Multi-label Classification using Labels as Hidden Nodes

Jesse Read  
jesse.read@aalto.fi

Jaakko Hollmén  
jaakko.hollmen@aalto.fi

Aalto University, Department of Computer Science  
PO Box 15400, FI-00076 Aalto, Espoo, Finland

Helsinki Institute for Information Technology (HIIT), Finland

Editor:

Abstract

Competitive methods for multi-label data typically invest in learning labels together. To do so in a beneficial way, analysis of label dependence is often seen as a fundamental step, separate and prior to constructing a classifier. Some methods invest up to hundreds of times more computational effort in building dependency models, than training the final classifier itself. We extend some recent discussion in the literature and provide a deeper analysis, namely, developing the view that label dependence is often introduced by an inadequate base classifier, rather than being inherent to the data or underlying concept; showing how even an exhaustive analysis of label dependence may not lead to an optimal classification structure. Viewing labels as additional features (a transformation of the input), we create neural-network inspired novel methods that remove the emphasis of a prior dependency structure. Our methods take an important advantage particular to multi-label data: they leverage labels to create effective units in middle layers, rather than learning these units from scratch in an unsupervised fashion with gradient-based methods. Results are promising. The methods we propose perform competitively, and also have very important qualities of scalability.

Keywords: multi-label classification, meta-labels, problem transformation, neural networks, stacking

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. The typical argument is that, since these labels are often strongly correlated, modeling the dependencies between them allows MLC methods to obtain higher performance than if labels were modelled independently.

As in general classification scenarios, an $n$-th feature vector (instance) can be represented as $x^{(n)} = [x_1^{(n)}, \ldots, x_D^{(n)}]$, where each $x_d \in \mathbb{R}$. In the traditional binary classification task, we are interested in having a model $h$ to provide a prediction for test instances $\tilde{x}$, i.e., $\hat{y} = h(\tilde{x})$; where $h$, probabilistically speaking, seeks the expectation $\mathbb{E}[y|x]$ of unknown $p(y|x)$. In MLC, there are $L$ binary output class variables (labels), and we are interested
Figure 1: BR (1a) and CC (1b) as graphical models, $L = 4$. Unlike typical Bayesian or neural-network diagrams, we have lumped $x = [x_1, \ldots, x_D]$ into a single variable, so as to focus on label variables.

in predictions

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

where $y_j = 1$ indicates the relevance of the $j$-th label; $j = 1, \ldots, L$. From $N$ labelled examples (training data) $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$, we infer $h$. Tab. 1 summarizes the main notation.

| Notation | Description |
|----------|-------------|
| $x = [x_1, \ldots, x_D]^T$ | instance (input vector); $x \in \mathbb{R}^D$ |
| $y = [y_1, \ldots, y_L]^T$ | $L$-dimensional label/output vector, $y \in \{0,1\}^L$ |
| $y_j \in \{0,1\}$ | a binary label indicator, $j = 1, \ldots, L.$ |
| $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$ | Training data set, $n = 1, \ldots, N$ |
| $\hat{y} = h(\tilde{x})$ | predicted label vector given test instance |
| $w_j = [w_1, \ldots, w_D]$ | model parameters/weights for $j$th binary model |
| $\hat{y}_j = h_j(\tilde{x})$ | individual binary classification for the $j$-th label |
| $z = [z_1, \ldots, z_H]^T$ | $H$-dimensional vector of hidden variables, $z \in \{0,1\}^H$ |

A most basic and probably most well known solution is to train $L$ binary models, one for each label. This method is called binary relevance (BR); illustrated graphically in Fig. 1a. BR classifies an $\tilde{x}$ individually for each of the $L$ labels, as $h_{BR}(\tilde{x}) := [h_1(\tilde{x}), \ldots, h_L(\tilde{x})]$.

 Practically the entirety of the multi-label literature points out that the independence assumption among the labels leads to suboptimal performance (e.g., Godbole and Sarawagi (2004); Cheng and Hüllemeier (2009); Read et al. (2014, 2011) 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.

Several approaches try to ‘fix’, or regularize BR. For example, Meta-BR (MBR, also known variously in the literature as ‘stacked-BR’ and ‘2BR’) Godbole and Sarawagi (2004); Cheng
and Hüllermeier (2009) stacks the output of one BR as input into a second (meta) BR, optionally with a skip layer\(^1\), so as to learn to correct the errors of the first model.

A related approach uses subset mapping, such as employed by Schapire and Singer (1999), to automatically map infrequent label vector predictions to a more frequent ones. Typically, frequent label vectors are considered as those found in the training set. BR's output label vector can be mapped to one of these existing vectors, with some penalty function typically rewarding small Hamming distance and high frequency in the training set. This is one approach in the family of error-correcting output code methods (Ghani (2000)) and, like MBR, can be seen as a regularizer; attempting to correct an initial classification based on meta knowledge such as label co-occurrence frequencies.

Rather than ‘correcting’ classifications, several families of methods attempt to learn and classify the labels together. The label powerset method (LP, see (Tsoumakas and Katakis, 2007)) is one such family, where each label vector as a single class value in a multi-class problem. The vector chosen as the ‘class label’ is trivially decoded (deterministically) into separate label relevances (i.e., \(\hat{y}_1, \ldots, \hat{y}_L\)) = \(\hat{y} = h_{\text{LP}}(\tilde{x})\), where \(\hat{y}\) is – for example – one of the label vectors observed in the training set).

Another of these is the family of methods based on classifier chains (CC, Read et al. (2011)), illustrated in Fig. 1b. A classifier chain models is related to BR, but uses binary label predictions as extra input attributes for the following classifier, cascaded along a chain. Like LP therefore, it models labels and inputs together, rather than correcting labels as a separate step. In the original formulation with greedy inference,

\[
h_{\text{CC}}(\tilde{x}) := \left[ h_1(\tilde{x}), h_2(\tilde{x}, h_1(\tilde{x})), \ldots, h_L(\tilde{x}, h_1(\tilde{x}), \ldots, h_{L-1}(\tilde{x})) \right].
\]

CC variants have consistently performed strongly in the literature and there have been numerous extensions, variations and analyses in the last few years, e.g., Dembczyński et al. (2010); Zaragoza et al. (2011); Read et al. (2014); Kumar et al. (2012); Dembczyński et al. (2012). 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, Read et al. (2011) suggested an ensemble of randomly-ordered chains (ECC) with voting, whereas two recent high-performing CC methods (Kumar et al., 2012; Read et al., 2014), use beam and Monte Carlo search, respectively to obtain one well-ordered chain. These methods analyse up to hundreds of potential chain configurations.

Likewise many LP-based methods have been proposed (Tsoumakas et al. (2011, 2008) are two of the most well-known), mainly focusing on different subsets of the label space. The purpose of using subsets is mainly for tractability, but certain subsets will lead to better performance than others, much in the way that different chain orders and linkages will lead to better performance of CC. In both methods, modelling label dependence has been a key focus for many methods.

This paper is based on our earlier work of (Read and Hollmén, 2014), in which we discuss modelling label dependence with respect to classifier chains. The main points argued were: 1) the best label order CC is impossible to obtain from analysis of observational data only; 1. In other words, the input instance can be included again as additional input to the second layer, or just the predictions of the first layer; there is no particular consensus in the literature regarding this
and 2) the high performance of classifier chains is due to leveraging earlier labels in the chain as additional feature attributes, much like basis functions or deep learning approaches – contrasting with other methods like MBR and SM which provide only a regularization. This extended earlier discussion by Dembczyński et al. (2012); Dembczyński et al. (2012) on the role of label dependence on the performance of classifier chains and multi-label classification in general.

In this manuscript we more thoroughly elaborate and formalize the earlier arguments, and provide theoretical backing to the respective claims. Building on this, we create novel methods. These methods draw inspiration from deep learning and neural network methods in the sense of having a hidden layer representation but have the particular advantage that they do not need gradient based learners to learn the hidden nodes. The first method uses synthetic labels to allow reliable performance of a classifier chain without any effort invested in chain ordering (contrasting with earlier methods that invest much computational power into this step). This method can be extended with a second layer of BR; combining the advantages of BR (independent outputs), MBR (regularization) and CC (modelling labels together). The other method we develop is additionally inspired by modern methods of the LP family: it uses label transformations, in addition to input-attribute transformations, to act as nodes in inner layers of a layered network. As the second variation, it ensures greater independence among final layer labels, meaning that the top-layer chain is no longer necessary at all (with important implications for practical deployments). These methods show strong performance on multi-label datasets, and they also provide a strong base for promising future work along the same line.

The rest of the paper is organized as follows. In Section 2 we review and expand the discussion of the role of label dependence in multi-label classification. In Section 3 we lean on theory from probabilistic graphical models and deep learning to derive a top performing multi-label classifier where output nodes are unconnected (as in BR), demonstrating the equivalence of such a classifier with an optimally-structured classifier based on label dependence. In Section 4 we present our novel methods, and discuss additional related work in Section 5. In Section 6 we conduct and discuss results of an empirical evaluation of our proposed methods against related and competitive methods from the literature; and finally, in Section 7 we summarize and draw conclusions.

2. The Role of Label Dependence in Multi-label Classification

Throughout the multi-label literature the binary relevance method (BR) has been cast out of serious consideration on account of assuming independence among target variables. A lot of empirical evidence has been provided indicating that this method can indeed be outperformed by several, or even tens of percentage points on standard multi-label datasets.

The main feature of BR that sets it aside from more advanced methods is its assumption of label dependence. Literally hundreds of papers on multi-label classification have been published over recent years presenting new variations of modelling label dependence, and showing that a proposed method improves over BR. The underlying idea is that a method that models label dependence will lead to a higher-performing structure. For example, if two labels occur together frequently, rarely, or not at all – this should be implicitly or explicitly modelled. The literature has focussed both marginal dependence (frequency of co-
occurrence among labels $y_1, \ldots, y_L$) and conditional dependence (after conditioning on, i.e., taking into account, the input $-x$). Since modelling complete dependence is intractable for most datasets, many approaches have focussed on pairwise marginal dependence ([Zaragoza et al., 2011] is one example), and/or employ an ensemble to compensate for an incomplete model of conditional dependence (Tenenboim et al., 2010; Read et al., 2014). Modelling conditional dependence is particularly intensive, as it inherently involves training classifiers. Zhang and Zhang (2010) developed a method that only needs to train binary models internally and then use the errors of these models to model the dependence, as an approximation of measuring conditional dependence for each pair of labels (which entails training $L(L-1)/2$ classifiers).

Among the methods put forth in the literature, all report results higher than BR. Scalability varies from intractable on anything but the smallest datasets, to much faster than BR itself. However, in most cases the difference between the performance of models is negligible over a large range of datasets. Many methods are attractive for reasons of interpretation or reduced running time, but many of the top recent methods claiming to leverage label dependence in a particular way will invariably not outperform each other by any significant margin (overall, over a reasonable number of datasets). A good large-scale evaluation is given in Madjarov et al. (2012), showing that LP and CC-based methods, and random forest ensembles, were among the most competitive algorithms.

As improvements of prediction on standard multi-label datasets has begun to reach a plateau, several authors have begun questioning the implied logic behind developments over the past years: namely, that if the ground-truth label dependence could be known and modelled, multi-label predictive performance would be optimal (given a fixed base classifier); and therefore as more technique and computational effort is invested into modelling label dependence (narrowing the gap to the ground truth), the lead of new methods over BR and other predecessors will widen. Already Dembczyński et al. (2012) make the case that it should be possible make risk-minimizing predictions without any particular effort to detect or model label dependence; which we followed up and expanded on in Read and Hollmén (2014). This seems to throw into doubt the bulk of the contributions to the multi-label literature. In a more empirical take, Read et al. (2011); Dembczyński et al. (2012) ponder if BR has been underrated and could equal advance methods’ performance with enough training data (under the implication that the big data era will render label-dependence irrelevant). In other words, that modelling label dependence is a compensation of lack of training data, and one could assume that given infinite data, two separate binary models on labels $y_j$ and $y_k$ could achieve as good performance as one that models them together (again, assuming the same base classifier).

To illustrate in practical terms, if we take two labels from the commonly-used Scene dataset (say, beach and urban): the assumption that label dependence is key to optimal multi-label accuracy, is analogous to assuming that an expert (human or otherwise) trained for visually recognising urban environments is more likely to classify an image correctly after knowing if a peer believes the same image contains a beach or not.

In (Read and Hollmén, 2014) we explained how modelling dependence only helps when a base classifier behind one or more labels is inadequate. This depends, of course, on the base classifier, and crucially, it means that there is no guarantee that an ideal structure based on label dependence can be found at all given any amount of training data. That
Figure 2: With a linear base learner, and non-noisy data $\mathbf{x}_i \in \{0,1\}^2, y_i \in \{0,1\}^3$, model 2a and 2b perform equally poorly (50% under an exact match measure), whereas 2c performs perfectly in the same scenario. A human expert knows that there is no conditional dependence between XOR and OR (they are independent of each other given $\mathbf{x}$): this dependence is introduced by using an inadequate (linear) base learner, but even detecting or assuming this dependence will not necessarily lead to an improvement.

is to say: the structure is likelihood equivalent, and changing the base classifier will change the ideal structure.

Fig. 2 illustrates the dilemma with a toy example. The only possible guaranteed method to overcome this is if all possible models of dependence (of factorial complexity) can be trained with the same classifier that will be used for evaluation (implying at least one internal train-test split for each model). Some methods, such as that of (Read et al. (2014)), indeed sample this space of dependence models, but a full search thereof is limited to only the tiniest datasets.

In summary, we take the view that BR can perform as well as any other method if there is no dependence among the outputs given (conditioned on) the inputs. However, we can add, and strongly emphasise, that this is not the same as saying that BR should perform as well as other methods if there is no dependence detected. Due to noisy data or an insufficient model, dependence may missed or even introduced. If a multi-label method outperforms BR (under the same base classifier), then we can say that it is using label dependence to compensate for an inadequacy in its base classifiers. In the following section, we use techniques inspired from deep learning to remove the dependence among the labels (top-layer outputs), and thus provide a BR layer theoretically as powerful as any other model could manage with dependence among the outputs in this top layer.

3. Obtaining a State-of-the-Art Binary Relevance Classifier

In (Read and Hollmén, 2014) we explained that classifier chains (CC) obtains its performance advantage by using early labels in the chain as a feature-projection for later labels (in order to compensate for the dependence introduced by inadequate base learners\(^2\)). Similar conclusions have been separately provided with respect to the label powerset method (LP) by Dembczyński et al. (2012): namely, that predictive power comes as a result of working in a higher space. As mentioned above, CC and LP represent most of the popular approaches in the multi-label literature, and lie behind many advanced methods.

---

2. They may be inadequate for any number of reasons. We have exemplified the case of non-linearity, but using a non-linear base learner does not guarantee that the non-linearity is correctly modelled
It is clear that approaches to multi-label learning based on these methods have had considerable success in the multi-label literature, providing state-of-the-part performance and many options for scalability. Examples include (Tsoumakas et al., 2011, 2008; Dembczyński et al., 2010; Read et al., 2011). Nevertheless, they have many disadvantages: special considerations are necessary to scale up to large numbers of labels, neither are naturally flexible enough to allow for the addition and removal of labels over time, and they cannot provide a truly parallelized implementation. Both the label powerset and classifier chains problem transformations have proved difficult to apply to an incremental scenarios such as data streams (Read et al., 2012): LP on account of an ever increasing number of classes, and CC on account of overfitting. To create a scalable LP-based learner requires serious forethought with regard to subset division and/or pruning. CC is relatively fast ‘out-of-the-box’ via greedy inference on the chain, but at the cost of error-propagation. Additionally, the huge selection of methods and parameterizations thereof under each method family can be daunting to practitioners, increasing the temptation to go with the ‘default’ option of BR and build one separate model for each label.

Therefore the basic binary relevance approach, with no inter-linkage among outputs, remains an attractive option. In the following, we show that for any dataset, an optimal BR classifier can be obtained.

3.1 Removing Label Dependence with Hidden Units

**Proposition 1** If, with classifier \( h_2 \), we can estimate \( \mathbb{E}(Y_2|Y_1, X) \) to a certain degree of accuracy under some evaluation measure, then there also can exist a classifier \( h'_2 \) with which it is also possible to estimate \( \mathbb{E}(Y_2|X) \) with at least the same accuracy under the same measure.

In the following we derive such a classifier in all possible scenarios. For the purposes of illustration, and without lacking generality, we assume a scenario with input \( X \), and two labels, \( Y_1 \) and \( Y_2 \). For the sake of a toy example, we can imagine that \( X \) represents some kind of document, and \( Y_1 \) and \( Y_2 \) represent the relevance (or not) of two subject categories for it. We can imagine that latent variable \( Z \) represents the unobservable current events which may affect both the composition of \( X \) and the decisions for labelling it.

Various scenarios are shown graphically in Fig. 3. To summarize,

| \( X \) | text document |
| \( Z \) | current events |
| \( Y_1 \) | labeller 1’s decision |
| \( Y_2 \) | labeller 2’s decision |

Let us first safely ignore the case where all input and label variables are randomly generated from completely independent distributions (\( P(X, Y_1, Y_2) = P(X)P(Y_1)P(Y_2) \)), since this provides no interest from a learning point of view.

With this case aside, the most basic case is that of conditional independence, illustrated in Fig. 3a. For example, where a text document is given independently to two human labellers, who each independently identify if the document is relevant to their expert domain.

---

3. In CC, training can be done in parallel, but not prediction.
In an alternative generative view, the two specialists could write pieces of text which are placed independently into document $X$. In this case, we can write

$$p(y, x) = p(y_1, y_2|x)$$
$$= p(y_1|x)p(y_2|y_1, x)$$
$$= p(y_1|x)p(y_2|x)$$

and it is obvious that BR is sufficient, where $h_j(\tilde{x}) := \arg\max_{y_j} p(y_j|\tilde{x})$.

In another feasible scenario, a text document is labelled by the first labeller (an expert in a particular domain), and afterwards by the second expert – potentially biasing the decision to label relevance or not with this second label. This is illustrated in Fig. 3b. If we do not impose any restriction (e.g., linearity) on any $h_j(x)$, it is straightforward to make $h_2(x) \equiv h_2(x, h_1(x)) \equiv h(x, z)$ where $z \equiv h_2(x)$ is some latent variable, thus leading to a representation like that of Fig. 3c. Since $Y_1 \equiv Z$, if we recover latent variable $Z$, we can recover $Y_2$ to the same degree of accuracy. In the analogy, the second labeller must learn also the first labeller’s knowledge. This makes the first labeller redundant. Indeed, if we drop $Y_1$, we immediately return to structure of Fig. 3b.

The most complicated (i.e., most connected) scenario is that of Fig. 3d, where interaction between the two experts and the observation $X$ and an unobserved variable $Z$ (for example, two experts label a document $X$, but both are biased by each other and – possibly to alternate degrees – by an external source of information $Z$).

With reference to Fig. 3e, we can derive a model by adding in extra ‘copies’ of the label features as $z^y = [z_1, z_2] \equiv [y_1, y_2]$, combining it with $z$, and then marginalizing these as

---

Figure 3: Undirected graphical models elaborating possible scenarios behind the generation of multi-label data $\{(x, y_i)\}_{i=1}^N$. Directionality can be added as desired.
latent variables, as follows:

\[
p(\mathbf{y}, \mathbf{x}) \propto f(\mathbf{y}_2, \mathbf{y}_1) f(\mathbf{x}_1, \mathbf{x}_2) \sum_{\mathbf{z}} f(\mathbf{y}_1, \mathbf{z}) f(\mathbf{y}_2, \mathbf{z}) f(\mathbf{z}, \mathbf{x})
= f(\mathbf{z}^y, \mathbf{x}) \sum_{\mathbf{z}} f(\mathbf{y}_1, \mathbf{z}) f(\mathbf{y}_2, \mathbf{z}) f(\mathbf{z}, \mathbf{x})
= \sum_{\mathbf{z}} f(\mathbf{y}_1, \mathbf{z}) f(\mathbf{y}_2, \mathbf{z}) f(\mathbf{z}, \mathbf{x})
\times \sum_{\mathbf{z}} f(\mathbf{y}, \mathbf{z}) f(\mathbf{z}, \mathbf{x})
\propto \sum_{\mathbf{z}} f(\mathbf{x}, \mathbf{z})
\]

A possible representation is illustrated in Fig. 4 as a neural network, which is in fact a generic version of all models illustrated in Fig. 3. Note that there are no connections among label nodes. The top layer is essentially a BR model, with independence among outputs. Such a model can then be equivalent to having the ground truth label dependence and constructing an optimally connected (e.g., in the label layer) model.

This is not to say that all recent multi-label research (at least that having attempted to improve on BR) has contributed nothing, since we are only claiming here the existence and desirability of such a model, not that is possible, or even easy to learn.

Hornik et al. (1989) builds on the Kolmogorov theorem (Sprecher, 1965), leading to the idea of universal approximation: that with a finite number of neurons (i.e., functions), even with a linear output layer, a network can approximate any continuous function. In the multi-label case, it means that given a large enough (but finite) feature representation, the concept of each label can be learned, and thus with a finite number of labels, a linear BR layer can suffice for optimal classification performance. One or two layers should be enough, depending on the type of function creating the inner layer(s). With radial basis functions, a single layer is sufficient (we return to this in Section 5). Again, though, this does not deal with the learn-ability of such a model.

Energy models such as Restricted Boltzmann Machines (RBMs), widely studied and popular in deep learning (Hinton and Salakhutdinov, 2006), are one approach to search for such a model in probability space, where \( p(\mathbf{x}, \mathbf{y}, \mathbf{z}) \propto f(\mathbf{x}, \mathbf{y}, \mathbf{z}) \), and thus we can model the
layers with

\[ f(x, y, z) = \exp\{-E(x, y, z)\} \]

\[ = \exp\{-z^\top Wx - b^\top c - c^\top z - d^\top y - z^\top Uy\} \]

where \( U, W \) are weight matrices, and \( b, c, d \) are biases.

To summarize, if we find dependence between two labels \( Y_1 \) and \( Y_2 \), it can be seen as a result of marginalizing out hidden variables \( Z \) that generated them. The process can also be seen in reverse: we can add hidden variables to remove the dependence between labels, such that

\[ Y_2 \perp \perp Y_1 | Z \]

and also, \( Y_2 \perp \perp X | Z \)

The overlap between energy models and multi-label classification is the basis of inspiration for our novel methods; given in the following.

### 3.2 Multi-Label Learners as Deep Learners

Although there exists an optimal representation where multiple labels are predicted independently of each other, this does not imply having a method to learn it. For this, we turn to the rapidly growing field of deep learning for inspiration. This area faces the same challenge: how to learn powerful higher-level feature representations of the data. It approaches this task by using multiple hidden layers, rather than a single hidden layer (as we exemplified in Fig. 4). Typically each layer is initially trained in an unsupervised fashion, and higher layers represent higher-level abstractions from the input. State of the art methods for training these models involve contrastive divergence (Hinton, 2002). Recent adaptive gradient (Duchi et al., 2011) and dropout (Srivastava et al., 2014) methods have become popular. Any off-the-shelf method can be used to predict the class label(s) from the top-level hidden layer, or back propagation can be used to fine-tuned all weights for discriminative prediction (Hinton and Salakhutdinov, 2006).

Deep learning focuses heavily on the unsupervised aspect of learning to discover high-level feature representations. However, as discussed above, in multi-label classification we already have a number of such feature representations available: the labels themselves. Existing methods, such as classifier chains, can in fact already be viewed as ‘deep learners’, as hinted at in Fig. 5: under CC, \( y_3 \) can be seen as a composition of layers of features cascaded together. In the random \( k \)-label subset method (RAkEL) of Tsoumakas et al. (2011), meta labels can be viewed as inner-layer features (for which a deterministic composition to the label space is used in the form of ensemble voting). Deep learning researchers probably disagree that methods like CC or LP can be considered deep learners, as the features, although ordered in a hierarchical way, are not learned, but given as part of the dataset (as supervised labels); and furthermore RAkEL only has one middle layer. However, in any case, it is a particular advantage to multi-label data that multiple labels are available and can be treated as additional higher-level features. Indeed, many existing multi-label methods are implicitly gaining predictive performance by stacking layers of labels in a related way to deep learners stack feature attribute representations. In the next section we present novel methods for multi-label classification that explicitly exploit this ambiguity between feature and label.
Figure 5: Methods CC and RAKEL (subsets of LP) as networks. Both create different projections via layers in the output space.

4. Novel Methods Using Labels as Hidden Nodes

In this section we develop approaches that stem from the conceptual overlap between labels and high-level features.

4.1 Classifier Chains Augmented with Synthetic Labels (CCASL)

A common ‘hassle’ of classifier chains methods is searching for and evaluating a good order for the labels. This was discussed earlier and illustrated in Fig. 2 with the toy example considering logical operations as labels. If a ‘difficult’ label is at the beginning a given base learner may fail to learn it and, even worse, propagate this error to the other learners. This problem cannot easily be identified or solved by dependency analysis. Some methods consider aligning the chain based on label difficulty (e.g., Senge et al. (2013)). But this requires training and evaluating classifiers to decide which is easiest, and furthermore, the difficulty wrt the chain is conditional: it depends on other labels. Other methods search the space of chain orderings (Kumar et al., 2012; Read et al., 2014) but all efforts come with a computational price that increases prohibitively with the number of labels. Guo and Gu (2011) and others create an undirected network, rather than a directed chain, but this relies on relatively costly Gibbs sampling at inference time.

Since CC learns as a kind of deep method, and higher-level labels are best learned at the end of the chain, we present a method that simply adds synthetic labels to the beginning of the chain, and builds up a non-linear representation, which can be leveraged by other classifiers further down the chain. We call this Classifier Chains Augmented with Synthetic Labels (CCASL).

Namely, we create $H$ artificial labels. We choose to base this construction on Rectified Linear Units (ReLUs) due to their recent success in deep networks. To create synthetic label $z_k$,

$$z_k = h_k(x; w_k, t_k) = a_k \cdot 1_{a_k > 0}$$

using $a_k = w_k^T x'_k$ as the activation function; where $w'_k = w_{k,1:D+k-1}$ and input $x'_k = [x_1, \ldots, x_D, z_1, \ldots, z_{k-1}]$; where $W$ is a $H \times (D + H - 1)$ matrix $W$ such that $w_{jk} \sim \mathcal{N}(\cdot|0, 0.2)$. Fig. 6 shows the terrain of this function. Fig. 7 shows the resulting network view.
Figure 6: Terrain of a Rectified Linear Unit (ReLU) in two dimensions.

Figure 7: Classifier Chains Augmented with Synthetic Labels (CCASL). Random labels \( z \) are appended to the chain prior to the real labels. The method learns as per ordinary classifier chains: a number of binary classifiers, each responsible for one of the nodes \( z_1, \ldots, y_L \). The graph shows the input to each classifier.

Thus note that we build up synthetic labels in a classifier chain: each one has a slightly more complex space (expanded by one dimension) than the last. A further possible modification is to use a dropout-inspired technique where \( w'_k = w_k \ast m_k \) where \( m_k \sim \text{Bernoulli}() \) is a dropout mask, dropping out (masking) completely some input units to each label. Note that we assume standardized \( X \) space.

We then incorporate these labels into \( y' = [z_1, \ldots, z_H, y_1, \ldots, y_L] \): a new augmented label vector such that the chain follows on from the synthetic labels to the true labels. It is not an arbitrary choice that the synthetic labels go at the beginning: we want to leverage these labels as a higher space projection to improve prediction of the real labels. We train classifier chains model, such that we obtain predictions

\[
\hat{y}' = h_{\text{CC}}(\tilde{x})
\]

of which we extract \( \hat{y} = [\hat{y}_{H+1}, \ldots, \hat{y}_{H+L}] = [\hat{y}_1, \ldots, \hat{y}_L] \) as the ‘official’ classification. As results show, this method is able to learn the toy Logical much more often than CC, and does so regardless of the order chosen for the original labels \( y_{\text{OR}}, y_{\text{AND}}, y_{\text{XOR}} \).

Essentially CCASL creates a deep unsupervised projection of the input space. Scalability is easily tunable by reducing the number of connections: each node in the chain can project to only a random subset of nodes further down the chain. It has already been shown that ‘partially-linked’ CC usually leads to little to no performance degradation with respect to CC (Zaragoza et al., 2011).
4.2 Adding a Meta Layer: CCASL+BR

Results show that CCASL performs well. However, earlier we discussed desirability to assume independence among top layer labels, and instead ‘hide’ the dependency model in inner, hidden layers. In a related fashion to MBR, we can simply stack a BR model on top of CCASL, to get CCASL+BR. The network now looks like that of Fig. 8.

This method combines the advantages of many important methods in the multi-label literature: namely BR (outputs are predicted independently of each other\(^4\)), MBR (predictions are stacked again as a form of regularization), and CC (a chain is used to leverage interdependency of labels). It can be seen as a single network where only a number of \(Z\) units must be parameterised, with respect to computational limitations. All nodes are trained in a supervised fashion, no back propagation is necessary.

4.3 Classification with an Augmented Middle Layer (CAML)

In the multi-label literature there are numerous comparisons of CC and LP family methods. From a probabilistic perspective, LP can be viewed as modeling the joint probability directly,

\[
h_{LP}(\mathbf{x}) := \arg\max_{\mathbf{y}} p(\mathbf{y}, \mathbf{x})
\]

whereas CC approximates this distribution using the chain rule from probability theory,

\[
h_{CC}(\mathbf{x}) := \arg\max_{\mathbf{y}} p(y_1|\mathbf{x}) \prod_{j=2}^{L} p(y_j|\mathbf{x}, y_1, \ldots, y_{j-1})
\]

In reality, for reasons of scalability, both methods use approximations. CC is typically employed with a greedy approximation (each \(y_j\) is propagated greedily down the chain, i.e., with \(L\) binary decisions, each \(y_1\) is classified once) or sampling (a limited number of iterations of this). RAkEL is a good example of a method that makes LP tractable, namely by creating an ensemble of smaller label sets, training LP on each subset, and later combine the votes. For example, if \(\hat{y}_{1,3,6} \in \{[1, 0, 1], [0, 0, 0], [0, 0, 1]\}\) representing the LP problem for

---

4. At least in the final layer
one such set (where 1, 3, 6 ⊂ \{1, \ldots, L\}) If \([1, 0, 1]\) then the ensemble gets one vote each for the 1-st and 6-th labels.

If the RAkEL methodology is cast to a network of binary nodes, we can imagine a number of feature functions created \textit{from the label space} for each subset. For example,

\[
\phi_k(y) = 1_{[y_1, y_3, y_6=1,0,1]}
\]

where appropriate weights \(w_{kj}\) can emulate a contribution to the ensemble vote from this node to the relevant output label nodes \((y_1, y_6)\) – all other output nodes would receive 0 weight from this node. In this example, \(w_k = \left[\frac{1}{H}, 0, \frac{1}{H}, 0, 0, \frac{1}{H}\right]\), supposing \(H\) such functions, \(k = 1, \ldots, H\). In other words, classification is made into a different space. Dembczyński et al. (2012) already mentioned the effect of LP projecting into a higher space. Here we have taken this view wrt a neural network.

We expand on and generalize this view with functions to basis functions from the label space as,

\[
\phi_k(y; S_k, c_k) = 1_{\text{int}(y_{S_k})=c_k}
\]

for random \(S_k \subseteq \{1, \ldots, L\}\), and \(c \in \{0, \ldots, 2^{|S_k|}\}\); and where \(\text{int}(y_{S_k})\) returns an integer representation of the bits of \(y\) indexed by indices \(S_k\), e.g., \(\text{int}([1, 0, 1]_{2,4}) = \text{int}([0, 1]) = 1\).

The resulting network is exemplified in Fig. 9. We then use these nodes to replace the inner chain, just as LP and CC can be seen as different representations of the same distribution. Thus we do \textit{not} use an internal chain like CCASL. However, we do cascade the synthetic labels (i.e., \textit{input} basis functions) as additional input to these nodes: this could be. In fact, it could be drawn alternatively but equivalently as as a four-layer skip-layer network.

This methods allows the particular advantages mentioned of BR (training and testing in parallel, flexibility, adaptability to data-streams). Note in Fig. 9 that each binary output \(y_j\) has its own set of weights, that can be trained in parallel on different machines. Units can be added and removed as with any network. If a label is removed, some of the \(\phi_k\) feature functions may not be \textit{as} relevant, but these can also be removed and replaced over time, for example in a data stream environment.

The structure itself of Fig. 9 can remind us of a neural network formulations. We review several related approaches in the following section.
5. Related Work

This section deals with related literature, not already covered in the Introduction (Section 1).

Adding non-linearity to an otherwise linear method via basis expansions is a fundamental technique in statistical learning (Hastie et al., 2001). Under this methodology, basis functions can be either chosen suitably by a domain expert, or simply chosen arbitrarily to achieve a more dimensioned representation of the input, and higher predictive performance. In this latter case, polynomials are a common choice, such that $\phi = x, x^2, \ldots, x^p$ up to some degree $p$. Under this approach learning can proceed ordinarily by treating $\phi$ as if it were the input.

RBMs are a probabilistic method for finding higher-level feature representations $z = \phi(W^T x)$, where $W$ is learned with gradient-based methods and $\phi$ is some non-linearity (typically a sigmoid function: $\phi(a) = \frac{1}{1+e^{-a}}$, but more recently ReLUs have been particularly popular). The RBM can then cast any input instance stochastically into a different-dimensioned space, as $z$. They are often stacked together into deep neural networks (Hinton and Salakhutdinov, 2006).

Radial Basis Function (RBF) networks, (reviewed in (Zhang, 2009)) build on the idea of using basis functions for non linearity, in a neural network framework, with a linear sum to the output layer variables. Unlike simple polynomial transformation, the parameters for the basis function (a Gaussian is typical) are usually learned with a gradient-based method. Thus, similarly to other neural network approaches, learning is formulated as a minimization of the mean-squared-error function. Both RBM and RBF based networks can be universal approximators of any function.

The multi-layer multi-label neural-network ‘BPMLL’ proposed earlier by Zhang and Zhou (2006) became well known in the multi-label literature, although its out-of-the-box performance was consistently shown to under-compete with the state-of-the-art. Later, a multi-label RBF network was proposed by Zhang (2009). However, this network was not picked up widely in the literature, possibly due to the fiddly ‘hit-and-miss’ nature of the gradient-based learning (wrt the selection of learning rates, hidden units and so on).

Recently Nam et al. (2014) readdressed the application of neural networks for multi-label classification, in particular for the task of large scale text classification. They implemented a number of recently proposed techniques, namely ReLUs, AdaGrad, and dropout. For their selection of text datasets, these networks were shown to be highly competitive compared to the earlier neural network varieties.

Even with the latest techniques, the disadvantage often associated with multi-layer neural networks is that of back propagation typically used for learning, which involves many iterations of propagating the input up through the network to the outputs and then the error back down, so as to adjust weights for learning (i.e., adjusting weights). Even in greedily-stacked deep networks of RBMs, back propagation is used to tune the weights for discriminative prediction. An early alternative to back propagation in neural networks was proposed by W. Thomas Miller III et al. (1990), to use the idea of random functions to project the input layer into a new space in the hidden layer, rather than learn this hidden layer with error propagation.
Table 2: A collection of datasets and associated statistics, where LC is label cardinality: the average number of labels relevant to each example.

| Type   | N    | L    | D    | LC  | Type    |
|--------|------|------|------|-----|---------|
| XOR    | 20   | 3    | 2    | 1.50| logical |
| Music  | 593  | 6    | 72   | 1.87| audio   |
| Scene  | 2407 | 6    | 294  | 1.07| image   |
| Yeast  | 2417 | 14   | 103  | 4.24| biology |
| Medical| 978  | 45   | 1449 | 1.25| medical/text |
| Enron  | 1702 | 53   | 1001 | 3.38| text    |
| Reuters| 6000 | 103  | 500  | 1.46| text    |

More recently, this idea has appeared in the so-called extreme learning machines (ELMs, Huang et al. (2011)), which also project into a new space / hidden layer, as neural networks, but with an important difference (to ordinary neural networks): the weights \( W \) and function parameters are chosen randomly. Again, this is built on the premise of eliminating the need for gradient-based methods in the hidden layer to obtain huge speed ups in learning. RBFs are a typical choice in ELMs, making them closely related to (or in fact, an instance of) RBF networks. ELMs are the most closely related to the methods we propose. However, unlike the methods we propose, ELMs do not work down from the label layer (another difference is that we choose ReLUs over RBFs). In our approach we create random functions from both the input and the output label features.

6. Experiments

All methods were implemented with a combination of the Java framework Meka and in the Python library sklearn. Datasets are listed in Tab. 2.

We use standard multi-label evaluation metrics,

\[
\text{Jaccard Index} := \frac{1}{N} \sum_{n=1}^{N} \frac{|y^{(n)} \cap \hat{y}^{(n)}|}{|y^{(n)} \cup \hat{y}^{(n)}|}
\]

\[
\text{Exact Match} := \frac{1}{N} \sum_{n=1}^{N} [y^{(n)} = \hat{y}^{(n)}]
\]

\[
\text{Hamming Score} := \frac{1}{NL} \sum_{n=1}^{N} \sum_{j=1}^{L} [y_{j}^{(n)} = \hat{y}_{j}^{(n)}]
\]

which are used in many multi-label empirical comparisons. Hamming Score rewards methods for predicting individual labels well, whereas Exact Match rewards a higher proportion of instances with all label relevances correct. An ‘intermediate’ measure the Jaccard Index (often called multi-label accuracy) (Madjarov et al., 2012; Read et al., 2011; Godbole and Sarawagi, 2004).

We carry out ten iterations for each dataset, each time with a random instance order and label order, and a standardized instance space. A 60/40 train/test split is made on
the randomized instances. Tab. 3 shows the results of predictive performance over standard datasets. Fig. 10 shows the effect of different numbers of synthetic and hidden labels for performance on the Scene dataset.

We compare the methods described throughout the paper. From the LP family we use RAkELd: it makes LP scalable by training it only on disjoint subsets of 2—3 labels then combining votes. For all methods we propose, and in addition ELMs, we consider $H = 2L$ ReLU units ($L$ is given for each dataset in Tab. 2. We use Logistic regression is used as the base learner in all cases.

Overall results highlight the performance of two of our proposed methods: CCASL+BR and CAML, particularly the latter, where it is the top performer on most datasets.

Fig. 10 explains the relatively poor performance of ELM in the tables: it needs a relatively huge random hidden layer to be effective. Conversely, CAML outperforms CC already with only small $H$. Recall that we set $H = 2L$ in experiments for Tab. 3. The plot indicates that using more feature function units would increase performance even more, albeit naturally at a computational cost. In any case we see that the performance gap is very consistent between these two.

In Tab. 3 there are only a few cases where CC performs strongly: Music and Yeast and Reuters. It could be that ReLUs are inappropriate basis functions for the input data of these datasets. For Reuters another reason is possible: the advantage of synthetic labels
Table 3: Predictive Performance. Average result across 10 randomizations (of instance and label space) and a 60/40 train/test splits. The average rank is shown in (parenthesis) for each dataset

(a) Jaccard Index

| Dataset | BR | CC | RAkELd | CCASL | CCASL+BR | CAML | ELM |
|---------|----|----|--------|-------|----------|------|-----|
| XOR     | 0.669 (7) | 0.896 (5) | 0.856 (6) | 0.971 (3) | 1.000 (1) | 1.000 (1) | 0.917 (4) |
| Music   | 0.506 (5) | 0.525 (2) | 0.501 (6) | 0.523 (3) | 0.523 (3) | 0.534 (1) | 0.367 (7) |
| Scene   | 0.592 (6) | 0.647 (3) | 0.623 (5) | 0.646 (4) | 0.651 (2) | 0.654 (1) | 0.195 (7) |
| Yeast   | 0.495 (5) | 0.502 (2) | 0.502 (2) | 0.495 (5) | 0.496 (4) | 0.504 (1) | 0.447 (7) |
| Medical | 0.493 (6) | 0.510 (5) | 0.516 (4) | 0.635 (1) | 0.627 (2) | 0.545 (1) | 0.233 (7) |
| Enron   | 0.381 (6) | 0.393 (4) | 0.399 (2) | 0.392 (5) | 0.397 (3) | 0.401 (1) | 0.313 (7) |
| Reuters | 0.394 (5) | 0.432 (1) | 0.388 (6) | 0.422 (2) | 0.422 (2) | 0.397 (4) | 0.327 (7) |
| avg rank | 5.71 | 3.14 | 4.43 | 3.29 | 2.43 | 1.71 | 6.57 |

(b) Exact Match

| Dataset | BR | CC | RAkELd | CCASL | CCASL+BR | CAML | ELM |
|---------|----|----|--------|-------|----------|------|-----|
| XOR     | 0.438 (7) | 0.812 (6) | 0.825 (5) | 0.925 (3) | 1.000 (1) | 1.000 (1) | 0.887 (4) |
| Music   | 0.247 (5) | 0.271 (2) | 0.240 (6) | 0.263 (4) | 0.267 (3) | 0.274 (1) | 0.172 (7) |
| Scene   | 0.501 (6) | 0.580 (3) | 0.567 (5) | 0.577 (4) | 0.593 (2) | 0.596 (1) | 0.171 (7) |
| Yeast   | 0.130 (6) | 0.195 (1) | 0.152 (5) | 0.186 (2) | 0.180 (3) | 0.164 (4) | 0.096 (7) |
| Medical | 0.378 (6) | 0.391 (4) | 0.388 (5) | 0.513 (2) | 0.516 (1) | 0.454 (3) | 0.141 (7) |
| Enron   | 0.111 (6) | 0.118 (1) | 0.112 (5) | 0.117 (4) | 0.118 (1) | 0.118 (1) | 0.066 (7) |
| Reuters | 0.300 (5) | 0.342 (1) | 0.299 (6) | 0.329 (3) | 0.335 (2) | 0.318 (4) | 0.237 (7) |
| avg rank | 5.86 | 2.57 | 5.29 | 3.14 | 1.86 | 2.14 | 6.57 |

(c) Hamming Loss

| Dataset | BR | CC | RAkELd | CCASL | CCASL+BR | CAML | ELM |
|---------|----|----|--------|-------|----------|------|-----|
| XOR     | 0.812 (7) | 0.938 (5) | 0.917 (6) | 0.975 (3) | 1.000 (1) | 1.000 (1) | 0.946 (4) |
| Music   | 0.791 (1) | 0.783 (2) | 0.773 (6) | 0.783 (2) | 0.783 (2) | 0.779 (5) | 0.748 (7) |
| Scene   | 0.883 (6) | 0.886 (3) | 0.885 (5) | 0.886 (3) | 0.889 (1) | 0.887 (2) | 0.834 (7) |
| Yeast   | 0.793 (1) | 0.781 (6) | 0.791 (2) | 0.777 (7) | 0.786 (4) | 0.787 (3) | 0.783 (5) |
| Medical | 0.708 (7) | 0.752 (6) | 0.892 (5) | 0.942 (4) | 0.984 (1) | 0.982 (2) | 0.960 (3) |
| Enron   | 0.920 (7) | 0.928 (5) | 0.933 (3) | 0.928 (5) | 0.936 (2) | 0.939 (1) | 0.932 (4) |
| Reuters | 0.981 (6) | 0.983 (4) | 0.984 (3) | 0.981 (6) | 0.985 (2) | 0.987 (1) | 0.982 (5) |
| avg rank | 5.00 | 4.43 | 4.29 | 4.29 | 1.86 | 2.14 | 5.00 |
Table 4: Average Running Time (seconds) cross ten train/test splits.

| Dataset | BR | CC | RAkELd | CCASL | CCASL+BR | CAML | ELM |
|---------|----|----|--------|-------|----------|------|-----|
| XOR     | 0.002 | 0.002 | 0.019 | 0.005 | 0.007 | 0.006 | 0.002 |
| Music   | 0.054 | 0.055 | 0.392 | 0.145 | 0.159 | 0.410 | 0.016 |
| Scene   | 2.021 | 1.857 | 6.116 | 4.388 | 4.451 | 12.653 | 0.095 |
| Yeast   | 0.695 | 0.872 | 4.991 | 2.784 | 3.030 | 6.021 | 0.285 |
| Medical | 14.835 | 14.651 | 29.026 | 37.932 | 38.850 | 130.226 | 1.054 |
| Enron   | 37.342 | 37.946 | 92.788 | 79.011 | 79.432 | 253.927 | 3.128 |
| Reuters | 102.930 | 118.042 | 256.455 | 317.860 | 327.104 | 703.117 | 62.940 |

diminishes with larger labelsets and datasets, since more labels means a smaller proportion of labels are in the first few positions of the chain, and correspondingly a greater proportion of labels have a higher feature space projection to learn from. Larger numbers of instances means that each label can be learned more thoroughly, and the advantages of additional synthetic labels at the beginning of the chain, or meta labels, are less pronounced.

Running time for these experiments is given in Tab. 4. Incorporates additional hidden units / synthetic labels, our proposed methods are several times more expensive to run on these datasets. Under our parameterization, CAML will always have more inputs on average, namely:

| Method | CC | CAML |
|--------|----|------|
| Inputs | $L \cdot D + \frac{L(L-1)}{2}$ | $D \cdot H + H \cdot L$ |

If $H = L$, then scalability is similar on average, but note that the maximum is different. A model within CAML has at most $\max(H, D)$ inputs, whereas the final model in CC has $D + L - 1$. For increasing $L$, base learners sensitive to the number of attributes will slow down overall performance more so than under CAML. With larger datasets, scalability will become an increasing concern, and a proportion of links must be dropped.

The largest gap between CAML and CC is under Medical, where the former performance roughly 10 times more expensive than the latter. If then extra time was used for trialling chain orders with internal train/test split for CC, it would allow for trialling only around 10 of the factorial (wrt $L = 45$) possible chain orders for this dataset – probably not enough to make up the gap in predictive performance.

On the Logical dataset, CC learns the XOR label only when it occurs in the second or third position (about two thirds of the time). This means it obtains approximately a score of 1 two thirds of the time, and a score of 0.5 the rest of the time. Indeed, $1 \cdot \frac{2}{3} + 0.5 \cdot \frac{1}{3} \approx 0.833$: almost exactly what was attained in practice (0.812). The small difference can be explained by an occasional unfortunate distribution of training examples, which can lead to a poor model even with a good label order. All other methods improve on this, including ELM, even though each of these other models use the same linear base learner, and all labels are conditionally independent (in the true underlying concept). CCASL did not improve over CC as frequently as expected on this dataset (and in general). This perhaps reinforces the idea that there is more at stake than simply ensuring that complicated labels are not at the beginning of the chain. Possibly the fully-cascaded chain is in fact overfitting.
Note that many methods are typically presented in ensemble. In fact, ensembles of classifier chains (ECC) and ensembles of label-powerset such as \texttt{RAkEL}\textsuperscript{5} are among the most common. We avoid an empirical evaluation involving ensembles, since we are more interested in improving the underlying mechanism, rather than compensating for the lack of that mechanism with an ensemble (a reason typically given, explicitly or implicitly, for electing to use an ensemble). Therefore, instead, we took ten randomizations of the datasets to similar effect in terms of evaluation.

7. Summary and Conclusions

Many methods for multi-label classification model dependence explicitly among the label outputs. These methods have performed well in the literature in empirical evaluations, but understanding of their mechanism has lagged behind these results. We contribute to the analysis in this area and looked into this mechanism in greater depth. We explained how modelling dependence among outputs actually only helps when an inadequate base learner is chosen; labels function as advanced feature-transformations, to help the prediction of other labels.

By modelling dependence at inner layers of a network, methods for multi-label classification may assume independence at the outer layer, and we showed how even a simple linear base learner can obtain competitive results in this manner. Furthermore, creating conditional independence of the label space has important practical benefits such as parallel deployments and flexibility wrt data streams.

As a consequence of our exploration of this area, we produced a series of novel methods. Namely, we exploit the fact that in multi-label data, when learning one label, the other labels can be appreciated as nonlinear feature-space projections. This is unlike traditional single-label classification where a single target is predicted for each instance. We leveraged this ambiguity, and experimented with methods that create synthetic labels from a cascade of feature transformations and also derive feature functions in a different space from the labels themselves.

Although we found inspiration in deep-learning and neural-network methods, by leveraging the label space as (an advantage particular to multi-label data), we were able to create multi-layer networks which do not need gradient-based back propagation to train inner units: multiple hierarchical layers are created instantly from combinations of input and output.

On ordinary datasets our automatically created inner layers layers appear to be sufficient. The methods we presented performed related algorithms. For more complex concepts, more layers may help. In future work, we intend to expand our methodology by integrating this work with advanced gradient descent algorithms.

\textsuperscript{5} The difference between \texttt{RAkEL} and our implementation of \texttt{RAkELd} is that the former selects subsets \textit{with replacement} for $M$ models. The latter simply divides the label space into disjoint subsets once, and thus cannot be regarded as a true ensemble method.
References

Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009. doi: 10.1007/s10994-009-5127-5.

Krzysztof Dembczyński, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML ’10: 27th International Conference on Machine Learning*, pages 279–286, Haifa, Israel, June 2010. Omnipress.

Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Mach. Learn.*, 88(1-2):5–45, July 2012. ISSN 0885-6125. doi: 10.1007/s10994-012-5285-8. URL http://dx.doi.org/10.1007/s10994-012-5285-8.

Krzysztof Dembczyński, Willem Waegeman, and Eyke Hüllermeier. An analysis of chaining in multi-label classification. In *ECAI: European Conference of Artificial Intelligence*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 294–299. IOS Press, 2012. ISBN 978-1-61499-097-0.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

Rayid Ghani. Using error-correcting codes for text classification. In *ICML ’00: 17th International Conference on Machine Learning*, pages 303–310, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US. URL http://citeseer.ist.psu.edu/ghani00using.html.

Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *PAKDD ’04: Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30. Springer, 2004.

Yuhong Guo and Suicheng Gu. Multi-label classification using conditional dependency networks. In *IJCAI ’11: 24th International Conference on Artificial Intelligence*, pages 1300–1305. IJCAI/AAAI, 2011.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Geoffrey Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
Guang-Bin Huang, DianHui Wang, and Yuan Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2): 107–122, 2011. ISSN 1868-8071. doi: 10.1007/s13042-011-0019-y. URL http://dx.doi.org/10.1007/s13042-011-0019-y.

Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. Learning and inference in probabilistic classifier chains with beam search. In *ECML-PKDD ’12: 25th European Conference on Machine Learning and Knowledge Discovery in Databases*, volume 7523, pages 665–680. Springer, 2012.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9): 3084–3104, September 2012. ISSN 0031-3203. doi: 10.1016/j.patcog.2012.03.004. URL http://dx.doi.org/10.1016/j.patcog.2012.03.004.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencia, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification - revisiting neural networks. In *ECML-PKDD ’14: 25th European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 437–452, 2014.

Jesse Read and Jaakko Hollmén. A deep interpretation of classifier chains. In *Advances in Intelligent Data Analysis XIII - 13th International Symposium, IDA 2014*, pages 251–262, October 2014. ISBN 978-3-319-12570-1.

Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

Jesse Read, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, 88(1-2):243–272, 2012.

Jesse Read, Luca Martino, and David Luengo. Efficient monte carlo methods for multidimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535–1546, 2014.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

R. Senge, J.J. del Coz, and E. Hüllermeier. Rectifying classifier chains for multi-label classification. In *Lernen, Wissen, Adaption (LWA) 2013*, pages 162–169, 2013.

David A. Sprecher. On the structure of continuous functions of several variables. *Transactions of the American Mathematical Society*, 115:340–355, 1965. ISSN 0002-9947.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.
Lena Tenenboim, Lior Rokach, and Bracha Shapira. Identification of label dependencies for multi-label classification. In MLD '10: 2nd International Workshop on Learning from Multi-Label Data from ICML/COLT 2010, 2010.

Grigorios Tsoumakas and Ioannis Katakis. Multi label classification: An overview. International Journal of Data Warehousing and Mining, 3(3):1–13, 2007.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In ECML/PKDD 2008 Workshop on Mining Multidimensional Data, 2008. URL http://mlkd.csd.auth.gr/publication_details.asp?publicationID=276.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multi-label classification. IEEE Transactions on Knowledge and Data Engineering, 23(7):1079–1089, 2011.

W. Thomas Miller III, Filson H. Glanz, and L. Gordon Kraft III. CMAC: An associative neural network alternative to backpropagation. Proceedings of the IEEE, 78(10):1561–1567, October 1990.

Julio H. Zaragoza, Luis Enrique Sucar, Eduardo F. Morales, Concha Bielza, and Pedro Larrañaga. Bayesian chain classifiers for multidimensional classification. In 24th International Conference on Artificial Intelligence (IJCAI ’11), pages 2192–2197, 2011.

Min-Ling Zhang. ML-RBF : RBF neural networks for multi-label learning. Neural Processing Letters, 29(2):61–74, 2009.

Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In KDD ’10: 16th ACM SIGKDD International conference on Knowledge Discovery and Data mining, pages 999–1008. ACM, 2010. ISBN 978-1-4503-0055-1. doi: 10.1145/1835804.1835930. URL http://doi.acm.org/10.1145/1835804.1835930.

Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering, 18(10):1338–1351, 2006. ISSN 1041-4347.