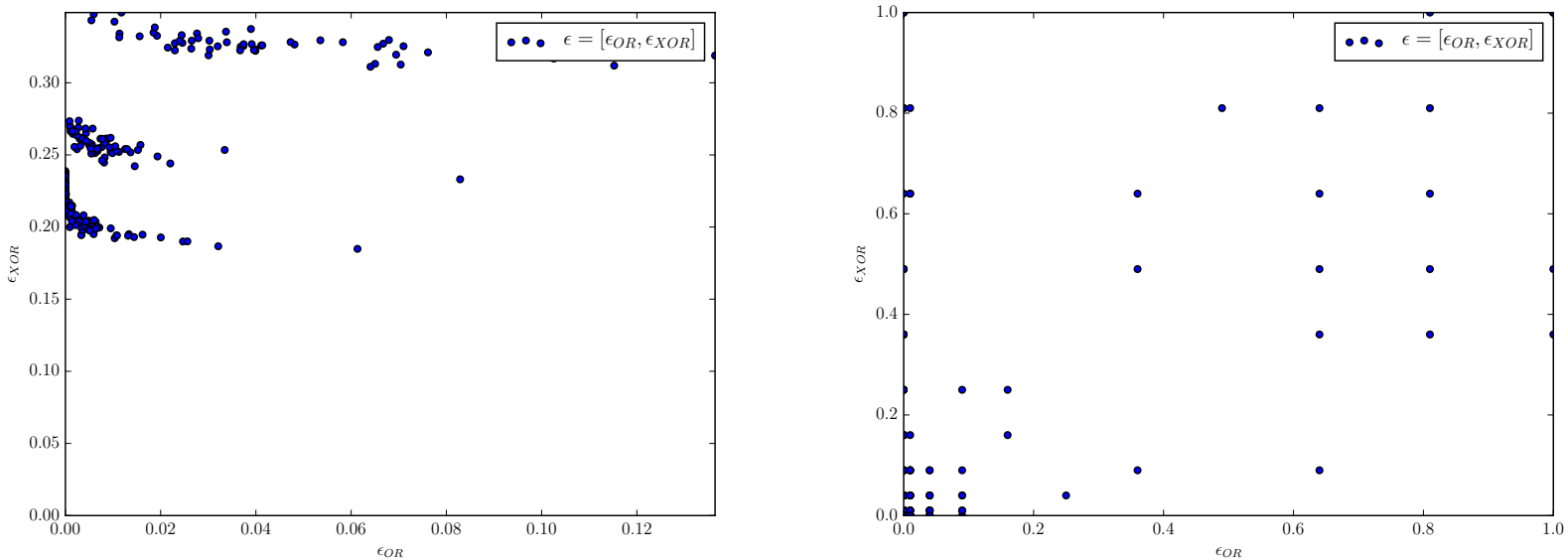


Figure: Errors from logistic regression (left) and decision tree (right).

Conditional dependence in Multi-Output Learning

Suppose two problems:

$$y_1 = f_1(\mathbf{x}) + \epsilon_1, \text{ and } y_2 = f_2(\mathbf{x}) + \epsilon_2$$

If

1. ϵ_1 and ϵ_2 are independent from $\mathbf{x}$, *and*
2. ϵ_1 and ϵ_2 are also independent from each other,

then y_1 and y_2 are conditionally independent given $\mathbf{x}$.

Conditional dependence in Multi-Output Learning

Suppose two problems:

$$y_1 = f_1(\mathbf{x}) + \epsilon_1, \text{ and } y_2 = f_2(\mathbf{x}) + \epsilon_2$$

If

1. ϵ_1 and ϵ_2 are independent from $\mathbf{x}$, *and*
2. ϵ_1 and ϵ_2 are also independent from each other,

then y_1 and y_2 are conditionally independent given $\mathbf{x}$.

We may (in *some cases*) assume 1. on account of MLE (although this depends on the model, data, basis functions, etc.). So therefore we may measure dependence among $\epsilon_1, \ldots, \epsilon_L$ to infer conditional dependence among $Y_1, \ldots, Y_L$.

Toy Problem – *Conditional* Dependence

X_1	X_2	Y_1	Y_2	$P(\mathbf{x}, \mathbf{y})$
0	0	0	0	0.25
0	1	0	1	0.25
1	0	0	1	0.25
1	1	1	0	0.25
...	...	...	...	0

Is there conditional dependence between labels Y_1 and Y_2?

Toy Problem – *Conditional* Dependence

X_1	X_2	Y_1	Y_2	$P(\mathbf{x}, \mathbf{y})$
0	0	0	0	0.25
0	1	0	1	0.25
1	0	0	1	0.25
1	1	1	0	0.25
...	...	...	...	0

Is there conditional dependence between labels Y_1 and Y_2? We may test the conditional distributions, independence if:

$$P_{\mathbf{x}}(Y_1, Y_2) = P_{\mathbf{x}}(Y_1) \cdot P_{\mathbf{x}}(Y_2)$$
$$P(Y_2|x_1, x_2) = P(Y_2|Y_1, x_1, x_2)$$

Toy Problem – *Conditional* Dependence (cont. . . .)

Supposing that $P(X_j) = 0.5$ (for both) and

X_1	X_2	$P(Y_2 \mid X_1, X_2)$
0	0	0.0
0	1	1.0
1	0	1.0
1	1	0.0
. . .	. . .	0.0

X_1	X_2	Y_1	$P(Y_2 \mid X_1, X_2, Y_1)$
0	0	0	0.0
0	1	0	1.0
1	0	0	1.0
1	1	1	0.0
. . .	. . .	. . .	0.0

then we see that for $\mathbf{x} = [1, 0]$,

$$P(Y_2 \mid \mathbf{x}) = \sum_{y_1 \in \{0,1\}} P(Y_2 \mid \mathbf{x}, y_1)$$

$$1 = 1 + 0 = 1$$

so there is conditional independence; $Y_2 \perp\!\!\!\perp Y_1 \mid X_1, X_2$.

The Logical Problem

Example (The Logical Toy Problem)

		OR	AND	XOR
X_1	X_2	Y_1	Y_2	Y_3
0	0	0	0	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	0

- Each label can be determined from the input independently of other labels

The Logical Problem

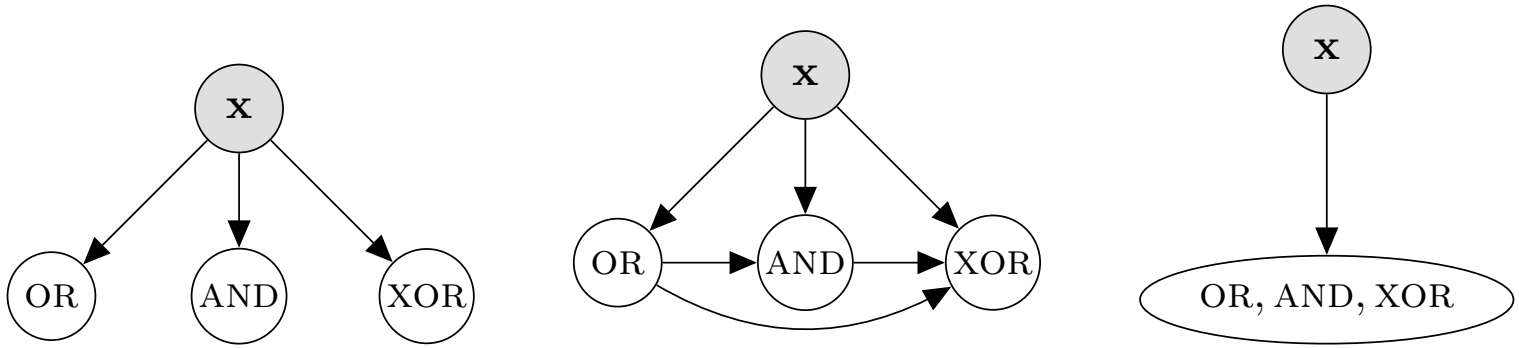


Figure: BR (left), CC (middle), LP (right)

Table: The Logical problem, base classifier logistic regression.

Metric	BR	CC	LP
Hamming score	0.83	1.00	1.00
Exact match	0.50	1.00	1.00

- "Each label can be determined from the input independently of other labels" ...
- ... 'can be' ... given the right base predictive model!

$P(Y_2|X_1, X_2, Y_1) \neq \hat{P}(Y_2|X_1, X_2, Y_1)$

The predictive model is built on[4] $\hat{P}$.

X_1	X_2	Y_1	$P(Y_2\|X_1, X_2)$
0	0	0	0.0
0	1	0	1.0
1	0	0	1.0
1	1	1	0.0
...	...	...	0.0

X_1	X_2	Y_1	$\hat{P}(Y_2\|X_1, X_2)$
0	0	0	1.0
0	1	0	1.0
1	0	0	1.0
1	1	1	0.0
...	...	...	0.0

X_1	X_2	Y_1	$P(Y_2\|X_1, X_2, Y_1)$
0	0	0	0.0
0	1	0	1.0
1	0	0	1.0
1	1	1	0.0
...	...	...	0.0

X_1	X_2	Y_1	$\hat{P}(Y_2\|X_1, X_2, Y_1)$
0	0	0	0.0
0	1	0	1.0
1	0	0	1.0
1	1	1	0.0
...	...	...	0.0

and it may be estimated badly; not enough training data, or iterations, or we were using a rough heuristic (e.g., *greedy* search) to search the probability tree quickly.

[4] Implicitly, except on this slide

The Logical Problem

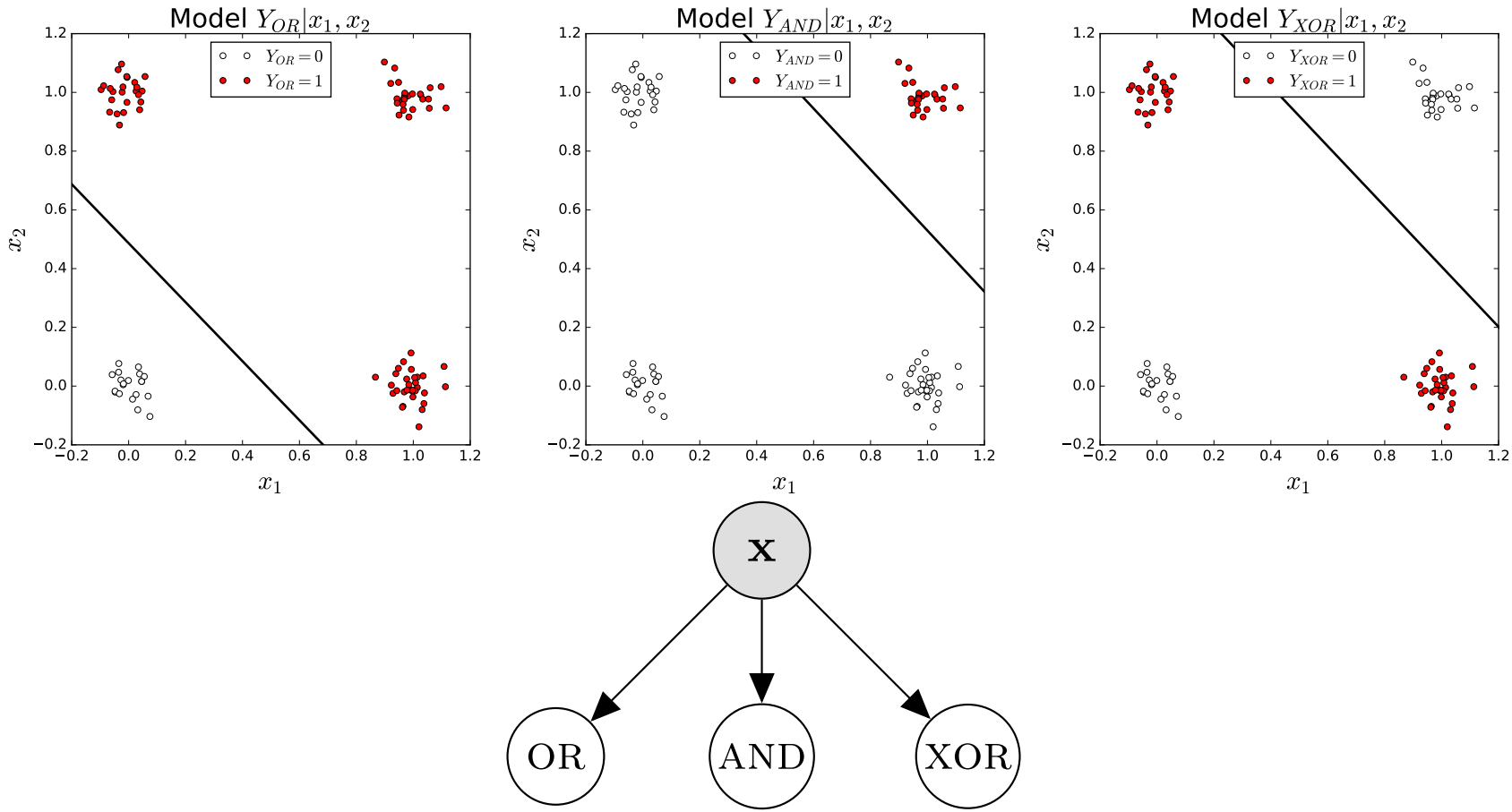


Figure: Independent models with linear decision boundaries (solid line, estimated with logistic regression) not viable for Y_{XOR} label

Solution via Structure

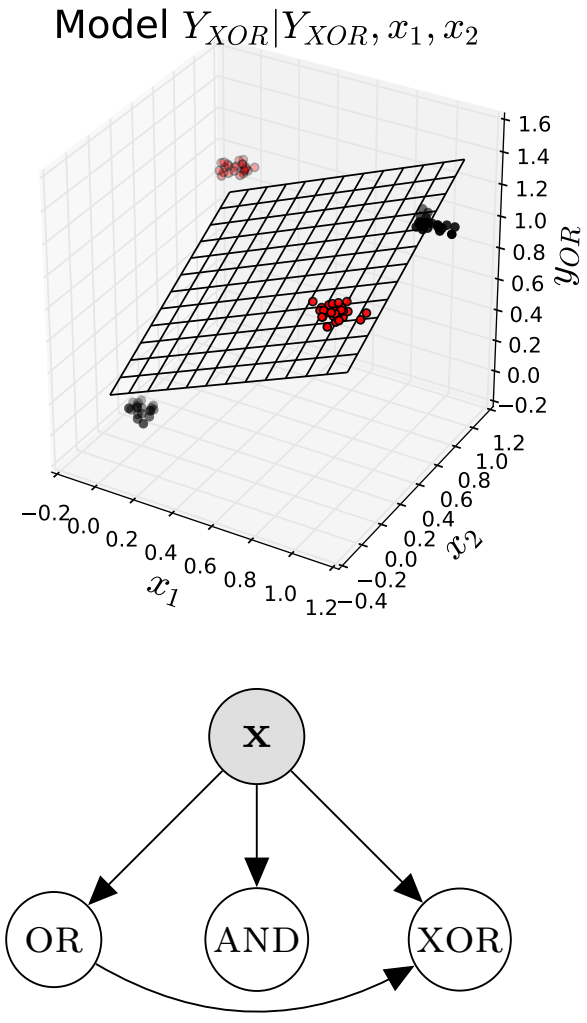
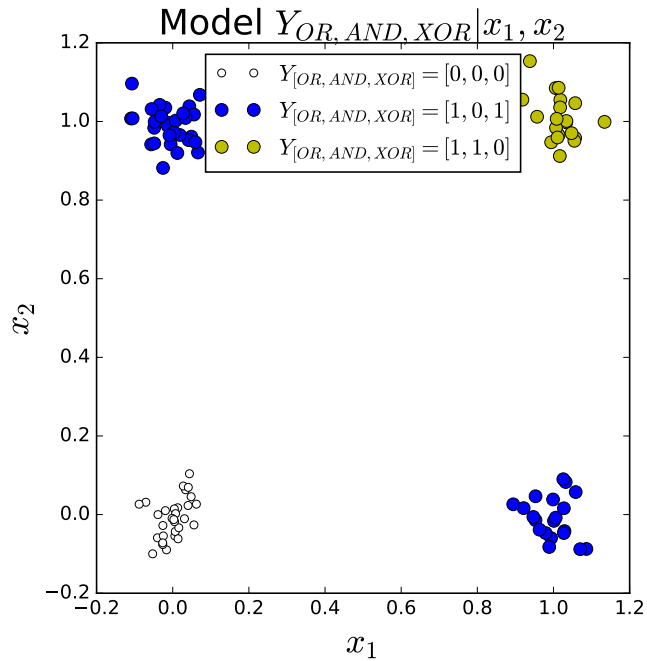


Figure: Classifier chains (CC): linear model now applicable to Y_{XOR}

Solution via Multi-class Decomposition



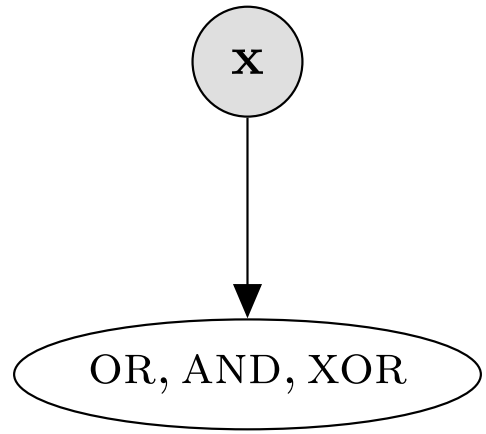


Figure: Modelling labels together in a single class variable taking 2^L values: solvable with one-vs-one multi-class decomposition for any (e.g., linear) base classifier.

Solution via Conditional Independence

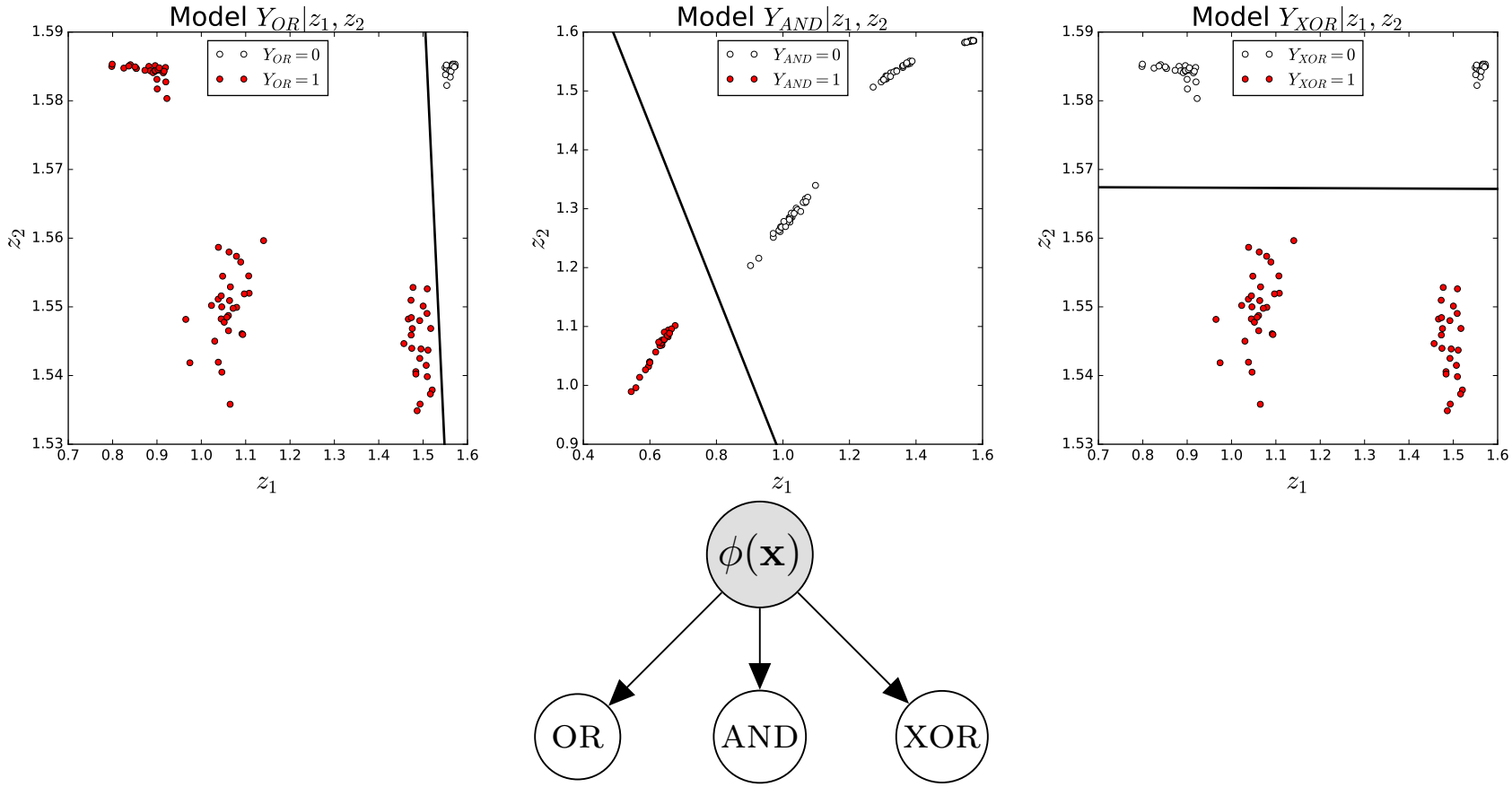


Figure: Solution via latent structure (e.g., random RBF) to new input space $\mathbf{z}$; creating independence: $P(y_{\text{XOR}}|\mathbf{z}, y_{\text{OR}}, y_{\text{AND}}) \approx P(y_{\text{XOR}}|\mathbf{z})$.

Solution via Suitable Base-classifier

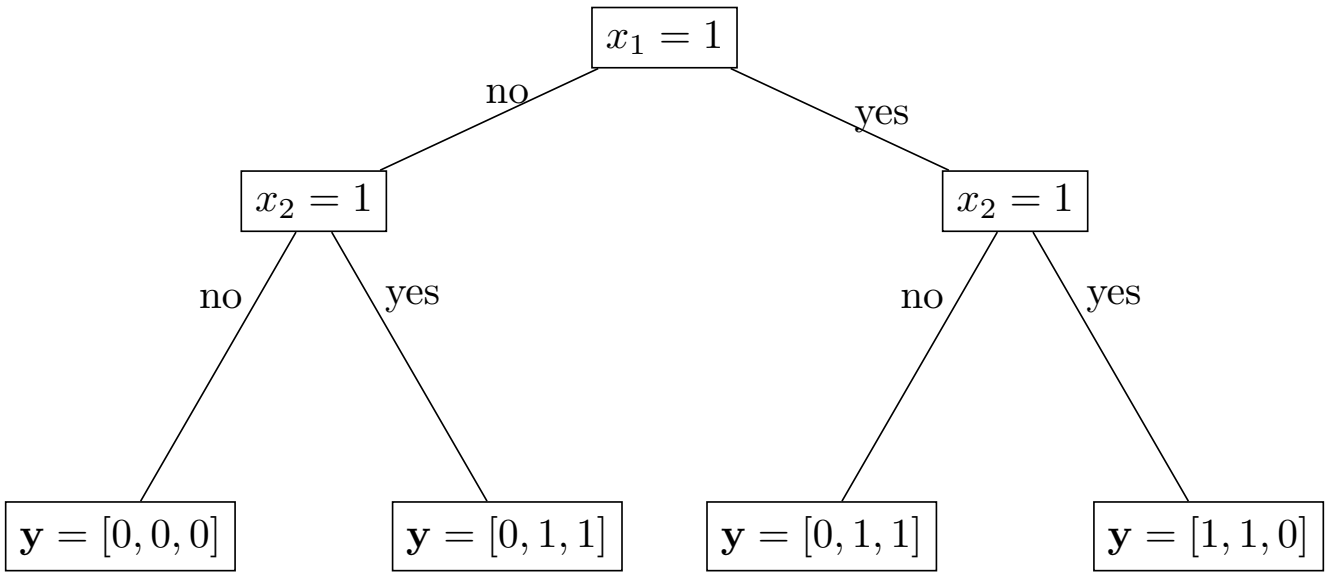
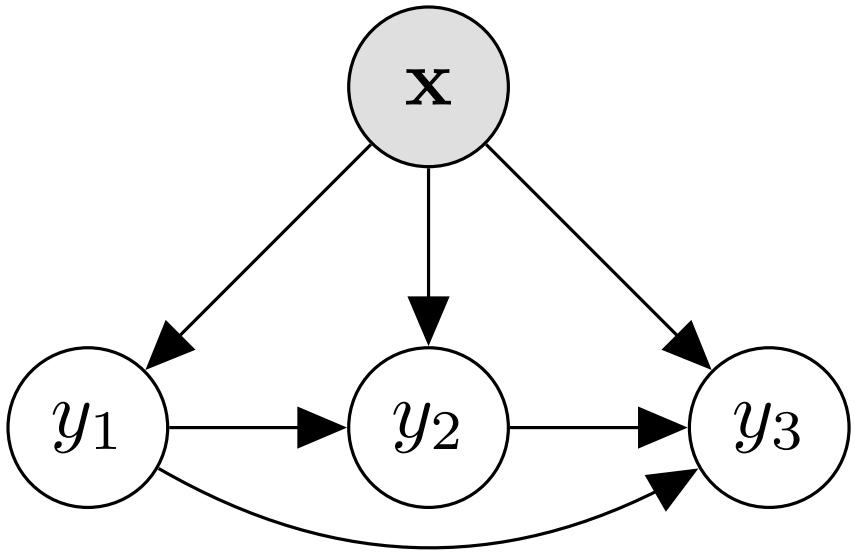


Figure: Solution via non-linear classifier (e.g., Decision Tree). Leaves hold examples, where $\mathbf{y} = [y_{\text{AND}}, y_{\text{OR}}, y_{\text{XOR}}]$

Outline

Inference

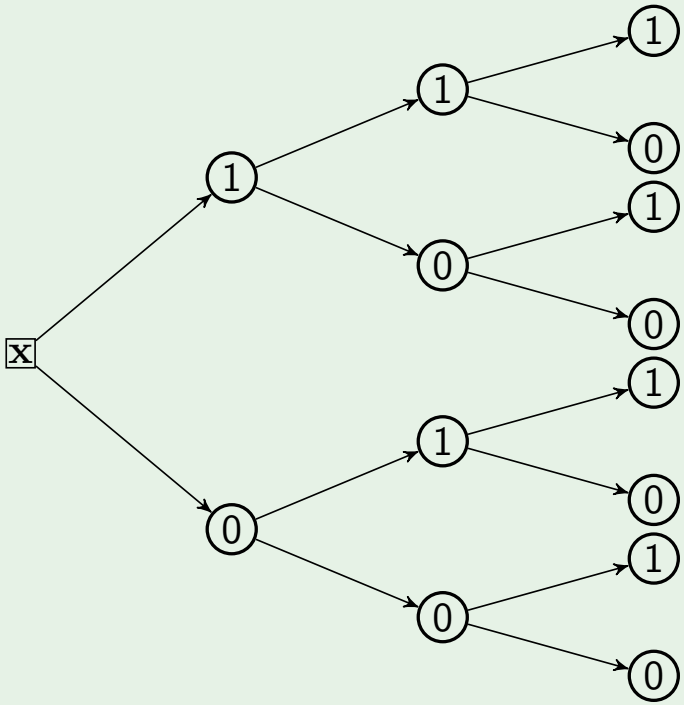


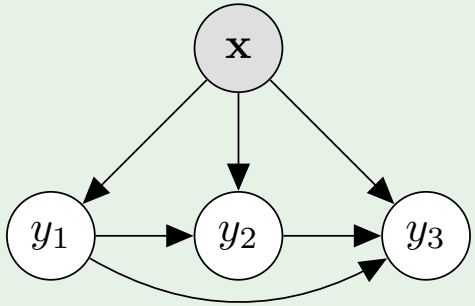
- We have trained out base models $\mathbf{h} = (h_1, h_2, h_3)$
- Now we observe a new $\mathbf{x}$. We formulate a query for our model:

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

What is the best $\hat{\mathbf{y}}$?

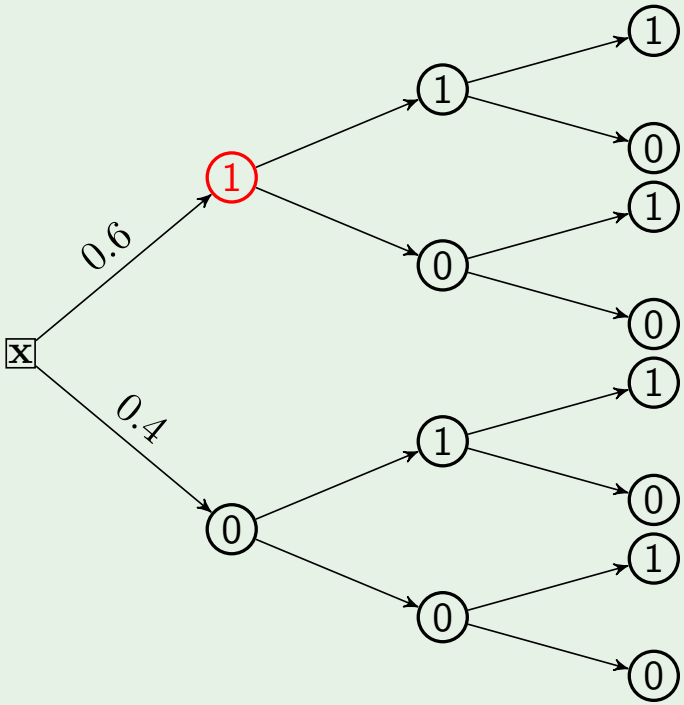
Greedy Search



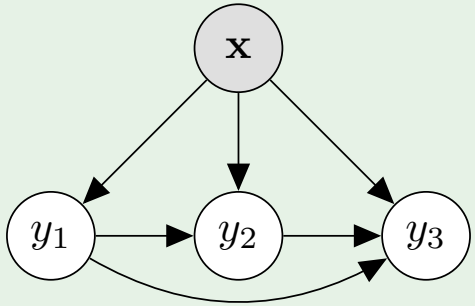


$$\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [?, ?, ?]$$

Greedy Search

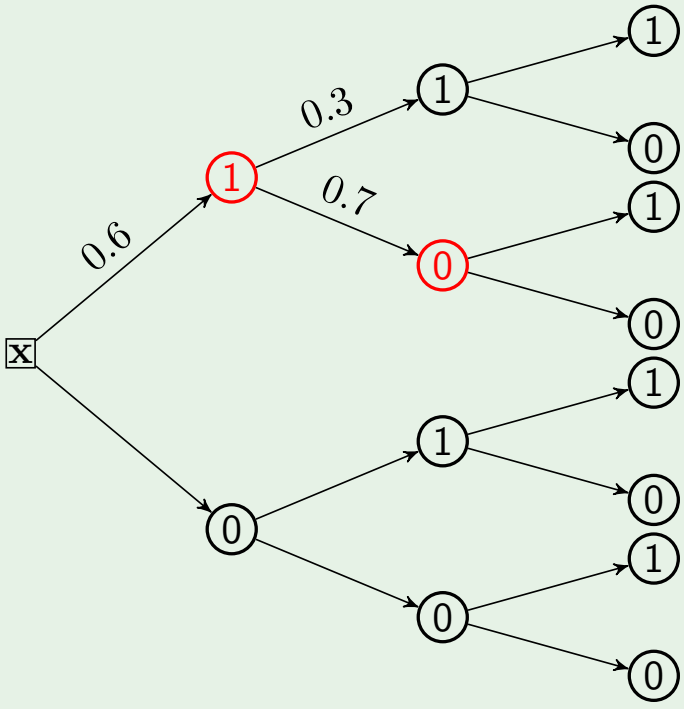


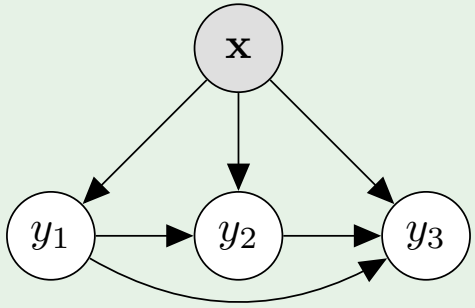
$\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [1, ?, ?]$



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

Greedy Search

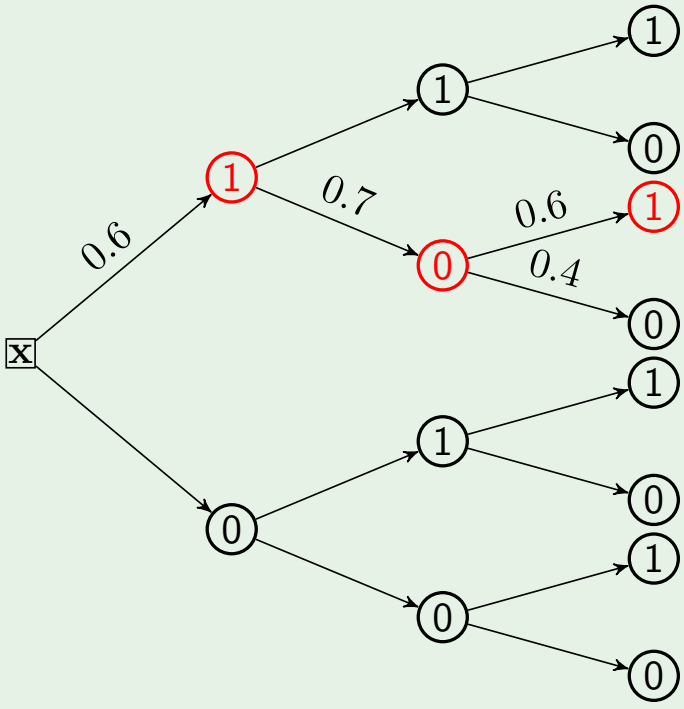




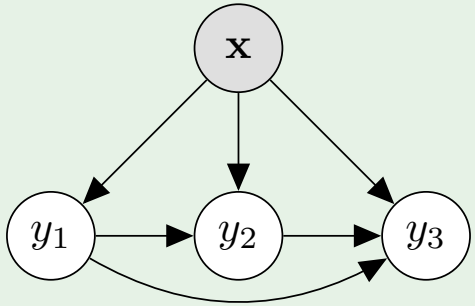
1. $\hat{y}_1 = h_1(\mathbf{x}) = \text{argmax}_{y_1} P(y_1|\mathbf{x}) = 1$
2. $\hat{y}_2 = h_2(\mathbf{x}, \hat{y}_1) = \ldots = 0$

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

Greedy Search

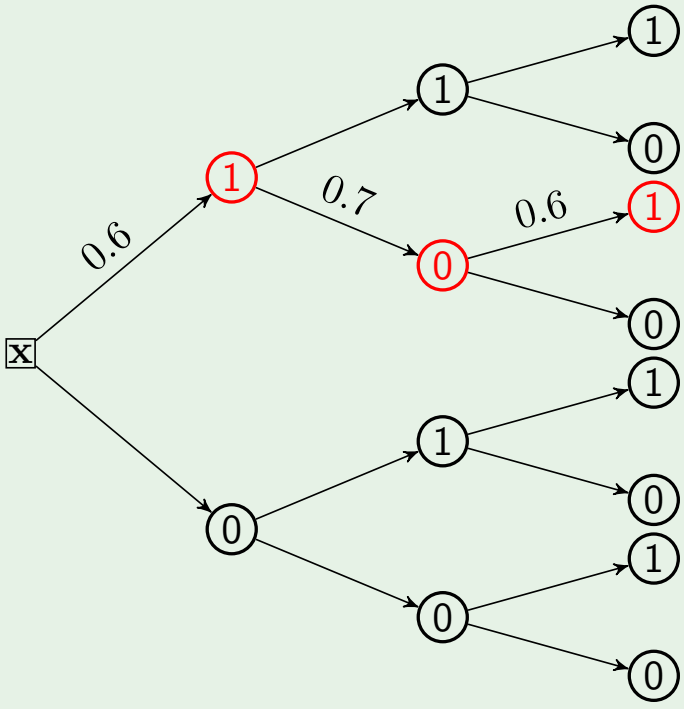


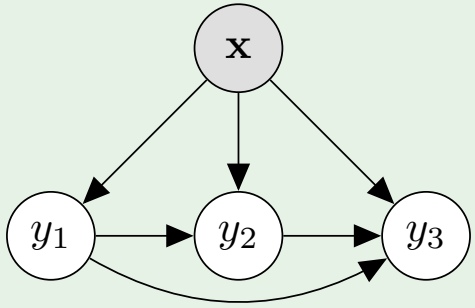
$\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x}) = [1, 0, 1]$



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

Greedy Search





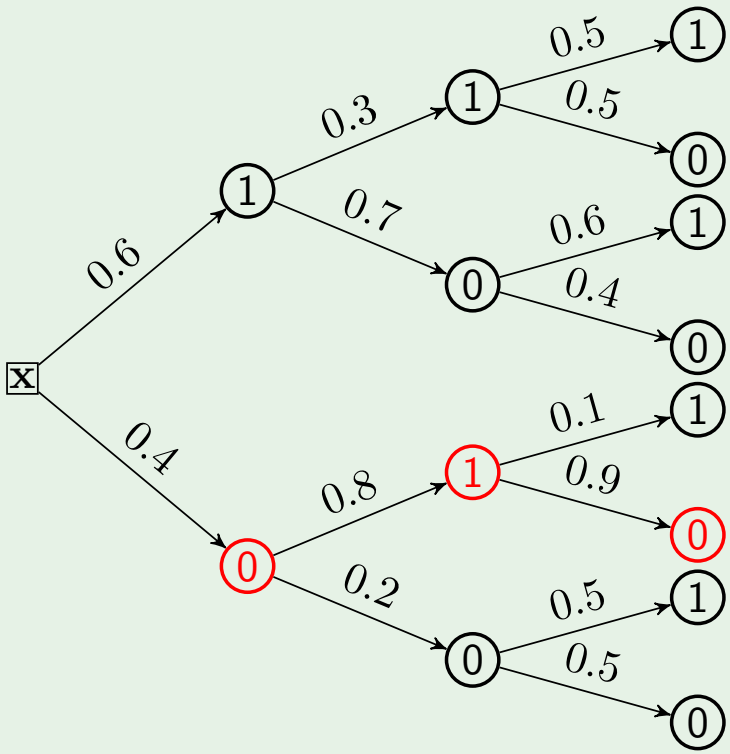
1. $\hat{y}_1 = h_1(\mathbf{x}) = \text{argmax}_{y_1} P(y_1|\mathbf{x}) = 1$
2. $\hat{y}_2 = h_2(\mathbf{x}, \hat{y}_1) = \ldots = 0$
3. $\hat{y}_3 = h_3(\mathbf{x}, \hat{y}_1, \hat{y}_2) = \ldots = 1$

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

Fast, but susceptible to error proagation, since in fact,

$$\underset{\mathbf{y} \in \{0,1\}^L}{\text{argmax}}\, P(\mathbf{y}|\mathbf{x}) \neq [\underset{y_1 \in \{0,1\}}{\text{argmax}}\, P(y_1|\mathbf{x}), \ldots, \underset{y_3 \in \{0,1\}}{\text{argmax}}\, P(y_1|\mathbf{x}, \hat{y}_1, \hat{y}_2)]$$

Bayes-Optimal (Full) Search

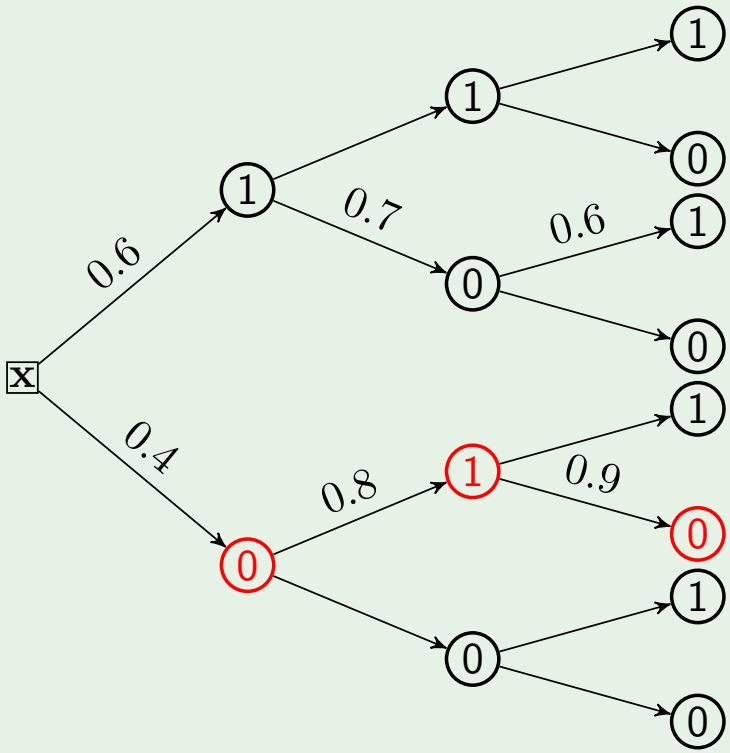


1. $P(\mathbf{y} = [0, 0, 0]|\mathbf{x}) = 0.040$
2. $P(\mathbf{y} = [0, 0, 1]|\mathbf{x}) = 0.040$
3. $P(\mathbf{y} = [0, 1, 0]|\mathbf{x}) = 0.288$
4. ...
6. $P(\mathbf{y} = [1, 0, 1]|\mathbf{x}) = 0.252$
7. ...
8. $P(\mathbf{y} = [1, 1, 1]|\mathbf{x}) = 0.090$

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

- Search space of $\{0, 1\}^L$ paths is too much

Monte-Carlo Search / Likelihood Weighting

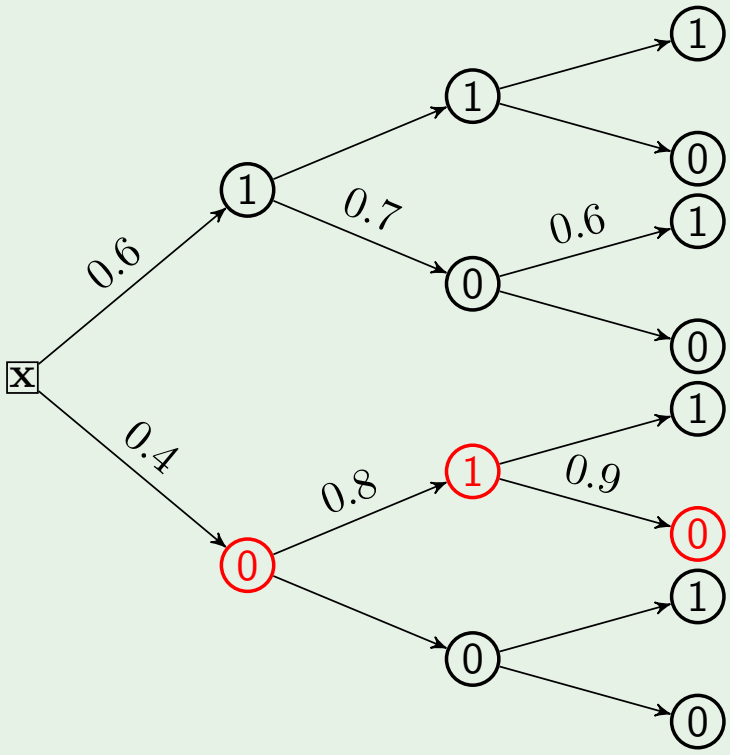


Generate samples $\{\mathbf{y}_t\}_{t=1}^{T}$, directly from P, where

$$y_1^{(t)} \sim P(y_1|\mathbf{x})$$
$$y_2^{(t)} \sim P(y_2|\mathbf{x}, y_1^{(t)})$$
$$y_3^{(t)} \sim P(y_3|\mathbf{x}, y_1^{(t)}, y_2^{(t)})$$

provides $\mathbf{y}_t = [y_1^{(t)}, y_2^{(t)}, y_3^{(t)}]$.

Monte-Carlo Search / Likelihood Weighting



And thus,

- $P([1,0,1]|\mathbf{x}) = 0.252$
- $P([0,1,0]|\mathbf{x}) = 0.288$

$$\text{return} \quad \underset{\mathbf{y}_t}{\text{argmax}}\, P(\mathbf{y}_t|\mathbf{x})$$

Note that,

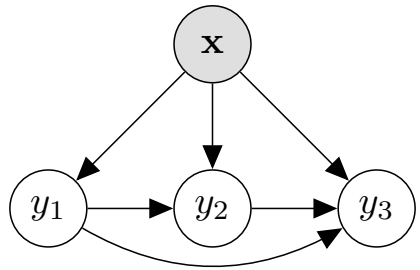
$$\lim_{T\to\infty} \texttt{count}(\mathbf{y}_t)/T = P(\mathbf{y}|\mathbf{x})$$

hence at the limit, we expect 28.8% of samples to be $[0,1,0]$.

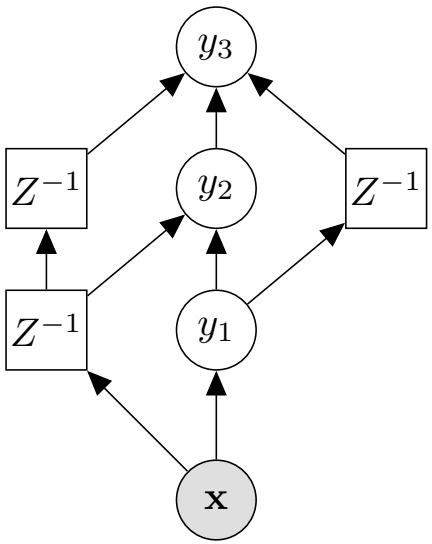
Outline

A Multi-label Classifier as a Neural Network

We can move from the probabilistic graphical model view of



to a neural network view . . .

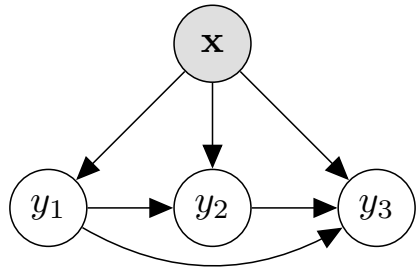


Where Z^{-1} is a delay node

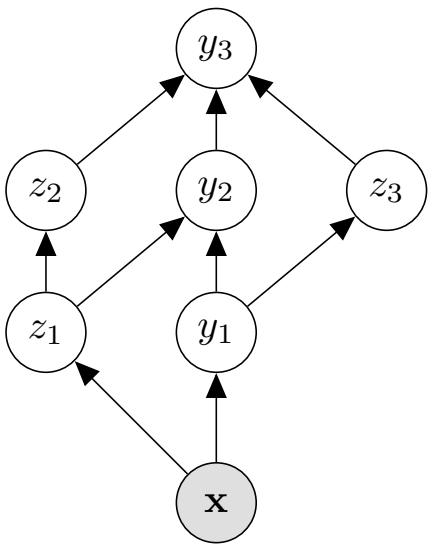
with forward propagation. This can be generalized . . .

A Multi-label Classifier as a Neural Network

We can move from the probabilistic graphical model view of



to a neural network view . . .



. . . or nodes where $w_j = 1$ and linear activation function, such that $y_2 = h(w_j \cdot x, y_1) = h(x, y_1)$.

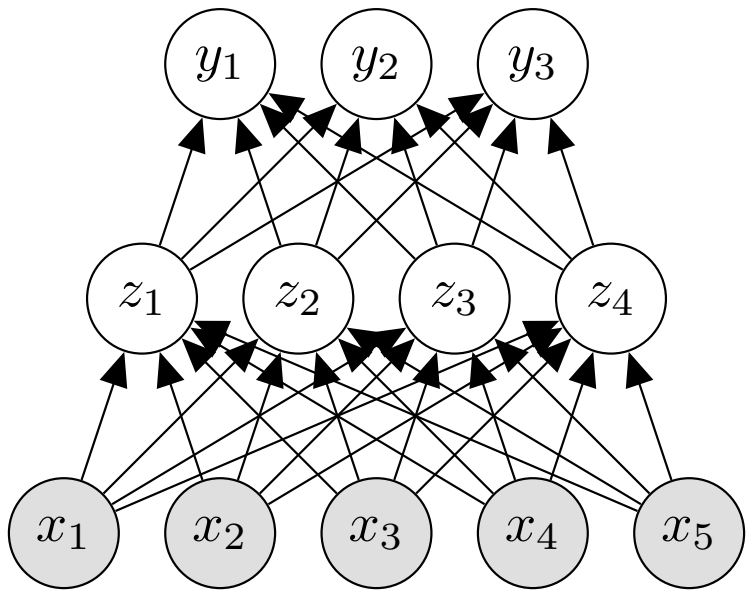
with forward propagation. This can be generalized . . .

Multi-layer Perceptron

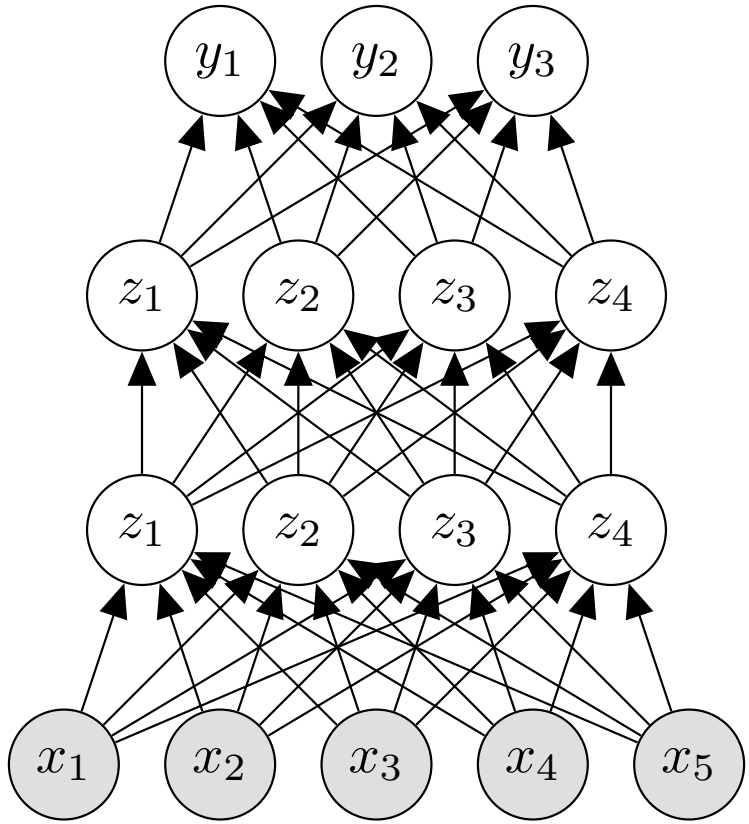
- We propagate upward, such that

$$z_k = f(\mathbf{w}_k^{(1)\top}\mathbf{x}) \quad \text{and then} \quad y_j = f(\mathbf{w}_k^{(2)\top}\mathbf{x})$$

- Usually we consider the softmax function so that outputs sum to 1. In multi-label classification, there is no need for this. We can just use a logistic loss.
- Train with, e.g., gradient descent + error back-propagation

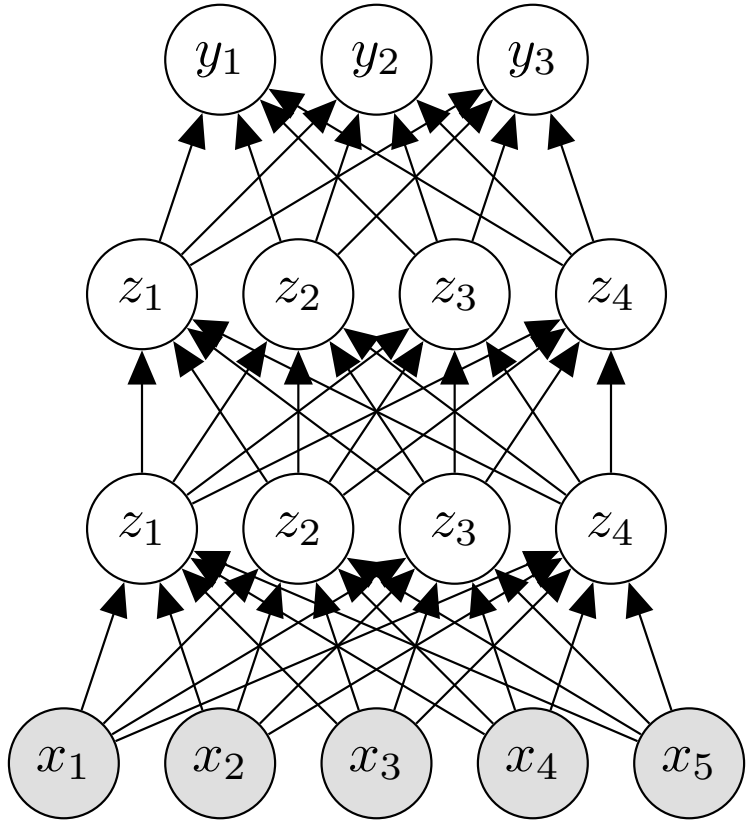


Multi-output Neural Networks



- Add many layers = deep learning scheme.
- Engineer features from the input *or the label space* (e.g., meta labels);
- Learn our own hidden nodes with back propagation

Multi-output Neural Networks

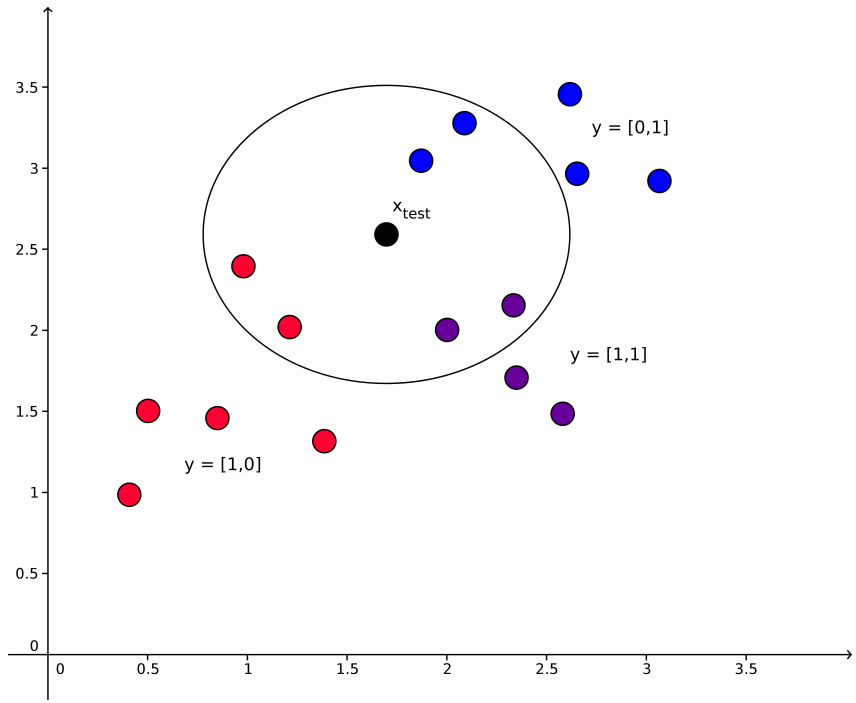


- Label dependence is removed by inner layers,
- May be effective, but many iterations to train, how many nodes, layers; what learning rate?, etc. (hyper-parameter tuning).

Outline

Multi-label k-Nearest Neighbours

In kNN, we compare each test instance $\mathbf{x}$ to the k-nearest instances in the training set (i.e., its neighbourhood).



Multi-label k-Nearest Neighbours

In kNN, we compare each test instance $\mathbf{x}$ to the k-nearest instances in the training set (i.e., its neighbourhood). In the single-label binary case:

$$P(Y = 1|\mathbf{x}) = \frac{1}{k} \sum_{n \in \mathrm{Ne}_k^{\mathbf{x}}} y_n$$

where $\mathrm{Ne}_k^{\mathbf{x}}$ contains [the indices of] the k *closest* points to $\mathbf{x}$ (e.g., Euclidean distance). Classification can be made with a threshold,

$$\hat{y} = h(\mathbf{x}) = \begin{cases} 1 & P(Y = 1|\mathbf{x}) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Multi-label k-Nearest Neighbours

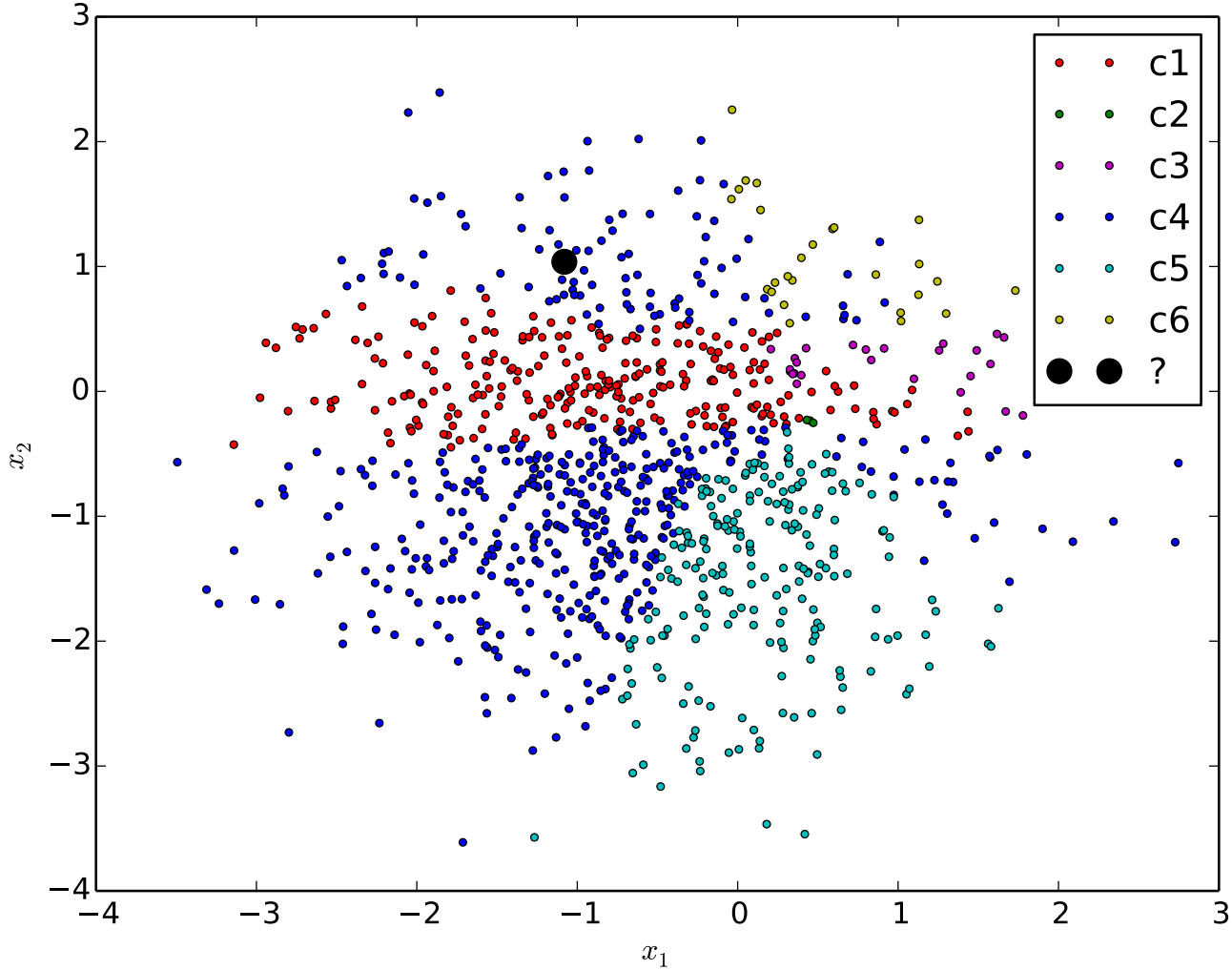
In kNN, we compare each test instance $\mathbf{x}$ to the k-nearest instances in the training set (i.e., its neighbourhood). For the j-th label (in the multi-label case):

$$P(Y_j = 1|\mathbf{x}) = \frac{1}{k} \sum_{n \in \mathrm{Ne}_k^{\mathbf{x}}} y_j^{(n)}$$

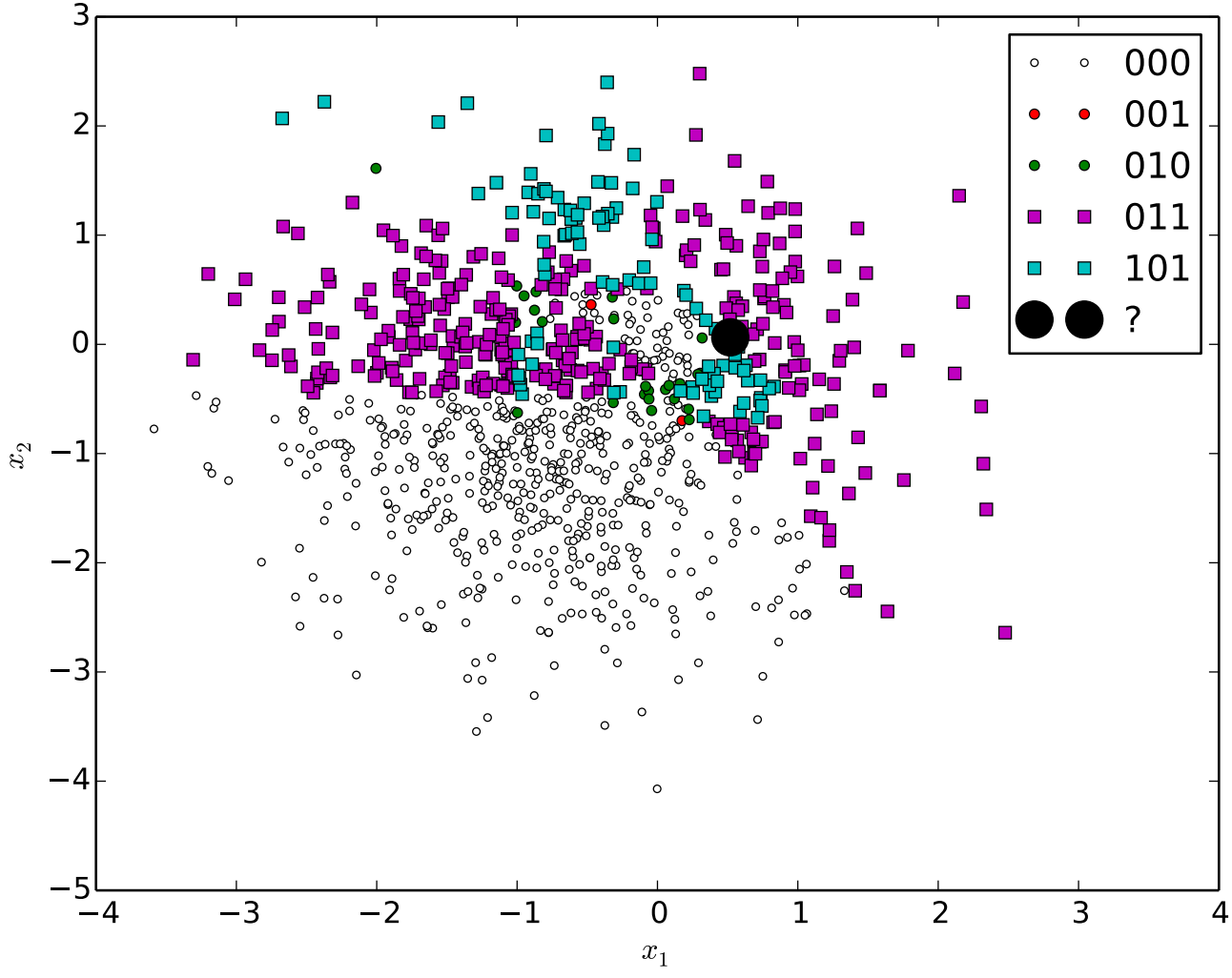
and classification can be made with a threshold,

$$\hat{y}_j = h_j(\mathbf{x}) = \begin{cases} 1 & P(Y_j = 1|\mathbf{x}) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

and we return $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_L]$.



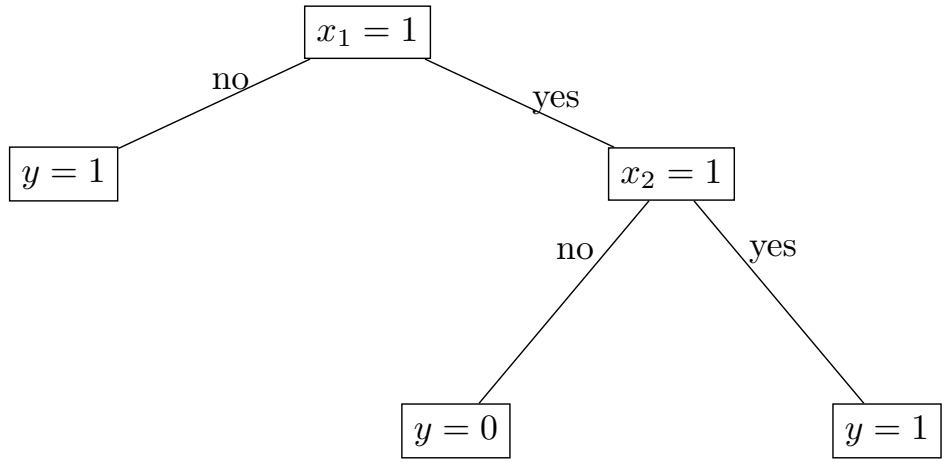
Single-label *k*NN



Multi-label *k*NN

Outline

Decision Trees



X_1	X_2	Y
0	1	1
0	0	1
1	0	0
1	1	1

- Start with dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ at the root
- Iteratively split into subsets $S_{x_1=0}$ and $S_{x_1=1}$, etc., e.g.,
  - first leaf contains set $S =$

X_2	Y
1	1
0	1

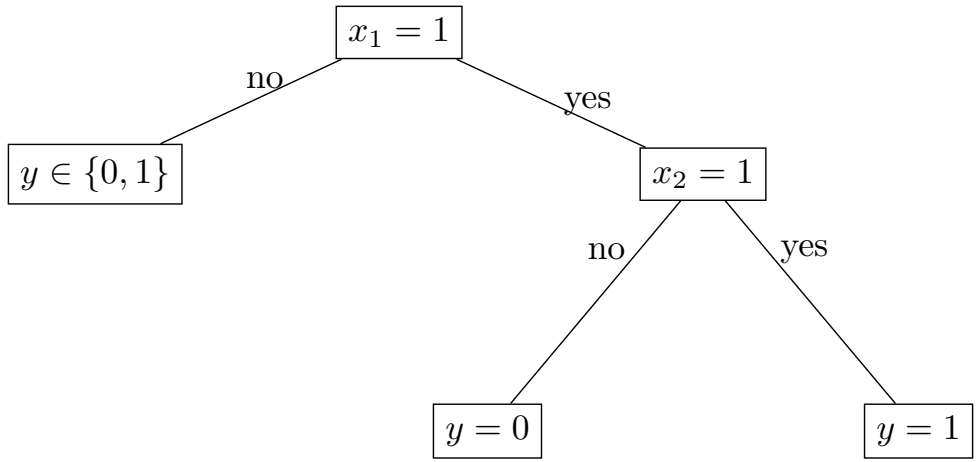
  - second node contains set $S =$

X_2	Y
0	0
1	1

  - second leaf contains $S =$

X_2	Y
0	0

Decision Trees



We do not split randomly; we take the *best* split, e.g., using:

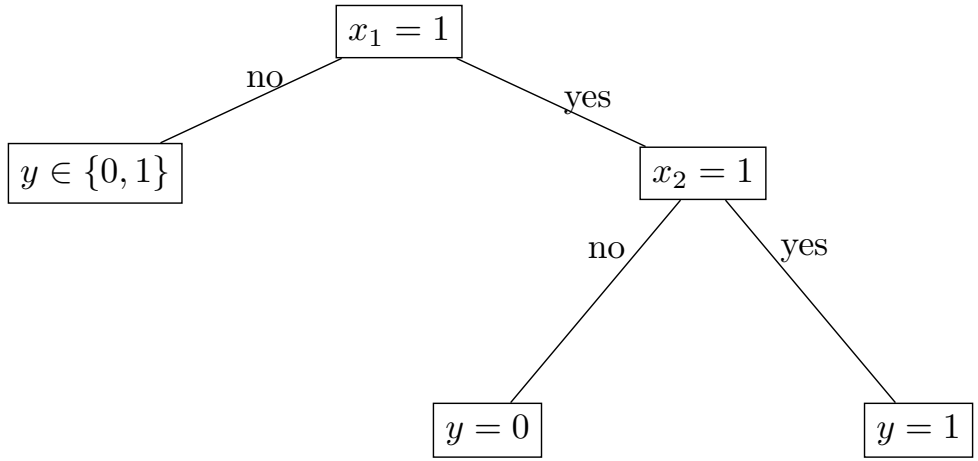
$$H(S) = -\sum_{k \in \{0,1\}} \Big(P(y = k) \log_2 P(y = k) \Big) \quad \triangleright \text{ entropy}$$

$$G(S, X) = H(S) - \underbrace{\sum_{v \in \mathcal{X}} \frac{|S_v|}{|S|} \cdot H(S_v)}_{H(S|X)} \quad \triangleright \text{ information gain}$$

$$S_v = \{(\mathbf{x}, y) \in S | x = v\}$$

where $y \in \{0, 1\}$, $x \in \mathcal{X}$ (for attribute X), e.g., $\mathcal{X} = \{0, 1\}$.

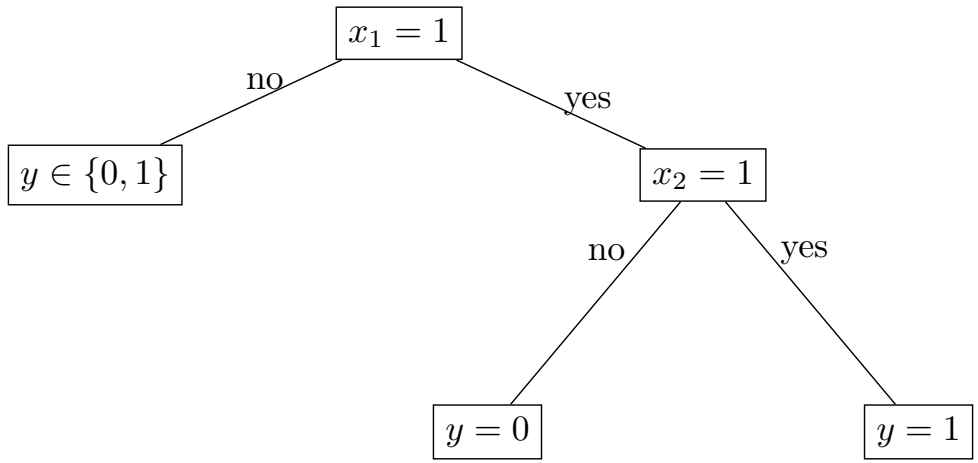
Decision Trees



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



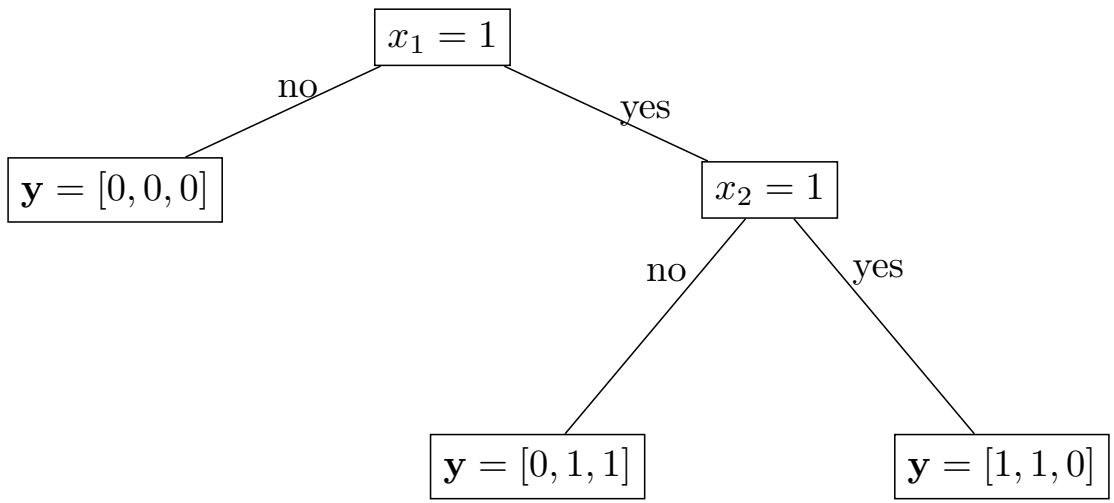
- Non-linear decision boundary
- We may stop early, prune afterwards, etc.
- Interpretable
- Other splitting measures can be used (e.g., Gini impurity)
- Can take the majority vote at inpure leaves.
- Susceptible to overfitting, but works well in an ensemble

Multi-label Decision Trees

Using multi-label entropy,

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

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



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

Outline

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

"Extreme" multi-label classification considers up to millions of labels.

Case study: Winning solution of the LSHTC4 challenge

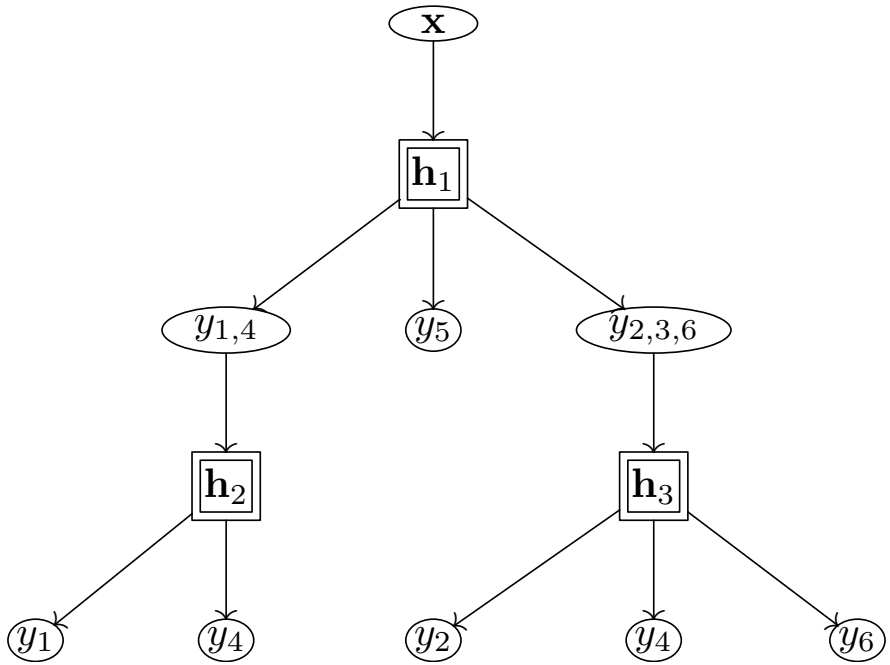
(325,056 labels, 2.4M examples)

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.

Obtained best performance. No cluster, no GPUs.

Hierarchy of Meta Labels

A useful approach to large-scale multi-label learning:



1. Cluster labels (e.g., randomly, hierarchical k-means, or use pre-defined hierarchy)
2. Apply problem transformation / base classifiers

A significant speed improvement, easy to distribute across machines if necessary.

Outline

Summary

- Multi-label classification: essential for many applications.
  - A generalization of single-label classification
  - A special case of multi-output and structured-output prediction

  Approaches with:
  - Bayesian Networks (and approximations) via *transformation*
  - Neural Networks
  - k-Nearest Neighbours
  - Decision Trees

Classification

Jesse Read

[D&K] IoT Stream Data Mining 2017-2018

December 13, 2017