

Label-Dependence in Multi-label Learning: A Fresh Look

Jesse Read



September 19, 2021, Grenoble.

**Multi-Label Learning: Current Trends and Open Challenges
(MLLCTOC – An ECML PKDD Workshop)**

Outline

- 1 Introduction
- 2 Why Model Labels Together: A Fresh Look
- 3 Model-based Model-Agnostic Transfer Learning

Introduction

- 1 Introduction
- 2 Why Model Labels Together: A Fresh Look
- 3 Model-based Model-Agnostic Transfer Learning

Multi-label Classification

Multi-label classification: a subset of labels may be assigned to each input instance.



Input	Beach	Sunset	Foliage	Urban
	1	0	1	0
	0	1	0	0
	0	1	0	1
	0	1	1	0
	0	0	1	1
	?	?	?	?

$y = [1, 0, 1, 0]$ \Leftrightarrow labels {Beach, Foliage} are relevant to x .

Often read in the literature:

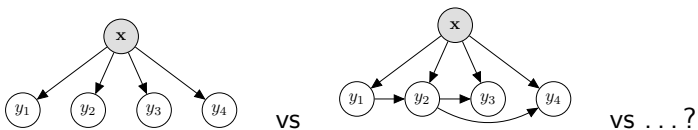
We model and predict labels together due to label dependence.

Often read in the literature:

We model and predict labels together due to label dependence.

And many empirical results appear to confirm this.

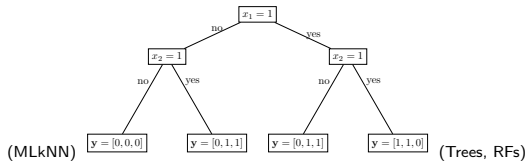
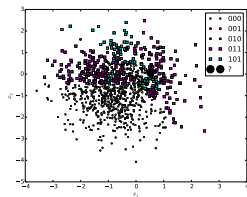
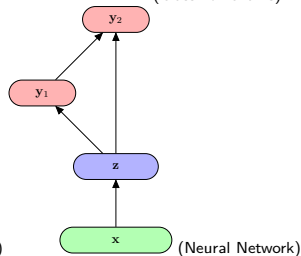
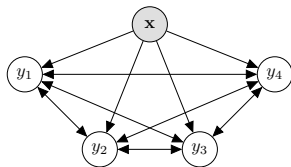
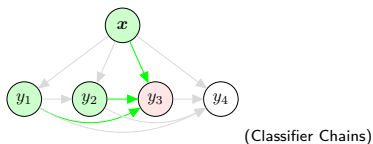
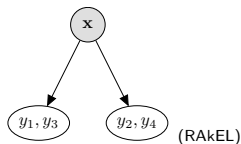
But what is the mechanism? When does it hold? ... Why?



A View/Timeline of Multi-label Learning in Academia

- < 2000s Just use independent models.
- ... 2010 **Model labels together**, based on label dependence/co-occurrences.
- ... 2015 Keep using label dependence, but in a more sophisticated/efficient way now.
- ... 2015 Multi-label learning for image, text, forecasting, recommendation, audio, health applications, distilling wine ...
- 2020 [... and for covid19].
- ... 2020 Just use independent models.
- ... 2020 Just use deep [convolution / recurrent] neural networks.
- 2020 deep [graph-embedding / residual / adversarial / transformer/...] neural networks.
- 2020 ... with [missing / partial / fast / incremental / weak / zillions of/...] labels.

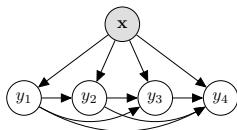
Multi-label Classifiers: Examples



Classifier Chains: An Example of 'Problem Transformation'

A chain (**structure**) over the output variables;

- Cascaded prediction across a chain/graph
- Motivation: Model label dependence

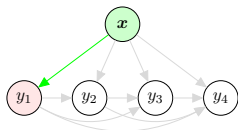


X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	1
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	1
$x^{(4)}$	1	0	0	0
$x^{(5)}$	0	0	0	0
\tilde{x}	\hat{y}_1	\hat{y}_2	\hat{y}_3	\hat{y}_4

Classifier Chains: An Example of 'Problem Transformation'

A chain (**structure**) over the output variables;

- Cascaded prediction across a chain/graph
- Motivation: Model label dependence



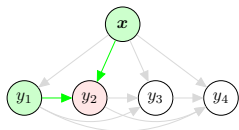
X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	1
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	1
$x^{(4)}$	1	0	0	0
$x^{(5)}$	0	0	0	0

\tilde{x}	\hat{y}_1			
-------------	-------------	--	--	--

Classifier Chains: An Example of 'Problem Transformation'

A chain (**structure**) over the output variables;

- Cascaded prediction across a chain/graph
- Motivation: Model label dependence



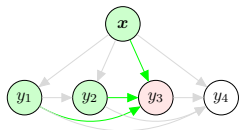
\mathbf{X}	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	1
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	1
$\mathbf{x}^{(4)}$	1	0	0	0
$\mathbf{x}^{(5)}$	0	0	0	0

$\tilde{\mathbf{x}}$	\hat{y}_1	\hat{y}_2		
----------------------	-------------	-------------	--	--

Classifier Chains: An Example of 'Problem Transformation'

A chain (**structure**) over the output variables;

- Cascaded prediction across a chain/graph
- Motivation: Model label dependence



X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	1
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	1
$x^{(4)}$	1	0	0	0
$x^{(5)}$	0	0	0	0
\tilde{x}	\hat{y}_1	\hat{y}_2	\hat{y}_3	

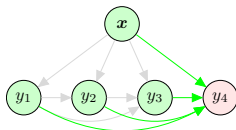
For example, $\hat{y}_3 = h_3(x, \hat{y}_1, \hat{y}_2)$ with **base classifier** (or regressor) h_3 (e.g., decision tree, logistic regression, ...).

Typical example of a "problem transformation" (or model agnostic) meta method that **works well** vs independent models

Classifier Chains: An Example of 'Problem Transformation'

A chain (**structure**) over the output variables;

- Cascaded prediction across a chain/graph
- Motivation: Model label dependence



X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	1
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	1
$x^{(4)}$	1	0	0	0
$x^{(5)}$	0	0	0	0

\tilde{x}	\hat{y}_1	\hat{y}_2	\hat{y}_3	\hat{y}_4
-------------	-------------	-------------	-------------	-------------

For example, $\hat{y}_3 = h_3(x, \hat{y}_1, \hat{y}_2)$ with **base classifier** (or regressor) h_3 (e.g., decision tree, logistic regression, ...).

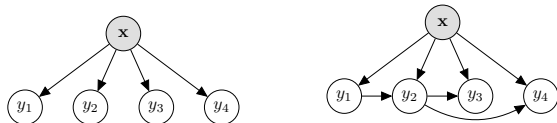
Typical example of a "problem transformation" (or model agnostic) meta method that **works well** vs independent models – **but why?**

Why Model Labels Together: A Fresh Look

- 1 Introduction
- 2 Why Model Labels Together: A Fresh Look
- 3 Model-based Model-Agnostic Transfer Learning

Suggestion 1: Because Label Dependence

Argument: If label variables are correlated/interdependent, we should model/predict them together; accuracy will better.



Maybe, **under metrics that require joint modelling**¹ (e.g., **exact match** accuracy). But there is more to this story! Consider:

- Dozens of methods improving on independent models under **Hamming loss** which *does not require* joint modelling
- Many methods improve their results with different node orderings yet *dependence is symmetrical!*
- Efforts to model label dependence correlate only weakly with accuracy. No one has found the best structure.

¹Dembczyński, Waegeman, and Hüllermeier, "An Analysis of Chaining in Multi-Label Classification", 2012

Taking a Step Back: What is Label Dependence?

Label dependence:

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

coincides with the intuition "beach and sunset may **co-occur** frequently" but also: the beach may indicate no urban, no beach may indicate no sunset (**mutual exclusivity**), etc.

Very rare cases: no correlation or perfect correlation measurable.

But in multi-label classification we are specifically interested in **conditional dependence** between Y_1 and Y_2 *having observed* x .

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

We **never know** P . There are many ways to estimate it.

Intuition conditional *independence*: I know a sunset when I see it.

Label Dependence and Loss Metrics

Suppose conditional dependence,

$$P(Y_1, Y_2|\mathbf{x}) \neq P(Y_1|\mathbf{x})P(Y_2|\mathbf{x})$$

y_1	y_2	$P(y_1, y_2 x)$
0	0	0.0
0	1	0.5
1	0	0.5
1	1	0.0

The question is not just of dependence, but of **uncertainty**.

Hamming similarity (or loss) (can **target labels independently**):

$$:= 0.5 \cdot \mathbb{1}[y_1 = \hat{y}_1] + 0.5 \cdot \mathbb{1}[y_2 = \hat{y}_2]$$

\Rightarrow Predict $\hat{y} = [0, 0]$ or $\hat{y} = [1, 1]!$

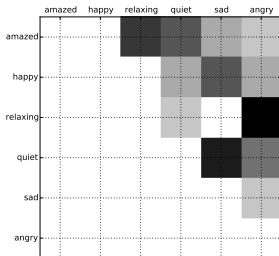
Exact match accuracy (or 0/1 loss) (need **label dependence**):

$$:= \mathbb{1}[\mathbf{y} = \hat{\mathbf{y}}] = \mathbb{1}[\text{Hamming similarity} = 1]$$

\Rightarrow Predict $\hat{y} = [1, 0]$ or $\hat{y} = [0, 1]!$

Modelling Label Dependence

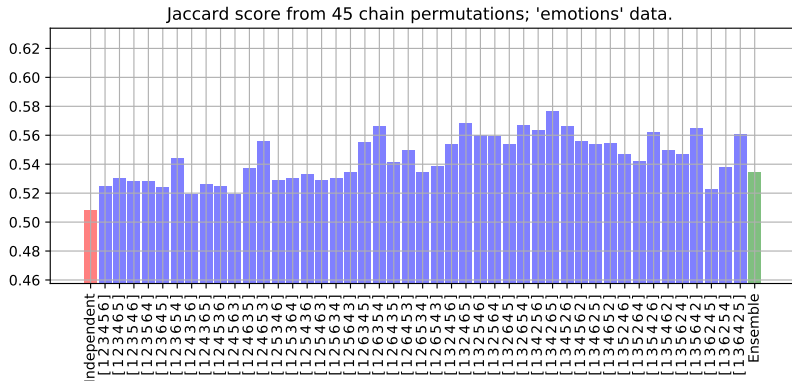
Attractive idea: 1) detect dependence, 2) form a structure.



Good news: Accuracy probably better than independent models.

Bad news: But not necessarily. Random structure also better. No single best structure. Would be impossible to find anyway (we don't have true P). Actually even with P , we cannot find it. Even if we do, it is only specific (optimal) to certain loss metric, base classifiers, parametrization, test instance, And it's all very expensive.

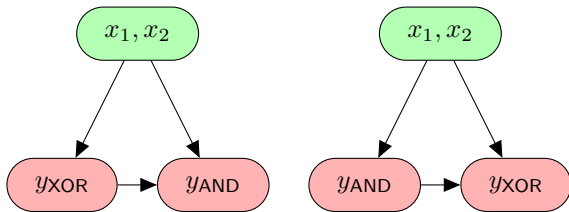
Case in point: trial *all structures* that fully model $P(Y_1, \dots, Y_6 | \mathbf{x})$, i.e., full label dependence (chain rule).



Indeed - modelling label dependence *gives better results* (in this case)! **But why so different?** There is very little correlation between structure and accuracy.

Even if I have the *true P*, using Hamming loss, ...

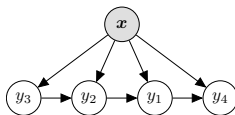
X_1	X_2	Y_{XOR}	Y_{AND}
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1



results may vary (consider: [base classifier](#), [inference](#), ...).

Suggestion 1b: Avoiding Error Propagation

Argument: There may be **error propagation** across the structure, so we should, e.g., **put easy labels first**.



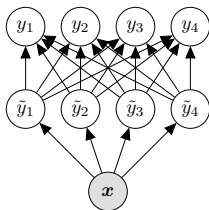
But

- Empirically: *Incorrect* label predictions may also *increase* the accuracy of *other* label predictions!
- Observation x is available at each step (error should *not* propagate!²)

²In the context of forward pass/greedy inference; a more complete discussion in Senge, Coz, and Hüllermeier, "On the Problem of Error Propagation in Classifier Chains for Multi-label Classification", 2014

Suggestion 1c: Correcting Predictions

Argument: We can 'correct' errors at prediction time, e.g., via **stacking**.

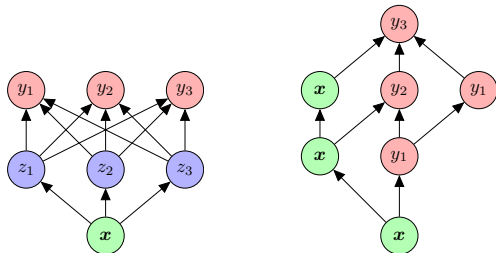


Yes (maybe). But

- $P(y_1 | \tilde{y}_1, \tilde{y}_2, \tilde{y}_3, \tilde{y}_4) \neq P(y_1, y_2, y_3, y_4 | x)$, i.e., this is **not label dependence modelling**, we only correct bias of individual models;
- involves a separate training mechanism for each layer.

Suggestion 2: Structure is acting like a Neural Network

Argument: Structure among labels \Rightarrow 'deep' 'neural' network.



Classifiers as activation/[transfer functions](#), labels as [hidden nodes](#).
A bit like ResNets. But:

- No back propagation (deep *prediction*, but not deep learning);
- the hidden nodes are not hidden.

Consider prediction task

$$\tilde{x} \mapsto \hat{y}_2$$

and the data available at training time (left) vs test time (right):

	X_1	X_2	X_3	Y_2	X_1	X_2	X_3	Y_2
Basis expansion	x	ϕ_1	ϕ_2	y_2	\tilde{x}	ϕ_1	ϕ_2	\hat{y}_2
Stacking	x	\tilde{y}_1	\tilde{y}_2	y_2	\tilde{x}	\tilde{y}_2	\tilde{y}_2	\hat{y}_2
Classifier chain	x	y_1		y_2	\tilde{x}	\hat{y}_1		\hat{y}_2
Neural network	x			y_2	\tilde{x}	\hat{z}_1	\hat{z}_2	\hat{y}_2

where $\phi_j \equiv \phi_j(x)$ (e.g., hand-coded/expert-designed **basis function**), $\tilde{y}_j = h_j(x)$ (trained on original dataset, then **stacked**), and y_j supplied directly in the original training set, and z_j are hidden units (not observed).

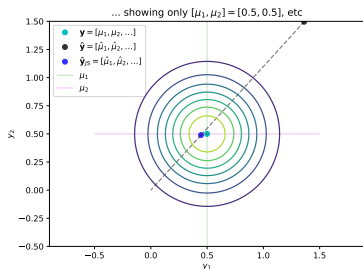
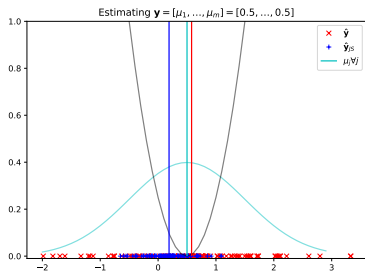
Suggestion 3: Structure is providing Regularisation

Argument: Modelling labels together provides better results **even if they are independent**.

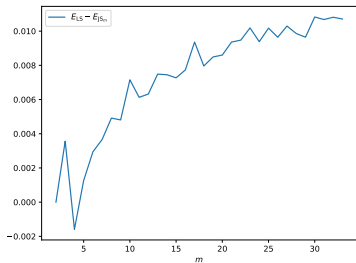
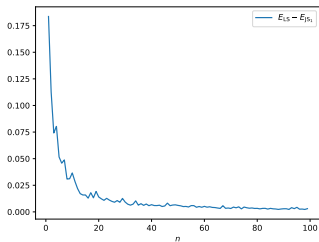
For example the James Stein estimator (for $m > 2$ labels, $y_j \in \mathbb{R}$):

$$\hat{\mathbf{y}}_{JS} = \frac{1 - (m - 2)\hat{\sigma}^2}{\|\hat{\mathbf{y}}\|^2} \hat{\mathbf{y}} = \lambda \cdot \hat{\mathbf{y}}$$

where λ *shrinks* (regularises) the max.-likelihood estimate $\hat{\mathbf{y}}$.

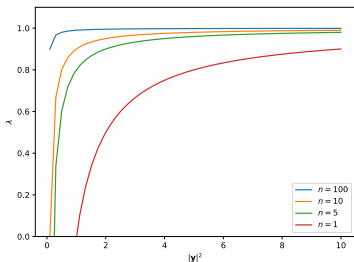


But: gains are minimal when $n \gg m$ (many examples, few labels):



(Showing the [Error of least squares - Error of James Stein estimator])

Indeed, the shrinkage factor λ loses strength quickly wrt n :

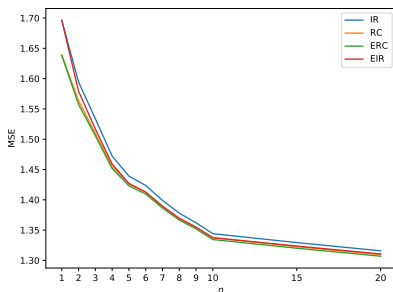


Suggestion 2b and 3b: The 'Ensemble Effect'

Argument: Modelling labels *appears* to provide better results but actually **the ensemble deserves the credit**, by providing

- More predictive power
- More regularisation, e.g.,

Ensembles of **independent models** on **independent labels** under mean squared error which **evaluates labels independently**; (and purely **linear concepts**):



i.e., ensemble having non-negligible benefit as a regulariser, this time for larger n !

So Which is it Then?

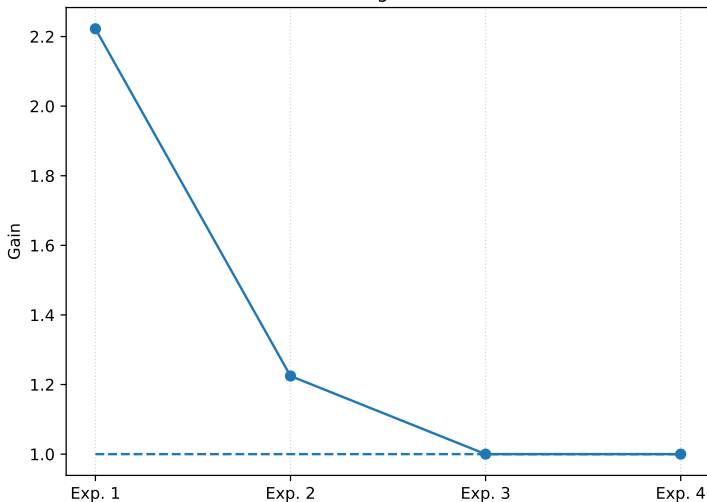
Goal: Isolate empirical evidence for such suggestions

Methodology: Progressively remove support for each 'Suggestion', measure accuracy Gain of **classifier chains** vs **independent models** (initially, base classifier = logistic regression):

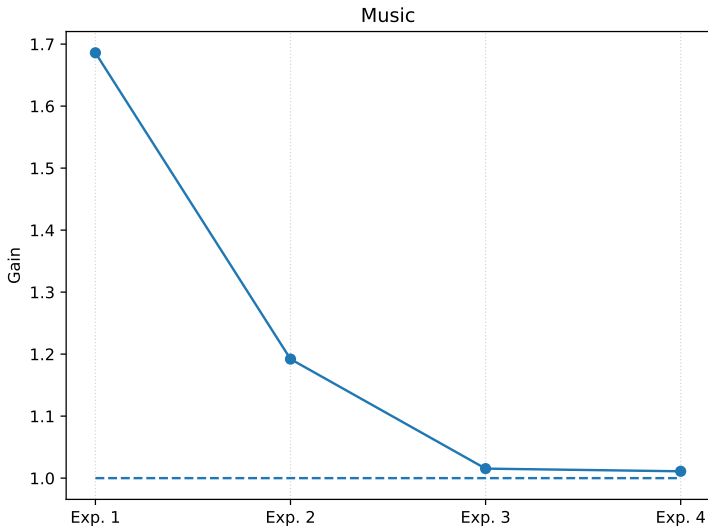
- Experiment 1 ('Suggestion 1'): under 0/1-loss, **dependence modelling** should shine! – i.e., gap should be significant
- Experiment 2 ('Suggestion 2 and 3'): ... switch to Hamming loss; i.e., *no* direct need for dependence modelling – any difference in accuracy must be **capacity vs regularisation!**
- Experiment 3 ('Suggestion 3'): now use multi-layer neural nets for non-linearity/extra capacity; i.e., any difference now must be **regularisation!**
- Experiment 4 ('Suggestion 0'): ... now heavily regularised neural net; if any significant difference now – I will be be confused!

And repeat many times to smooth out noise.

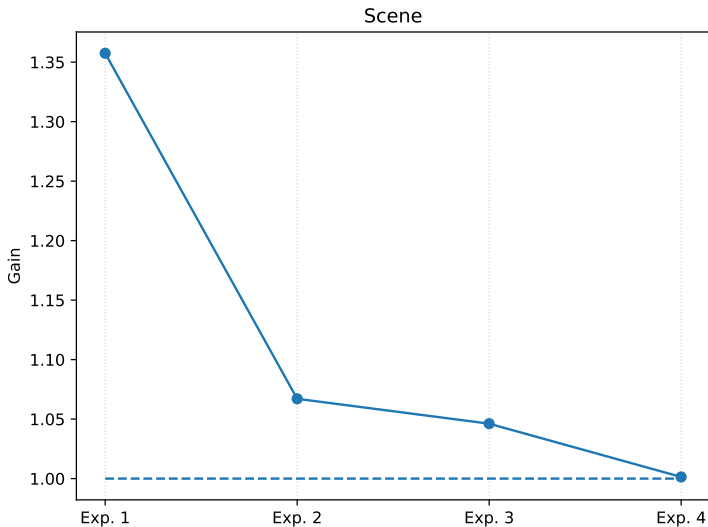
Logical



Conclusion: Modelling labels together is essential to capture label dependence, but a **significant gain** is obtained simply by the **extra capacity offered by the structure**.

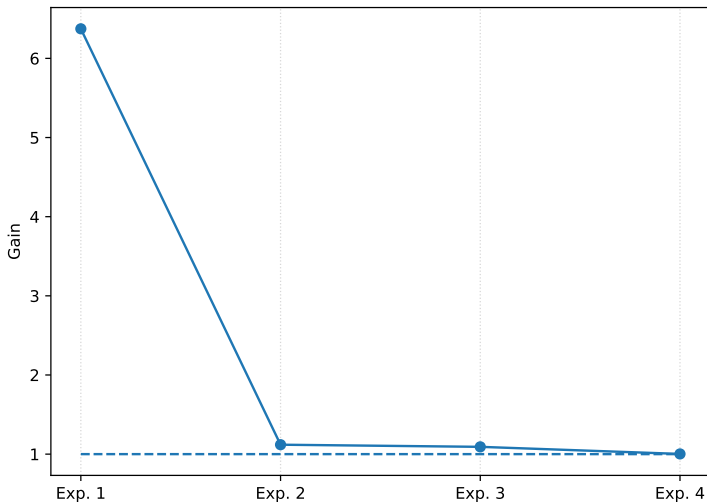


Conclusion: Modelling labels together is essential to capture label dependence, but a **significant gain** is obtained simply by the **extra capacity offered by the structure**.



Conclusion: Regularisation plays a non-negligible role, i.e., modelling **labels together simply to avoid overfitting!**

Yeast



Conclusion: Regularisation plays a non-negligible role, i.e., modelling **labels together simply to avoid overfitting!** N.B. Small, but non-negligible! Yeast has relatively high label density.

Lessons So Far

- ① We can offer a minor rephrasing:
*We should model and predict labels together **mainly** because of label dependence (i.e., **if our loss metric suggests that we need to learn it**), and we can also get **benefits from additional capacity and regularisation** brought by additional structure inherent to modelling labels together.*
- ② Answer to all multi-label problems = deep neural network architectures with standard regularised learning ... ?

Reasons to Retain Interest

Modelling labels together with model-agnostic/base-classifier approaches (and other algorithm-adaptations):

- still work well especially **on fewer training examples** (important for, e.g., small data and recovery from concept drift in data streams)
- require **no hidden units**; depth/non-linearity comes 'for free'
- requires **no back propagation**
- more choice (decision trees, including a mixture of different models, ...) – for reasons of **interpretability** or reliability; and

And we have shown several percentage points of **improvement from modelling totally unrelated tasks together**

Reasons to Retain Interest

Modelling labels together with model-agnostic/base-classifier approaches (and other algorithm-adaptations):

- still work well especially **on fewer training examples** (important for, e.g., small data and recovery from concept drift in data streams)
- require **no hidden units**; depth/non-linearity comes 'for free'
- requires **no back propagation**
- more choice (decision trees, including a mixture of different models, ...) – for reasons of **interpretability** or reliability; and

And we have shown several percentage points of **improvement from modelling totally unrelated tasks together** ← **very interesting!?**

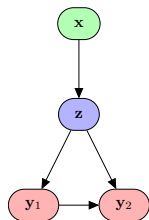
Model-based Model-Agnostic Transfer Learning

- 1 Introduction
- 2 Why Model Labels Together: A Fresh Look
- 3 Model-based Model-Agnostic Transfer Learning

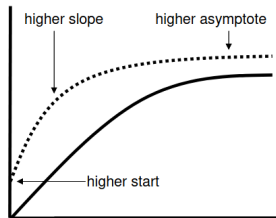
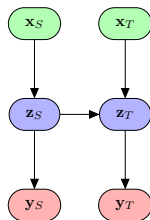
Transfer Learning

Quick guide to transfer learning:

- 1 Find related source task (S)
- 2 Use it to improve the model you deploy on target task (T)



VS

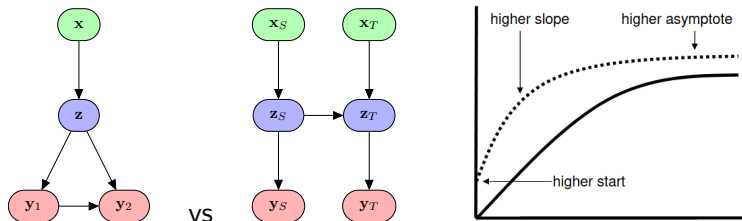


Plot (right) from Torrey and Shavlik, "Transfer learning", 2010.

Transfer Learning

Quick guide to transfer learning:

- 1 Find related source task (S)
- 2 Use it to improve the model you deploy on target task (T)



Plot (right) from Torrey and Shavlik, "Transfer learning", 2010.

A key word was: *related*. But **what if relatedness is not a requirement?**

Thoughts on That

Transfer learning by connecting the model from an unrelated source task. This is similar to connecting the first layer of a neural network randomly.

"[C]onnecting the first layer randomly is just about the stupidest thing you could do" – Yann LeCun

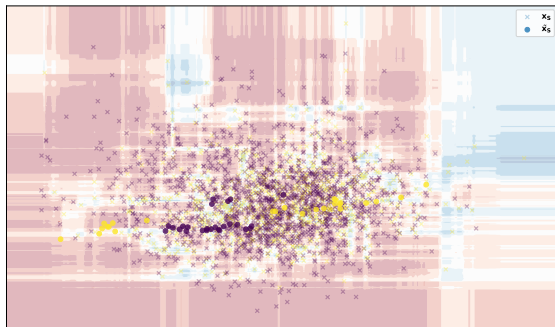
Remarks:

- He said "just about"
- He didn't say it didn't work
- Minor difference: In our case, not random in the sense of `randn()`, rather just totally unrelated

So let's try it anyway...

Insomniac Fungi

A model (random forest) for classifying patients³: suffering **insomnia** (red) or not (blue), based on sleep measurements $\mathbf{x}_S = [x_1, x_2]$:

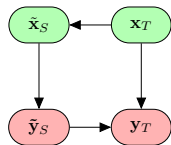
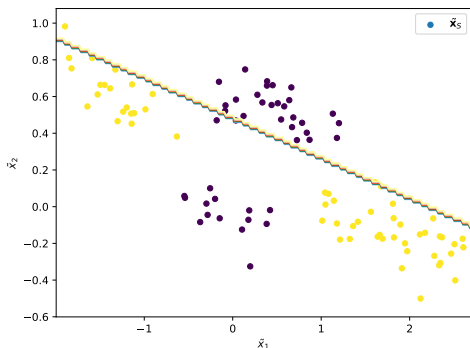


When a **yeast genome** is squeezed into $\tilde{\mathbf{x}}_S$, an insomnia diagnosis as an additional feature gives $+ \approx 2\%$ accuracy for predicting genome function.

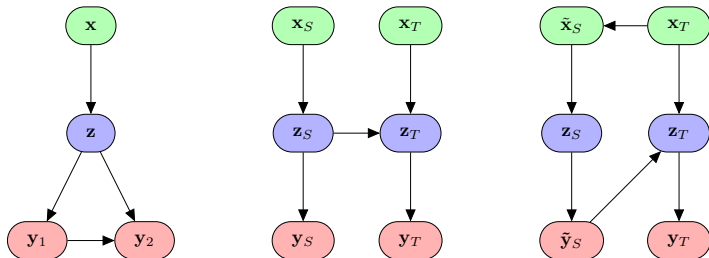
³Medical data thanks to Olivier Pallanca

Replicating on Synthetic Data

A target task XOR (data shown, some noise added) is solved via predictions from AND-function (decision boundary shown) as an additional feature:



Multi-Label Chain vs Deep Transfer vs Chain Transfer



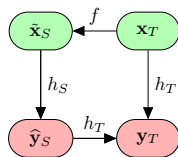
Main difference from Deep Transfer: A **model agnostic** approach; **require only outputs**.

A Random Projection or Manufacturing Dependence?

Not a random projection*, a randomly-chosen projection
 $h_S : \mathcal{X}_S \rightarrow \mathcal{Y}_S$ (target domain to source domain).

*Actually, we still need a projection $f : \mathcal{X}_S \rightarrow \mathcal{X}_T$ if different sizes.

Then feature selection/regularisation wrt h_T .

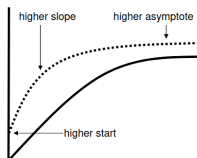


Implicit assumption: the source classifier h_S was/is useful for something (else).

Implicit challenge: information bottleneck.

More Experiments

Recall (from Torrey and Shavlik, "Transfer learning", 2010):



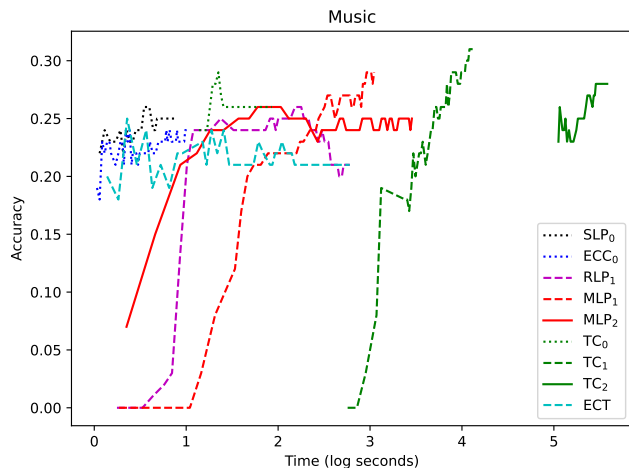
- MLP_ℓ : neural network, ℓ layers; $SLP_0 \equiv MLP_0$
- ECC: ensembles of classifier chains
- RLP: random-layer projection; \approx 'extreme learning machine'⁴
- TC: 'transfer chains'; from unrelated source classifiers
- ETC (ensembles of TC)

Subscripts indicate the number of hidden layers. At each step +100 iterations of gradient descent *or* (for ensemble methods) +1 model (with 100 iterations). Hyperparameters and capacity (nodes) roughly equivalent among layers ℓ .

What we're looking for: is TC at all not useless?

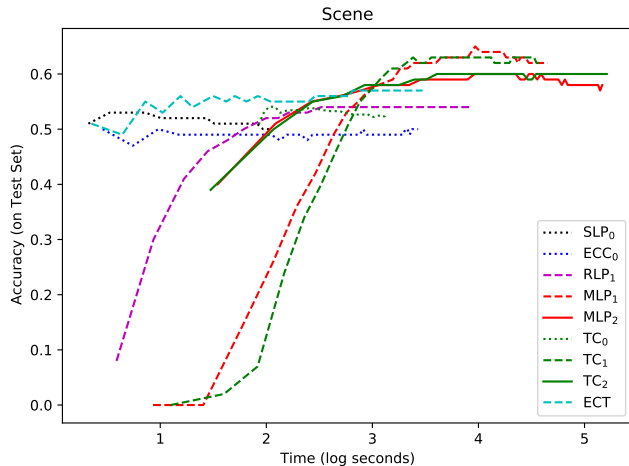
⁴Huang, Wang, and Lan, "Extreme learning machines: a survey", 2011

Results



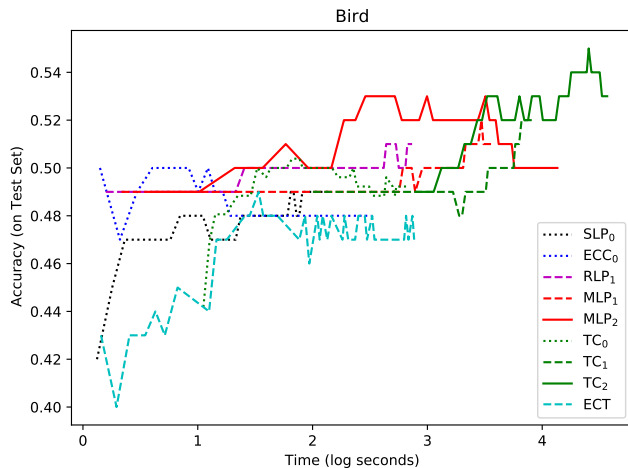
Music is a small dataset, transfer chains not efficient, but still powerful.

Results



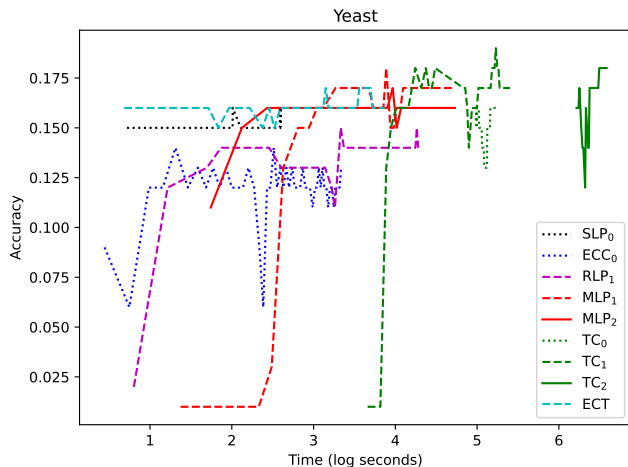
Scene is an image dataset, back-propagation sufficient, no benefit from transfer.

Results



Better results after transfer, same number of iterations.

Results



Learning starts a bit faster with transfer (step-wise). Random projections not so effective.

A Reflection on Results

- 'Transfer Chains' is not a state-of-the-art method.
- But it works – and that is interesting!
- In an extremely difficult transfer setting: no model introspection, no source data, no task dependence
- Standard neural network architectures worked best⁵
They cover all 'suggestions': **dependence-modelling**, **non-linearity** and **capacity**, and **regularisation**.
- Model-agnostic approaches offer these aspects too, but face usual limitations of learning without back-propagation;
Advantages: the depth without the need for deep learning – using other labels; and free choice of base model

⁵As seen in the Tutorial *Multi-Target Prediction with Deep Neural Network: A hands-on tutorial* by Iliadis and Waegeman this morning!

Looking Further Afield; Open Questions, Discussion Points

- High intersection with the deep learning community – multi-task learning, transfer learning, lifelong learning, . . . – ‘pretrained’ ‘frozen layers’ and ‘parameter isolation’, ‘universal computation’, . . .
- What does it mean for a label/task to be related to another?
- Transfer learning vs **reduction/reuse/recycling** of models?
- Transfer by analogy, rather than transfer of inner layers
- **A shift from data-driven learning** to **model-driven learning**?
- Immediate practical implications: **adapting to concept shift in data streams**; don't forget ‘irrelevant’ models!?

Conclusions

- Multi-label learning: important progress (and many methods) in recent decades and still relevant!
- It is more complex than modelling labels together (or not) *'because of label dependence'*
- More **explainability** on *how* methods work/predict
- Model labels/tasks together for other reasons than label dependence – and **even when there is 'none'!**
- Ever larger/more complex problems via data-driven learning 'from scratch' – increasingly challenging!
- The wider machine learning community is facing many of the same problems, let's continue to import their problems and also to contribute back our solutions!

Label-Dependence in Multi-label Learning: A Fresh Look

Jesse Read






Thank you!

<http://www.lix.polytechnique.fr/~jread/>

References I

This talk is based on preprint: *From Multi-label Learning to Cross-Domain Transfer: A Model-Agnostic Approach*, J. Read, 2022. <https://arxiv.org/pdf/2011.11197.pdf>
(many more references within!)

-  Bogatinovski, Jasmin et al. “Comprehensive comparative study of multi-label classification methods”. In: *Expert Systems with Applications* 203 (2022), p. 117215.
-  Dembczyński, Krzysztof, Willem Waegeman, and Eyke Hüllermeier. “An Analysis of Chaining in Multi-Label Classification”. In: *ECAI: European Conference of Artificial Intelligence*. Vol. 242. IOS Press, 2012, pp. 294–299. ISBN: 978-1-61499-097-0.
-  Dembczyński, Krzysztof et al. “On Label Dependence and Loss Minimization in Multi-label Classification”. In: *Mach. Learn.* 88.1-2 (July 2012), pp. 5–45. ISSN: 0885-6125. DOI: 10.1007/s10994-012-5285-8.

References II

-  Huang, Guang-Bin, DianHui Wang, and Yuan Lan. “Extreme learning machines: a survey”. English. In: *International Journal of Machine Learning and Cybernetics* 2.2 (2011), pp. 107–122. ISSN: 1868-8071. DOI: 10.1007/s13042-011-0019-y.
-  Read, Jesse et al. “Classifier Chains: A Review and Perspectives”. In: *Journal of Artificial Intelligence Research (JAIR)* 70 (2021), pp. 683–718. URL: <https://jair.org/index.php/jair/article/view/12376>.
-  —. “Classifier Chains for Multi-label Classification”. In: *ECML-PKDD 2009: 20th European Conference on Machine Learning*. Bled, Slovenia: Springer, 2009, pp. 254–269. URL: http://link.springer.com/chapter/10.1007%2F978-3-642-04174-7_17.

References III

-  Senge, Robin, Juan José del Coz, and Eyke Hüllermeier. “On the Problem of Error Propagation in Classifier Chains for Multi-label Classification”. In: *Data Analysis, Machine Learning and Knowledge Discovery*. Ed. by Myra Spiliopoulou, Lars Schmidt-Thieme, and Ruth Janning. Cham: Springer International Publishing, 2014, pp. 163–170. ISBN: 978-3-319-01595-8.
-  Torrey, Lisa and Jude Shavlik. “Transfer learning”. In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.
-  Waegeman, Willem, Krzysztof Dembczyński, and Eyke Hüllermeier. “Multi-target prediction: a unifying view on problems and methods”. In: *Data Mining and Knowledge Discovery* 33.2 (2019), pp. 293–324. ISSN: 1573-756X. DOI: 10.1007/s10618-018-0595-5. URL: <https://doi.org/10.1007/s10618-018-0595-5>.