A Deep Interpretation of Classifier Chains

Jesse Read and Jaakko Holmén

http://users.ics.aalto.fi/{jesse,jhollmen}/

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

Leuven, Belgium. Fri. 31st Oct., 2014

Multi-label Classification

$\mathbf{x} =$  , $\mathbf{y} = [y_1, \ldots, y_L] =$

$$[y_{\texttt{beach}}, y_{\texttt{sunset}}, y_{\texttt{foliage}}, y_{\texttt{field}}, y_{\texttt{mountain}}, y_{\texttt{urban}}] = [1, 0, 1, 0, 0, 0]$$

Multi-label Classification

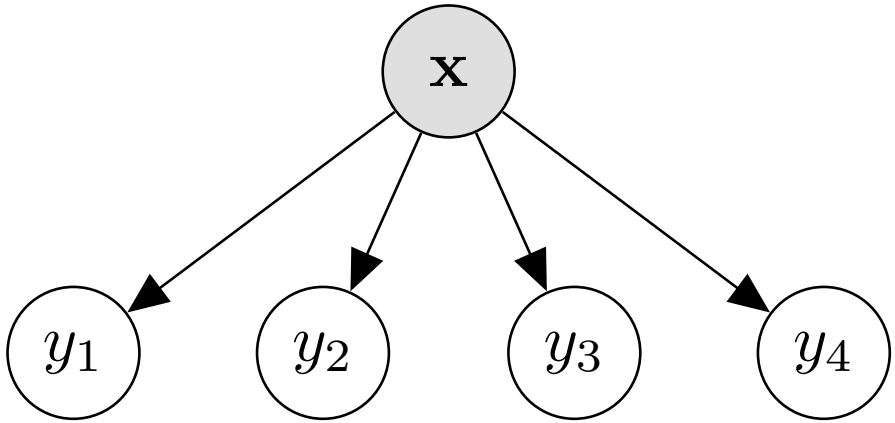
Map D input variables (feature attributes) to L output variables (labels).

X_1	X_2	X_3	$\ldots$	X_D	Y_1	Y_2	Y_3	Y_4
x_1	x_2	x_3	$\ldots$	x_D	0	0	0	1
x_1	x_2	x_3	$\ldots$	x_D	1	1	1	0
x_1	x_2	x_3	$\ldots$	x_D	0	0	0	1
x_1	x_2	x_3	$\ldots$	x_D	1	1	0	1
$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	$\ldots$	$\tilde{x}_D$	?	?	?	?

Build model $\mathbf{h}$, such that $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_L] = \mathbf{h}(\tilde{\mathbf{x}})$.

Binary Relevance (BR)

- Train L independent models $\mathbf{h} = (h_1, \ldots, h_L)$, one for each label,



- For $\tilde{\mathbf{x}}$, predict

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

Binary Relevance (BR)

- Train L independent models $\mathbf{h} = (h_1, \ldots, h_L)$, one for each label,

Table : Binary Relevance: Model $\hat{y}_3 = h_3(\tilde{\mathbf{x}})$

X_1	X_2	X_3	$\ldots$	X_D	Y_1	Y_2	Y_3	Y_4
x_1	x_2	x_3	$\ldots$	x_D	0	1	1	0
x_1	x_2	x_3	$\ldots$	x_D	1	0	0	0
x_1	x_2	x_3	$\ldots$	x_D	0	1	0	0
x_1	x_2	x_3	$\ldots$	x_D	0	0	1	1
$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	$\ldots$	$\tilde{x}_D$			?	

- For $\tilde{\mathbf{x}}$, predict

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

Binary Relevance (BR)

- Train L independent models $\mathbf{h} = (h_1, \ldots, h_L)$, one for each label,

Table : Binary Relevance: Model $\hat{y}_3 = h_3(\tilde{\mathbf{x}})$

X_1	X_2	X_3	$\ldots$	X_D	Y_1	Y_2	Y_3	Y_4
x_1	x_2	x_3	$\ldots$	x_D	0	1	1	0
x_1	x_2	x_3	$\ldots$	x_D	1	0	0	0
x_1	x_2	x_3	$\ldots$	x_D	0	1	0	0
x_1	x_2	x_3	$\ldots$	x_D	0	0	1	1
$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	$\ldots$	$\tilde{x}_D$			?	

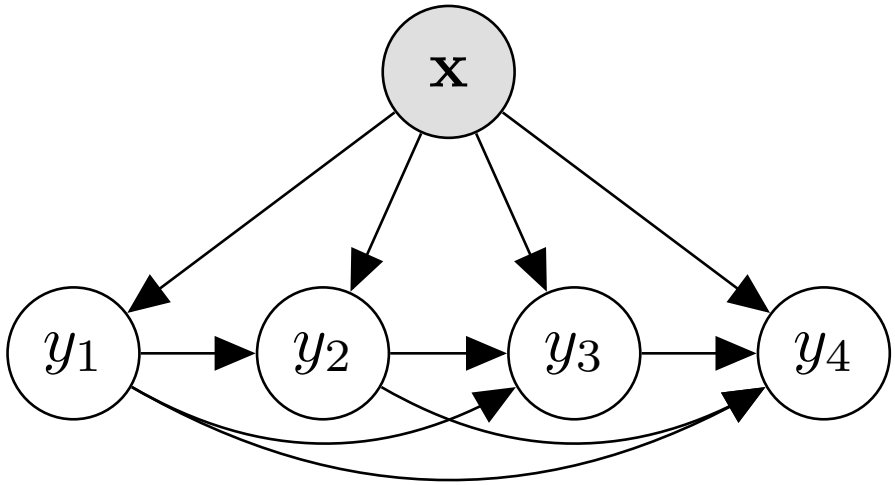
- For $\tilde{\mathbf{x}}$, predict

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

Consensus in the literature: should model relationship between labels!

Classifier Chains (CC)

- Predictions are cascaded along a chain as additional features



- For $\tilde{\mathbf{x}}$, predict

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

Classifier Chains (CC)

- Predictions are cascaded along a chain as additional features

Table : Classifier Chains: Model $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2)$

X_1	X_2	X_3	...	X_D	Y_1	Y_2	Y_3	Y_4
x_1	x_2	x_3	...	x_D	0	1	1	0
x_1	x_2	x_3	...	x_D	1	0	0	0
x_1	x_2	x_3	...	x_D	0	1	0	0
x_1	x_2	x_3	...	x_D	0	0	1	1
$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	...	$\tilde{x}_D$	$\hat{y}_1$	$\hat{y}_2$	?	

- For $\tilde{\mathbf{x}}$, predict

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

Classifier Chains (CC)

- Predictions are cascaded along a chain as additional features

Table : Classifier Chains: Model $\hat{y}_3 = h_3(\tilde{\mathbf{x}}, \hat{y}_1, \hat{y}_2)$

X_1	X_2	X_3	...	X_D	Y_1	Y_2	Y_3	Y_4
x_1	x_2	x_3	...	x_D	0	1	1	0
x_1	x_2	x_3	...	x_D	1	0	0	0
x_1	x_2	x_3	...	x_D	0	1	0	0
x_1	x_2	x_3	...	x_D	0	0	1	1
$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	...	$\tilde{x}_D$	$\hat{y}_1$	$\hat{y}_2$	?	

- For $\tilde{\mathbf{x}}$, predict

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

Typically better performance than BR, similar running time

Development of Classifier Chains

If we change the order of labels, predictive performance is different.

- Use the 'default' chain
- Use several random chains in ensemble
- Use chains based on performance (good accuracy, but *expensive*)
- Order the chain according to some heuristic,

Development of Classifier Chains

If we change the order of labels, predictive performance is different.

- Use the 'default' chain
- Use several random chains in ensemble
- Use chains based on performance (good accuracy, but *expensive*)
- Order the chain according to some heuristic, e.g., such as

label dependence!

Improvements to Classifier Chains

Measure dependence via

- empirical co-occurrence frequencies, pruned frequency counts, correlation coefficient, mutual information, tested with statistical significance tests, maximum spanning tree algorithm, probabilistic graphical model software, dependence among errors (to get conditional label dependence), test different chain orders with hold-out set / internal cross validation

and search the chain space with

- Monte Carlo search, simulated annealing, beam search, A* search

and then create,

- fully-cascaded classifier chains, population of chains, partially-connected chains, singly-linked chains, trees, directed graphs, undirected graphs, undirected chains, ensemble of trees, ensemble of graphs

Improvements to Classifier Chains

Measure dependence via

- empirical co-occurrence frequencies, pruned frequency counts, correlation coefficient, mutual information, tested with statistical significance tests, maximum spanning tree algorithm, probabilistic graphical model software, dependence among errors (to get conditional label dependence), test different chain orders with hold-out set / internal cross validation

and search the chain space with

- Monte Carlo search, simulated annealing, beam search, A* search

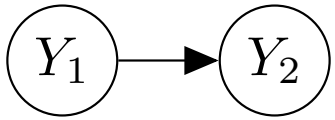
and then create,

- fully-cascaded classifier chains, population of chains, partially-connected chains, singly-linked chains, trees, directed graphs, undirected graphs, undirected chains, ensemble of trees, ensemble of graphs

Performance still $\approx$ ensemble of random chains (or too expensive)

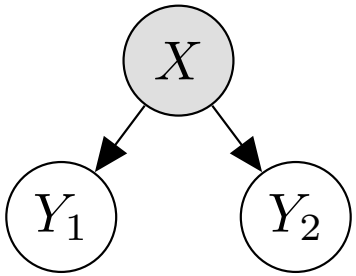
Classifier Chains: Label Dependence

We may find that . . .



$$\mathbb{E}(Y_2|Y_1) = \mathbb{E}(Y_2)$$

But if, *given the input*, the labels are *independent*,



$$\mathbb{E}(Y_2|Y_1, X) = \mathbb{E}(Y_2|X)$$

then independent classifiers (BR) $\equiv$ classifier chains (CC)?

Example: The XOR Problem

Toy problem,

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

Clearly, $\mathbb{E}(Y_3|Y_1, Y_2, X_1, X_2) = \mathbb{E}(Y_3|X_1, X_2)$, but ...

Table : Results: 20 examples, $h_j :=$ logistic regression, default label order.

Measure	BR	CC
Hamming acc.	0.83	1.00
Exact match	0.50	1.00

Label Dependence → Chain Structure?

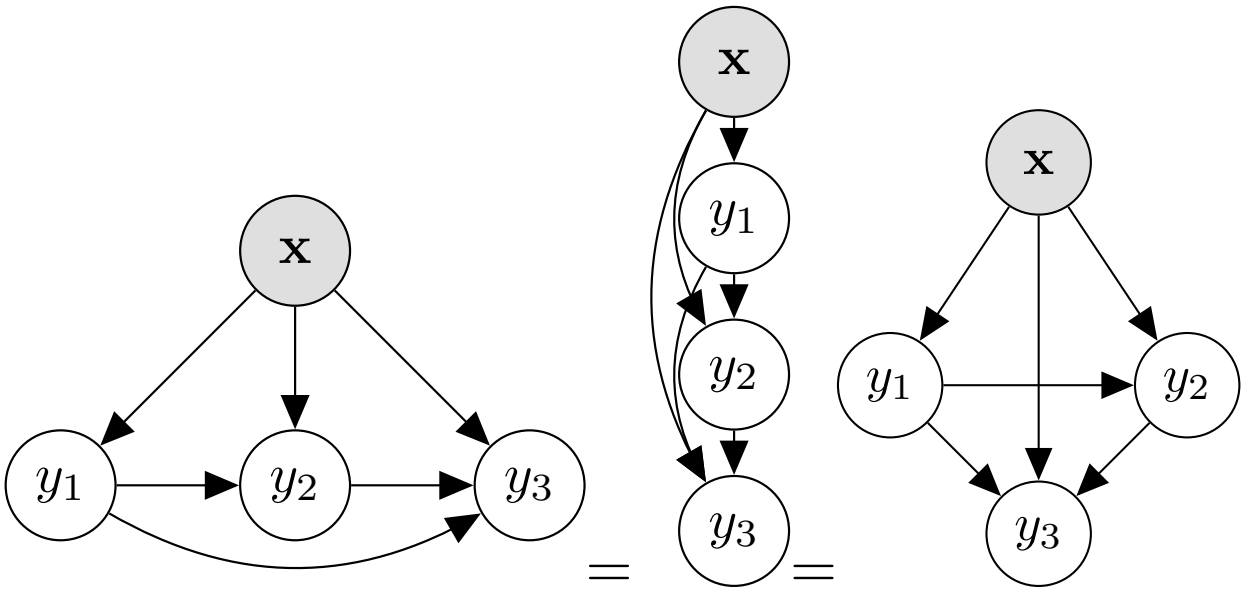
- The optimal structure / order is *likelihood dependent* – the one which performs best!
- Just looking at the labels is not enough
- We could model dependence between *errors*, rather than labels,

$$[\epsilon_1, \epsilon_2, \ldots, \epsilon_L] = [(y_1 - h_1(\mathbf{x}))^2, \ldots, (y_L - h_L(\mathbf{x}))^2]$$

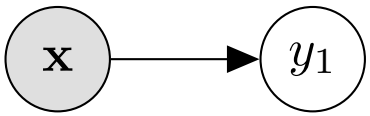
  (to take into account the input) but
  - we have to train **h**
  - pairwise only (or expensive), and
  - does not inform us of directionality!
- We could use an *undirected* graph, but
  - then we lose greedy inference.

Classifier Chains as a Neural Network

From the point of view of Y_3 (XOR),

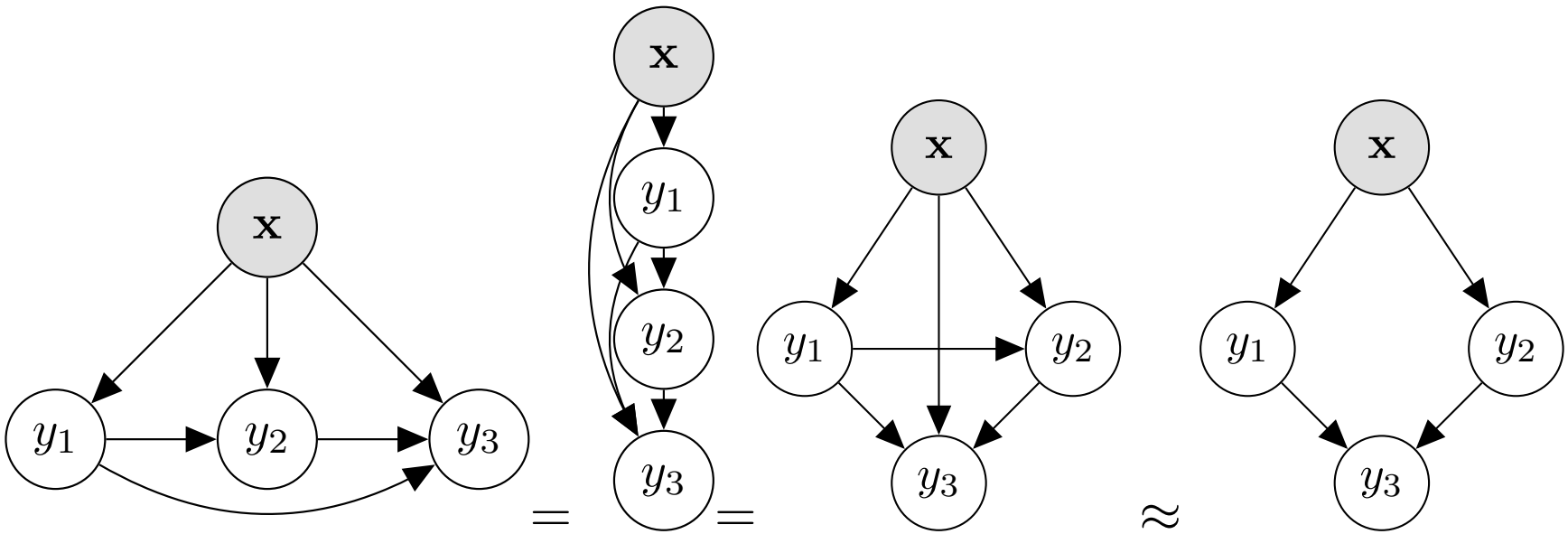


- $\approx$ A 'hidden' layer / feature space projection!
- does not work if we swap Y_3 (XOR) with Y_1 (OR)

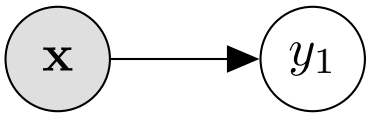


Classifier Chains as a Neural Network

From the point of view of Y_3 (XOR),



- ≈ A 'hidden' layer / feature space projection!
- does not work if we swap Y_3 (XOR) with Y_1 (OR)

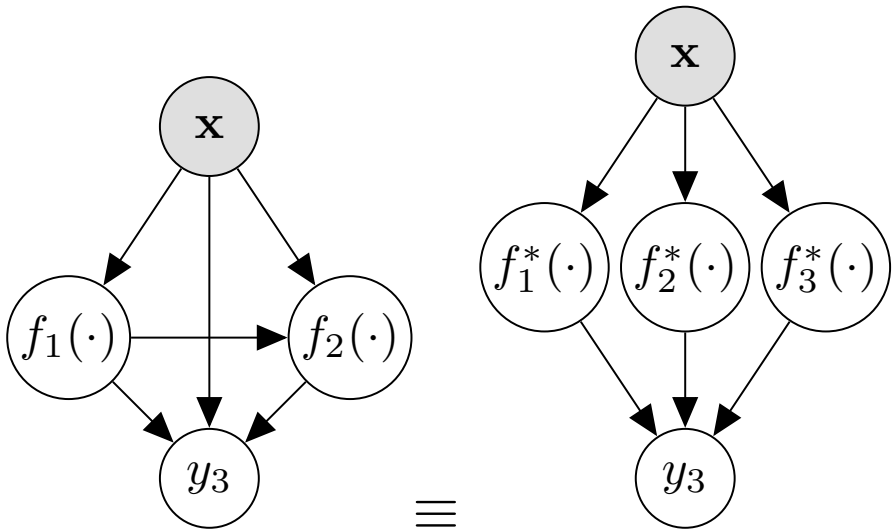


- The third graph is enough for all labels!

Labels are Transformations of the Input

- Labels are transformations of the input, which we learn from the training data. For example,

$$\begin{aligned}\hat{y}_2 &= f_2^*(\mathbf{x}) \\ &= \sigma(\mathbf{w}^\top \phi) \\ &= \sigma(\mathbf{w}^\top [x_1, \ldots, x_D, h(\mathbf{x})]\end{aligned}$$

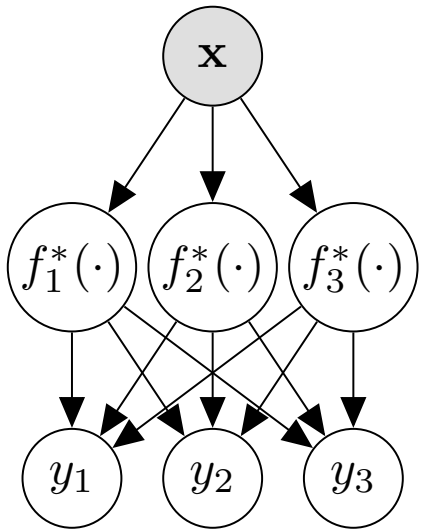


- For some $f_k^*(\mathbf{x}), \ldots, f_K^*(\mathbf{x})$, any label could be learned separately.

Labels are Transformations of the Input

- Labels are transformations of the input, which we learn from the training data. For example,

$$\begin{aligned}\hat{y}_2 &= f_2^*(\mathbf{x}) \\ &= \sigma(\mathbf{w}^\top \boldsymbol{\phi}) \\ &= \sigma(\mathbf{w}^\top [x_1, \ldots, x_D, h(\mathbf{x})]\end{aligned}$$



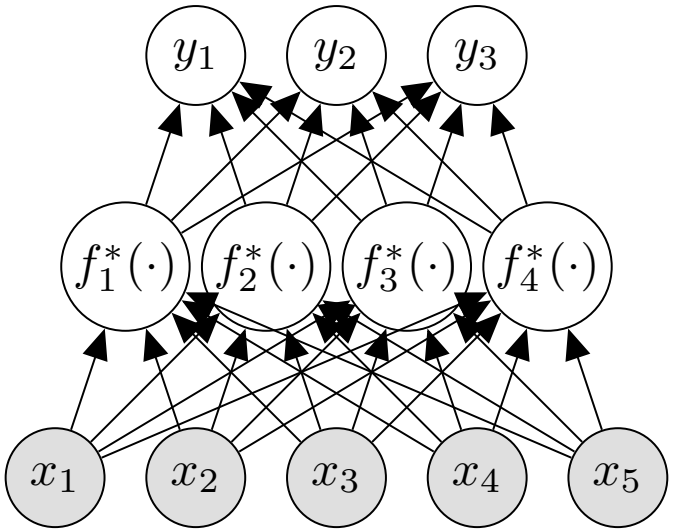
- For some $f_k^*(\mathbf{x}), \ldots, f_K^*(\mathbf{x})$, any label could be learned separately.

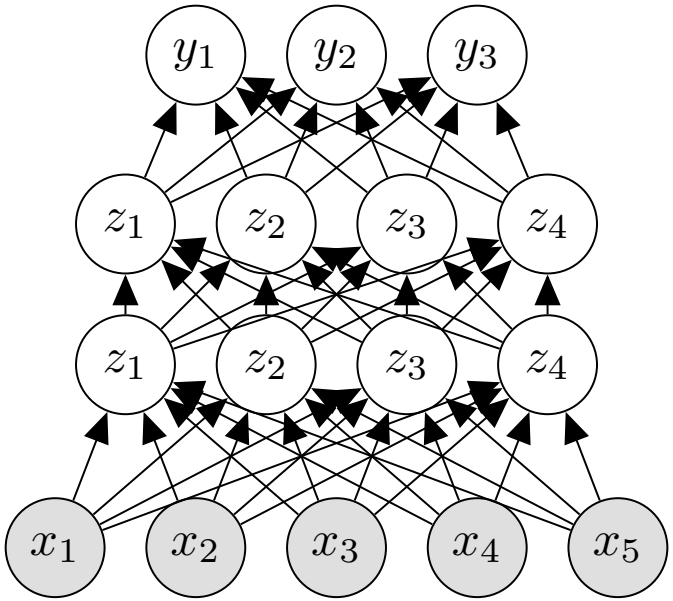
A Deep Neural Network

What if we replace labels with hidden units? If we

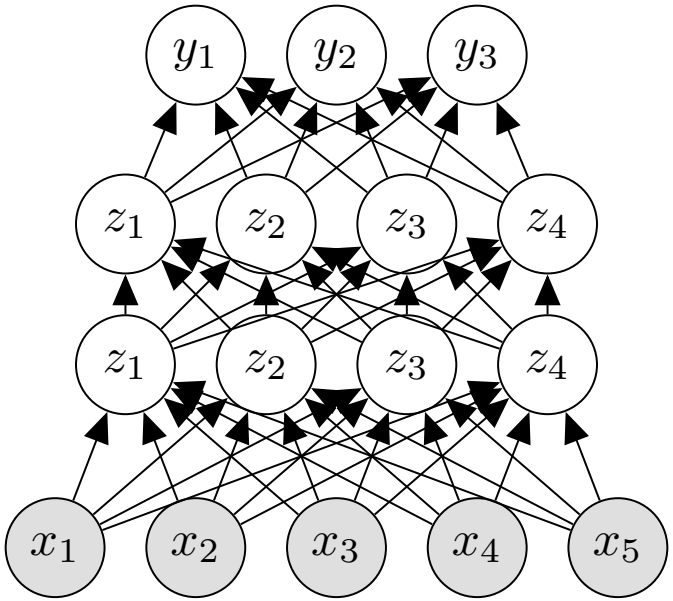
1. (expand **x**, and invert the order of layers) and
2. make linear units,

then we are looking at a deep neural network,





Learning Hidden Units



We can learn the hidden layers $\mathbf{z}^{[1]}$ and $\mathbf{z}^{[2]}$ with Restricted Boltzmann Machines (for example), and then

- plug in back propagation; or
- plug in a binary relevance learner

$$\hat{\mathbf{y}} = \mathbf{h}(\mathbf{z}^{[2]}) = \mathbf{h}(\mathbf{f}^{*[2]}(\mathbf{f}^{*[1]}(\tilde{\mathbf{x}})))$$

Results

Dataset	BR	·	CC	D·CC	D·BR	DBP
music	0.193 (6)	·	0.208 (5)	0.218 (4)	0.267 (2)	0.287 (1)
scene	0.286 (6)	·	0.353 (4)	0.476 (3)	0.582 (1)	0.183 (7)
yeast	0.150 (5)	·	0.198 (2)	0.204 (1)	0.149 (6)	0.179 (3)
genbase	0.960 (3)	·	0.965 (1)	0.965 (1)	0.950 (5)	0.950 (5)
medical	0.439 (4)	·	0.474 (2)	0.361 (5)	0.200 (6)	0.521 (1)
enron	0.022 (5)	·	0.028 (4)	0.161 (1)	0.054 (2)	0.043 (3)
avg. rank	4.83	·	3.00	2.50	3.67	3.33

D·**h** = Deep Structure + **h**-model on final layer (SVM as base classifier).

- BR performs poorly (as usual)
- . . . but not if we learn higher-level features first! (D·BR, DBP)
- This structure + CC (D·CC) performs best of all

Results

Dataset	BR	·	CC	D·CC	D·BR	DBP
music	0.193 (6)	·	0.208 (5)	0.218 (4)	0.267 (2)	0.287 (1)
scene	0.286 (6)	·	0.353 (4)	0.476 (3)	0.582 (1)	0.183 (7)
yeast	0.150 (5)	·	0.198 (2)	0.204 (1)	0.149 (6)	0.179 (3)
genbase	0.960 (3)	·	0.965 (1)	0.965 (1)	0.950 (5)	0.950 (5)
medical	0.439 (4)	·	0.474 (2)	0.361 (5)	0.200 (6)	0.521 (1)
enron	0.022 (5)	·	0.028 (4)	0.161 (1)	0.054 (2)	0.043 (3)
avg. rank	4.83	·	3.00	2.50	3.67	3.33

D·**h** = Deep Structure + **h**-model on final layer (SVM as base classifier).

- BR performs poorly (as usual)
- . . . but not if we learn higher-level features first! (D·BR, DBP)
- This structure + CC (D·CC) performs best of all

With the right features, we can use BR even with a linear learner

Why

- Mechanism behind classifier chains not well understood
- Investment in improvements to classifier chains not being rewarded
- There are disadvantages to classifier chains:
  - complexity with many labels
  - what structure/directionality to use?
  - inflexible (difficult to add/remove labels)

Reflections / Conclusions

- Classifier chains work as a 'deep' structure – other labels are 'supervised features'
- We can use binary relevance if final-layer inputs are independent of each other
  (we did this using multiple layers of feature transforms)
- This has advantages, such as semi-supervised learning, easier to add and drop labels

A Deep Interpretation of Classifier Chains

Jesse Read and Jaakko Holmén

http://users.ics.aalto.fi/{jesse,jhollmen}/

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

Leuven, Belgium. Fri. 31st Oct., 2014