A Deep Interpretation of Classifier Chains

Jesse Read and Jaakko Hollmén

Aalto University, Department of Information and Computer Science
PO Box 15400, FI-00076 Aalto, Espoo, Finland
Helsinki Institute for Information Technology (HIIT), Finland
{jesse.read,jaakko.hollmen}@aalto.fi

Abstract. In the "classifier chains" (CC) approach for multi-label classification, the predictions of binary classifiers are cascaded along a chain as additional features. This method has attained high predictive performance, and is receiving increasing analysis and attention in the recent multi-label literature, although a deep understanding of its performance is still taking shape. In this paper, we show that CC gets predictive power from leveraging labels as additional stochastic features, contrasting with many other methods, such as stacking and error correcting output codes, which use label dependence only as kind of regularization. CC methods can learn a concept which these cannot, even supposing the same base classifier and hypothesis space. This leads us to connections with deep learning (indeed, we show that CC is competitive precisely because it is a deep learner), and we employ deep learning methods – showing that they can supplement or even replace a classifier chain. Results are convincing, and throw new insight into promising future directions.

1 Introduction

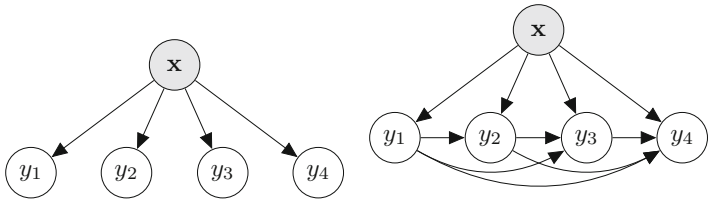
Multi-label classification (MLC) is the supervised learning problem where an instance is associated with multiple binary class variables (i.e., *labels*), rather than with a single class, as in traditional classification problems ([12]). The typical argument (which this paper reanalyzes) is that, since these labels are often strongly correlated, modeling the dependencies between them allows MLC methods to improve their performance.

As in general classification scenarios, an n-th feature vector (instance) can be represented as $\mathbf{x}^{(n)} = [x_1^{(n)}, \ldots, x_D^{(n)}]$, where each $x_d \in \mathbb{R} | d = 1, \ldots, D$. In the traditional *binary* classification task, we are interested in having a model h to provide a prediction for test instances $\tilde{\mathbf{x}}$, i.e., $\hat{y} = h(\tilde{\mathbf{x}})$; where h, probabilistically speaking, seeks the expectation $\mathbb{E}[\mathbf{y}|\mathbf{x}]$ of unknown $p(\mathbf{y}|\mathbf{x})$. In MLC, there are L binary output class variables (*labels*), and we are interested in predictions

$$\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_L] = h(\tilde{\mathbf{x}}) = \underset{\mathbf{y} \in \{0,1\}^L}{\operatorname{argmax}} \hat{p}(\mathbf{y}|\tilde{\mathbf{x}})$$

where $y_j = 1$ indicates the relevance of the j-th label; $j = 1, \ldots, L$.

H. Blockeel et al. (Eds.): IDA 2014, LNCS 8819, pp. 251–262, 2014.



(a) BR **(b)** CC

Fig. 1. BR (1a) and CC (1b) as graphical models, $L = 4$. Unlike many typical Bayesian networks, we have lumped $\mathbf{x} = [x_1, \ldots, x_D]$ into a single variable.

From N labelled examples (training data) $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$, we infer h. A most basic solution is to train L binary models. This method is called *binary relevance* (BR); illustrated graphically in Fig. 1a. BR classifies an $\tilde{\mathbf{x}}$ L times as $h_{\texttt{BR}}(\tilde{\mathbf{x}}) := [h_1(\tilde{\mathbf{x}}), \ldots, h_L(\tilde{\mathbf{x}})]$.

Practically the entirety of the multi-label literature points out that the independence assumption among the labels leads to suboptimal performance (e.g., [16,7,3,15,20] and references therein), and that for this reason BR cannot achieve optimal performance. A plethora of methods have been motivated by a perceived need to modelling this dependence and thus improve over BR. For example, Meta-BR (MBR, also known in the literature as 'stacked-BR' and '2BR') [7,3] stacks the output of one BR as input into a second (meta) BR[1], so as to learn to correct errors. For some $\tilde{\mathbf{x}}$,

$$h_{\texttt{MBR}}(\tilde{\mathbf{x}}) := h'_{\texttt{BR}}(h_{\texttt{BR}}(\tilde{\mathbf{x}}))$$

A related approach uses subset mapping (SM, e.g., as in [18]) to force infrequent label vector predictions to a more frequent ones,

$$h_{\texttt{SM}}(\tilde{\mathbf{x}}) := \underset{\mathbf{y} \in \mathcal{Y}_{\texttt{train}}}{\operatorname{argmin}} \ell(\mathbf{y}, h_{\texttt{BR}}(\tilde{\mathbf{x}}))$$

where $\mathcal{Y}_{\texttt{train}}$ are all distinct $\mathbf{y}^{(n)}$ from the training data $\mathcal{D}$ and $\ell(\mathbf{y}, \hat{\mathbf{y}})$ is some penalty function typically rewarding small Hamming distance and high frequency in $\mathcal{D}$. SM is very closely related to error-correcting output code methods [6] and, like MBR, can be seen as a regularizer. The penalty goes to ∞ if $\mathbf{y} \notin \mathcal{Y}_{\texttt{train}}$, meaning that any predictions of label combinations not seen in the training set will be 'corrected'.

As an example, movie genres adult and family may be mutually exclusive in the training set, and having a regularization/correction component to avoid this classification at test time may lead to improved performance (over BR).

The classifier chains method (CC, [16]), illustrated in Fig. 1b, models label dependence by using binary label predictions as extra input attributes for the following classifier, in a chain, and therefore models labels and inputs together,

[1] There is no consensus in the literature as to whether it is best to also include the $\mathbf{x}$-space input again, or simply the label outputs $h(\mathbf{x})$, as input to the meta BR.

rather than correcting labels as a separate step. In the original formulation with greedy inference,

$$h_{\texttt{CC}}(\tilde{\mathbf{x}}) := \Big[h_1(\tilde{\mathbf{x}}), h_2(\tilde{\mathbf{x}}, h_1(\tilde{\mathbf{x}})), \ldots, h_L(\tilde{\mathbf{x}}, h_1(\tilde{\mathbf{x}}), \ldots, h_{L-1}(\tilde{\mathbf{x}}))\Big].$$

CC variants have consistently performed strongly in the literature and there have been numerous extensions, variations and analyses, e.g., [2,20,15,11,5]. However, the reasons for its high performance are only recently being unravelled. In this paper, we throw new light on the subject.

Two focus points for improvement of CC have been the inference, and the order of the labels in the chain. Originally, [16] suggested an ensemble of randomly-ordered chains (ECC) with voting, whereas two recent high-performing CC methods [11] and [15], use beam and Monte Carlo search, respectively to obtain one well-ordered chain. We use the latter, which we denote MCC, in empirical comparisons, as well as ECC with 10 random chains.

2 Label Dependence in Multi-label Learning

The idea of leveraging label dependence to improve performance vs BR intuitively makes sense. However, the understanding behind this is only recently taking shape, with the authors of [2,5,4] opening an important discussion from a probabilistic perspective, noting the difference between

- marginal dependence, where $p(y_j|y_k) \neq p(y_j)$; and
- conditional dependence, where $p(y_j|y_k) \neq p(y_j|y_k, \mathbf{x})$.

Thus MBR and SM model marginal dependence, whereas CC models conditional dependence, by learning labels and input together.

An interesting point of debate is the following: given infinite data, can two *separate* binary models on labels y_j and y_k achieve as good performance as one that models them together (e.g., MBR, CC) – assuming the same base classifier (say, logistic regression)? Among others, [16,5] ponder if BR has been underrated and could equal CC's performance with enough training data. Indeed, [4] make the case that it should be possible make risk-minimizing predictions without any particular effort to detect or model label dependence. This seems to throw into doubt the bulk of the contributions to the multi-label literature.

It is also worth recalling here, that labels cannot only be learned together or separately, but also evaluated together or separately. A typical measure for the latter case is the Hamming score, which is widely used in MLC empirical evaluations. However, many MLC papers quietly overlook the fact that achieving statistically significant improvement over BR in this measure is difficult to obtain. This seems to add to [4]'s claim.

Proposition 1. *If we can predict* $\mathbb{E}(Y_2|Y_1, \mathbf{x})$ *to a certain degree of accuracy under some evaluation measure, then it is also possible to predict* $\mathbb{E}(Y_2|\mathbf{x})$ *with at least the same accuracy under the same measure.*

We elaborate on this in the following sections.

3 Analysis on Synthetic Datasets

Using the following synthetic datasets with the methods discussed above (Section 1), with logistic regression as a base classifier, we run some experiments to expand on the discussion from Section 2. Results are given in Tab. 1.

- Localization: a scenario (see Fig. 2) where labels correspond to pixels which represent floor tiles in a room, and are active/relevant ($y_j = 1$) if an object is on them. Light sensors signal detection (with 90% accuracy) $x_d = 1$ if an active tile lies between the sensor location and the light source. For each of 1000 instances (two thirds of which are used for training), a line of three tiles is activated ($y_j = 1$ for three j) in a random location in the grid, and another tile is activated in the furthest corner from that line; which is always a blind spot (undetectable by sensors). Based on the real-world deployment of [14].
- Logic: Two binary attributes, X_1, X_2, deterministically mapped to three labels Y_1, Y_2, Y_3, corresponding to AND(X_1, X_2), OR(X_1, X_2), XOR(X_1, X_2) (binary logical operations). We generate 20 X_1, X_2 randomly, and use 12 for training.

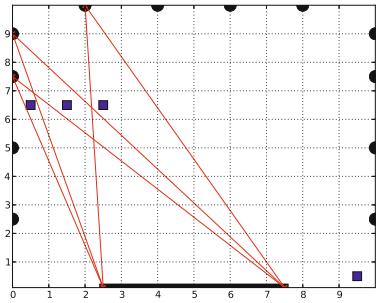


Fig. 2. Top-down view of a grid of L tiles in a room. There are D sensors, which signal detection ($x_d = 1$) with 90% probability if any of the active tiles ($y_j = 1$) lie between sensors (black semi circles) and the light source (black bar at the bottom). Here, three sensors have detected an object. Note the blind spot.

We use two standard, opposing evaluation methods,

$$\text{HAMMING SCORE} := \frac{1}{NL} \sum_{n=1}^{N} \sum_{j=1}^{L} \left[y_j^{(n)} = \hat{y}_j^{(n)} \right], \text{ and}$$

$$\text{EXACT MATCH} := \frac{1}{N} \sum_{n=1}^{N} \left[\mathbf{y}^{(n)} = \hat{\mathbf{y}}^{(n)} \right]$$

which are used in almost all MLC evaluations. HAMMING SCORE (which we mentioned in the previous section) rewards methods for predicting individual

Table 1. Predictive Performance on Toy Datasets with per-dataset (rank)

HAMMING SCORE

Dataset	BR	CC	SM	MBR	ECC	MCC
Localization	0.992 (1)	0.992 (1)	0.991 (4)	0.991 (4)	0.991 (4)	0.992 (1)
Logic	0.833 (5)	1.000 (1)	0.750 (6)	0.875 (3)	0.875 (3)	1.000 (1)

EXACT MATCH

Dataset	BR	CC	SM	MBR	ECC	MCC
Localization	0.412 (5)	0.491 (2)	0.455 (3)	0.408 (6)	0.447 (4)	0.497 (1)
Logic	0.500 (6)	1.000 (1)	0.625 (3)	0.625 (3)	0.625 (3)	1.000 (1)

labels well, whereas EXACT MATCH rewards a higher proportion of instances with *all* label relevances correct.

Given much of the discussion in the literature, one could understand that a method which models label dependence would excel on Localization. This could be claimed for EXACT MATCH; but the number of correct label relevances (HAMMING SCORE) is essentially identical across all methods. Despite the obvious label dependence here (wrt the position and shape of the 'relevant' pixels), conditioning on $\mathbf{x}$ (i.e., training BR) suffices for high label-wise precision.

The results on Logic indicate the opposite case: BR is clearly unable to learn the concept, even though we can be sure that $\mathbb{E}[y_3|y_2, y_1, \mathbf{x}] = \mathbb{E}[y_3|\mathbf{x}]$. Furthermore, although SM and MBR model label dependence, they cannot learn the concept either: their regularization cannot make up for the fact that their underlying BR models fail. CC and MCC score perfectly. ECC under-performs, so apparently CC only performs well under certain label orders (and got 'lucky'); confirmed by MCC, which finds one good order before final training. MBR does actually have a structure suitable for learning XOR, but apparently training cannot leverage it properly.

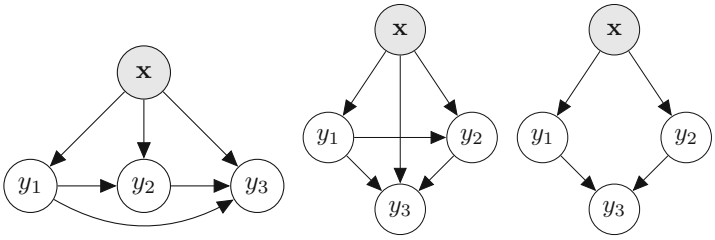
The authors of [5] uncovered a similar case with their *probabilistic classifier chains* (originally presented in [2], using Bayes-optimal inference instead of greedy inference for CC), putting it down to this method's "expanded hypothesis space" (it trials all 2^L combinations for $\hat{\mathbf{y}}$ at inference time). This, however, cannot be the complete answer, since the original CC makes L separate binary decisions just like BR; thus the same hypothesis space.

With real-world datasets it is difficult to postulate, but on the Logic dataset it is clear that BR's accuracy will *never* reach 1 even under infinite data, since its h_3 model will *never* learn XOR. The performance gap in Tab. 1 is a convincing 50 percentage points for EXACT MATCH. CC's h_3 is a perfect model, even with the same base classifier.

In the next section we explain how CC works as a deep structure, of up to L levels (let us simply state that any structure of more than one level is deep) and for this reason can outperform BR as well as the 'regularization'-type methods like MBR and SM.

4 Why Classifier Chains Works

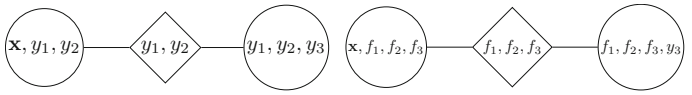
Fig. 3 shows CC on Logic. It is clear how it learns the XOR label (Y_3): by leveraging off labels Y_1 and Y_2, which are acting like hidden units of a neural network. In terms of neural networks it is actually more than required; Fig. 3c shows the smallest neural network that can learn XOR as demonstrated in [17][2].



(a) CC standard depiction **(b)** redrawn wrt y_3 **(c)** Minimum network able to learn XOR function

Fig. 3. CC with three labels, as in the Logic dataset. Note we show (for now) $\mathbf{x} = [x_1, x_2]$ as a single variable.

In Fig. 4, as a probabilistic graphical models interpretation, are the junction trees (see [1]) of two of the models, showing that Fig. 3c can be tractable. Standard CC (Fig. 3a—3b) is fully connected and thus many forms of inference are intractable; a known issue [2,15].



(a) Junction Tree of Fig. 3c **(b)** Junction Tree of Fig. 5b

Fig. 4. Junction Trees for different formulations

It is a straightforward interpretation of CC to think of labels being used as features to predict other labels. Let us not forget though that all estimated labels are derived from the input. Where labels are manually assigned to instances, then $y_j = f_j(\mathbf{x})$ are feature functions of the human mind, which (for a logistic regression base learner h_j) is being approximated by a sigmoid function on a linear combination of the input. From here we could get to conditional random fields (as in [2]), but we will continue through another route. From the point of view of y_3, there are three inputs ($\mathbf{x}$ still treated as a single variable),

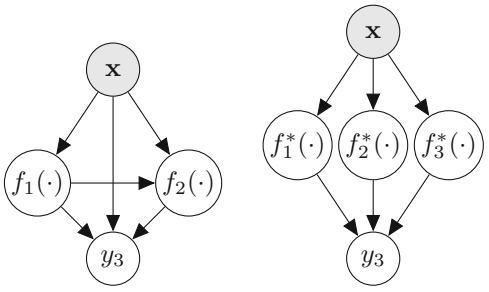
$$y_3 = h_3(f_1(\mathbf{x}), f_2(\mathbf{x}, f_1(\mathbf{x})), \mathbf{x})$$

[2] Earlier, pessimistic results about solving the XOR problem with neural networks [13] resulted in the decline of neural networks research.

See Fig. 5a. It is clear that f_1 and f_2 are simply transformations of the input. In the case of CC with logistic regression as a base classifier,

$$f_j(\mathbf{x}) := \sigma(\mathbf{w}_j^\top [x_1, \ldots, x_D, f_{j-1}(\mathbf{x}), \ldots, f_1(\mathbf{x})])$$

where σ is the logistic/sigmoid function, but we can easily imagine arbitrary (possibly non-linear) transformations of the input, and an arbitrary number of such functions: $f_1^*(\mathbf{x}), \ldots, f_K^*(\mathbf{x})$; see Fig. 5b. Note independence among $f_k^*(\mathbf{x})$.



(a) Sigmoid functions **(b)** Arbitrary functions

Fig. 5. As in Fig. 3b, but labels are shown as transformations of the input. Fig. 5a is easily equivalent to Fig. 5b in the case where $f_3^*(\mathbf{x}) = \mathbf{x}$, etc.

There is no reason to assume that the number of labels (L) equals the number of desired features (K). If we include the rest of the labels, expand $\mathbf{x}$ into D nodes, $X_1, \ldots, X_D$ and invert the graph such that the Y label variables are now at the top, the result is Fig. 6a. These last two changes are purely presentational, but important for what comes next.

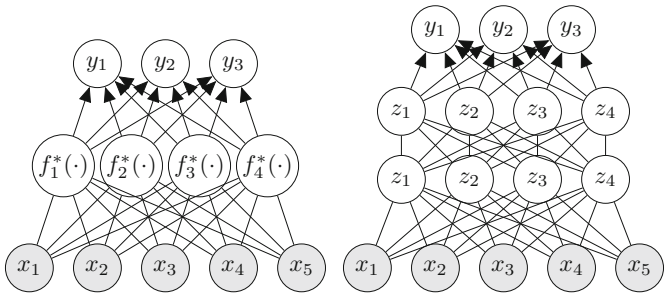
Since f_k^* is an arbitrary function, and two hidden layers are enough for universal approximation ability [9] of any arbitrary function, Fig. 6a is therefore equivalent to the *deep* network of Fig. 6b with, e.g., a sigmoid function for all layers; i.e.,

$$z_k^{[1]} = \sigma\Big(\sum_{d=1}^{D} x_d w_{dk}\Big)$$

for the first layer, where w_{jk} is a weight on the link between x_d and hidden node z_k.

The two hidden layers can be learned by restricted Boltzmann machines (RBMs) [8]. This means that training BR on the top layer ($\mathbf{z}^{[1]}$ vectors) can theoretically be as competitive as CC trained on the input ($\mathbf{x}$ vectors). In fact they can be equivalent, except that the RBMs discover feature functions, instead of trying to approximate the human feature functions (labels) available. Indeed, we do obtain top performance (as with CC, MCC) on Logic (not shown).

In other words, since all labels are related to the input, with an adequate (possibly non-linear) binary model, we can predict a label y_j just as well as



(a) A non-linear transform of the feature space **(b)** A deep belief network with two hidden layers

Fig. 6. A network with a non-linear transform of the feature space (left) and two layers to approximate it (right)

we could given also the prediction of another label y_k. Other labels are simply additional features of the input, albeit often quite powerful ones, since they often represent human neural circuitry (i.e., human concepts). The true function behind the concept is of course typically not known, but given the true outputs in the training data, they can be approximated (standard supervised learning).

Whereas a typical basis function is deterministic, the $f_k^*(\mathbf{x})$ are not (necessarily), as reflected in the RBMs. Guided by this, in the next section we employ some deep learning methods and show them to be effective. But Tab. 2 already hints that random features can help in a classifier-chains approach (particularly when the chain is carefully ordered). Models with random activations have been considered in e.g., [19], or in 'extreme learning machines' [10] – but as a single hidden layer and not directly into the label space as we consider here.

Table 2. Per-label accuracy on the Music dataset (see, e.g., [12]), from 5×CV, with (+) and without 10 random labels (i.e., feature functions), of the form $y_k = \sigma(\sum_{d=1}^{D} w_{dk} x_d)$ for random $\mathbf{w}$

label	CC	CC+	MCC	MCC+
amazed	0.759	**0.793**	0.772	**0.776**
happy	0.688	**0.692**	0.722	**0.734**
relaxing	**0.764**	0.755	0.764	**0.781**
quiet	**0.895**	0.890	0.882	**0.895**
sad	**0.835**	0.819	0.793	**0.827**
aggressive	0.759	**0.814**	0.793	**0.819**

5 Deep Multi-label Learning

Since labels can be seen as high-level features of the input, other higher-level features should also positively affect predictive performance. For example, from an image, a feature for the presence of a grainy surface such as sand or pebbles, or for being adjacent to a (significant) body of water should help us predict `beach` just as much (or better) than label `urban`. We can use RBMs to learn layers of such hidden features, in an unsupervised fashion. These hidden layers can capture complex dependencies and structure from the input space.

If the features are powerful, the label variables become independent. This is intuitively attractive, because humans do not recognise beaches depending on the probability that what they see is urban or not. Unfortunately, learning high level features in an unsupervised fashion is not as easy as trying to approximate labels from training data. Powerful algorithms and computational resources are needed – a currently active field of research.

Tab. 3 show results, comparing baseline `BR`, `MBR`, and `MCC`, with deep learning approaches[3]: namely two RBMs plus a multi-label learner, either `BR`, `MCC`, or with back-propagation (BP) as in [8] but for MLC; all denoted with `D`. Also we included [21]'s BP multi-label learner (`BPMLL`); a multi-layer neural network *not* initialized using RBMs. All experiments in this paper are carried out with the Weka-based Meka framework[4] with a setup like [15] (the datasets are described there). We used a single parameter combination for RBMs for all datasets (namely 30 hidden units per layer, learning rate 0.1, momentum 0.2, 5000 iterations) chosen ad-hoc – to avoid intensive parameter tuning on many datasets. Implementations are available within Meka. All base classifiers h_j are logistic regression (Weka's implementation). We evaluated using Hamming score and exact match described earlier, and additionally the micro averaged F-measure,

$$\text{Micro Averaged F1} := F_1([y_1^{(1)}, \ldots, y_L^{(N)}], [\hat{y}_1^{(1)}, \ldots, \hat{y}_L^{(N)}])$$

where $F_1(\mathbf{a}, \mathbf{b})$ returns the F_1 score of binary vectors $\mathbf{a}$ and $\mathbf{b}$.

Overall `D·MCC` performs best under exact match, but not as well as `D·BR` or `DBP` (which are closely related) under Hamming score – a result which corresponds with our discussion; `D·MCC` provides extra depth with a CC, but with the RBMs underneath `BR` already becomes very competitive – especially compared directly to baseline `BR`. A well-ordered CC (`MCC`) is still very powerful for exact match, but even better performance can be obtained with additional learned features. We could speculate that advances in deep learning should eventually reduce the effectiveness of `CC`, as higher-level features make labels more independent. Although, on the other hand, many kinds of CC models are more interpretable than RBMs (and usually faster to train), and may therefore still be interesting for many applications.

[3] Space does not permit a review of RBMs and deep learning, see e.g., [8] for details.
[4] `http://meka.sourceforge.net`

Table 3. Predictive performance on real datasets, with dataset-wise (rank)

Exact Match

Dataset	BR	MBR	BPMLL	MCC	D·MCC	D·BR	DBP
music	0.193 (6)	0.193 (6)	0.252 (3)	0.208 (5)	0.218 (4)	0.267 (2)	0.287 (1)
scene	0.286 (6)	0.292 (5)	0.554 (2)	0.353 (4)	0.476 (3)	0.582 (1)	0.183 (7)
yeast	0.150 (5)	0.137 (7)	0.161 (4)	0.198 (2)	0.204 (1)	0.149 (6)	0.179 (3)
genbase	0.960 (3)	0.955 (4)	0.271 (7)	0.965 (1)	0.965 (1)	0.950 (5)	0.950 (5)
medical	0.439 (4)	0.457 (3)	0.194 (7)	0.474 (2)	0.361 (5)	0.200 (6)	0.521 (1)
enron	0.022 (5)	0.022 (5)	0.010 (7)	0.028 (4)	0.161 (1)	0.054 (2)	0.043 (3)
avg. rank	4.83	5.00	5.00	3.00	2.50	3.67	3.33

Hamming Score

Dataset	BR	MBR	BPMLL	MCC	D·MCC	D·BR	DBP
music	0.761 (5)	0.762 (4)	0.776 (2)	0.742 (6)	0.726 (7)	0.772 (3)	0.791 (1)
scene	0.807 (4)	0.802 (6)	0.895 (1)	0.807 (4)	0.847 (3)	0.895 (1)	0.731 (7)
yeast	0.786 (3)	0.780 (5)	0.790 (2)	0.771 (7)	0.780 (5)	0.784 (4)	0.791 (1)
genbase	0.998 (3)	0.998 (3)	0.932 (7)	0.999 (1)	0.999 (1)	0.998 (3)	0.998 (3)
medical	0.980 (4)	0.981 (2)	0.969 (6)	0.981 (2)	0.971 (5)	0.967 (7)	0.984 (1)
enron	0.892 (6)	0.904 (5)	0.939 (3)	0.884 (7)	0.940 (2)	0.947 (1)	0.937 (4)
avg. rank	4.17	4.17	3.50	4.50	3.83	3.17	2.83

Micro-Averaged F1

Dataset	BR	MBR	BPMLL	MCC	D·MCC	D·BR	DBP
music	0.570 (7)	0.571 (6)	0.603 (2)	0.574 (5)	0.580 (3)	0.577 (4)	0.629 (1)
scene	0.463 (5)	0.406 (6)	0.668 (1)	0.480 (4)	0.621 (2)	0.576 (3)	0.199 (7)
yeast	0.599 (6)	0.618 (3)	0.633 (2)	0.601 (5)	0.607 (4)	0.588 (7)	0.639 (1)
genbase	0.987 (1)	0.985 (2)	0.276 (5)	0.985 (2)	0.341 (4)	0.200 (6)	0.174 (7)
medical	0.665 (3)	0.655 (4)	0.315 (7)	0.681 (2)	0.557 (5)	0.487 (6)	0.771 (1)
enron	0.372 (5)	0.248 (7)	0.483 (2)	0.353 (6)	0.475 (3)	0.493 (1)	0.466 (4)
avg. rank	4.50	4.67	3.17	4.00	3.50	4.50	3.50

6 Conclusions

The high performance of the classifier chains (CC) approach can be seen as stemming from its leverage of labels as high-level features in a deep cascading structure across binary classifiers. This contrasts with many other approaches based on binary classifiers, that leverage label dependence in a regularization step, but provide limited additional learning power. We demonstrated several scenarios where CC can learn a concept where other methods fail.

We argued that if labels can be considered high-level features stemming from the input, then it is possible to learn such features independently of the training data. We employed deep-learning approaches (using restricted Boltzmann machines) to learn such higher-level features, and obtained provide strong performance, particularly when supplemented with a top-layer chain. Results indicate

that further advances in multi-label classification will come from better models of features, and borrow from thus-related fields, rather than obsessive modelling of high-level label 'correlations'.

Many deep-learning methods have other important advantages, particularly in online and semi-supervised settings. We intend to investigate this, as well as produce further empirical study.

References

1. Barber, D.: Bayesian Reasoning and Machine Learning. Cambridge University Press (2012)
2. Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML 2010: 27th International Conference on Machine Learning, pp. 279–286. Omni Press, Haifa (2010)
3. Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning 76(2-3), 211–225 (2009)
4. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Mach. Learn. 88(1-2), 5–45 (2012)
5. Dembczyński, K., Waegeman, W., Hüllermeier, E.: An analysis of chaining in multilabel classification. In: ECAI: European Conference of Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 294–299. IOS Press (2012)
6. Ghani, R.: Using error-correcting codes for text classification. In: ICML 2000: 17th International Conference on Machine Learning, pp. 303–310. Morgan Kaufmann Publishers, Stanford (2000)
7. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 22–30. Springer, Heidelberg (2004)
8. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
9. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359–366 (1989)
10. Huang, G.-B., Wang, D., Lan, Y.: Extreme learning machines: A survey. International Journal of Machine Learning and Cybernetics 2(2), 107–122 (2011)
11. Kumar, A., Vembu, S., Menon, A.K., Elkan, C.: Learning and inference in probabilistic classifier chains with beam search. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part I. LNCS, vol. 7523, pp. 665–680. Springer, Heidelberg (2012)
12. Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition 45(9), 3084–3104 (2012)
13. Minsky, M., Papert, S.: Perceptrons — An introduction to Computational Geometry. The MIT Press (1969)
14. Read, J., Achutegui, K., Miguez, J.: A distributed particle filter for nonlinear tracking in wireless sensor networks. Signal Processing 98, 121–134 (2014)
15. Read, J., Martino, L., Luengo, D.: Efficient monte carlo methods for multi-dimensional learning with classifier chains. Pattern Recognition 47(3) (2014)

16. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning 85(3), 333–359 (2011)
17. Rumelhart, D.E., McClelland, J.L., Research Group, P.D.P. (eds.): Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. MIT Press, Cambridge (1986)
18. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3), 297–336 (1999)
19. Thomas Miller III, W., Glanz, F.H., Gordon Kraft III, L.: CMAC: An associative neural network alternative to backpropagation. Proceedings of the IEEE 78(10), 1561–1567 (1990)
20. Zaragoza, J.H., Sucar, L.E., Morales, E.F., Bielza, C., Larrañaga, P.: Bayesian chain classifiers for multidimensional classification. In: 24th International Conference on Artificial Intelligence (IJCAI 2011), pp. 2192–2197 (2011)
21. Zhang, M.-L., Zhou, Z.-H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering 18(10), 1338–1351 (2006)