Journal of Artificial Intelligence Research 70 (2021) 683-718 Submitted 08/2020; published 02/2021

Classifier Chains: A Review and Perspectives

Jesse Read JESSE.READ@POLYTECHNIQUE.EDU
LIX, Ecole Polytechnique
Institut Polytechnique de Paris
Palaiseau 91120. France.

Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Geoff Holmes GEOFF@CS.WAIKATO.AC.NZ
Eibe Frank EIBE@CS.WAIKATO.AC.NZ
University of Waikato,
Hamilton, New Zealand.

Abstract

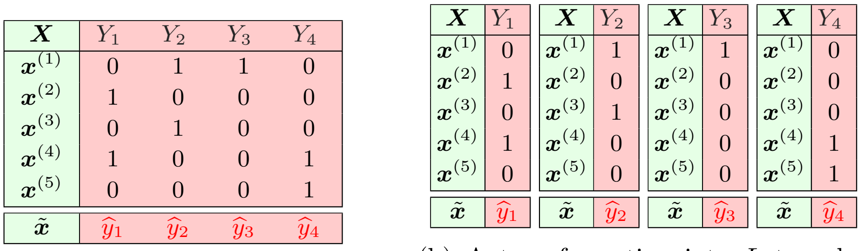
The family of methods collectively known as classifier chains has become a popular approach to multi-label learning problems. This approach involves chaining together off-the-shelf binary classifiers in a directed structure, such that individual label predictions become features for other classifiers. Such methods have proved flexible and effective and have obtained state-of-the-art empirical performance across many datasets and multi-label evaluation metrics. This performance led to further studies of the underlying mechanism and efficacy, and investigation into how it could be improved. In the recent decade, numerous studies have explored the theoretical underpinnings of classifier chains, and many improvements have been made to the training and inference procedures, such that this method remains among the best options for multi-label learning. Given this past and ongoing interest, which covers a broad range of applications and research themes, the goal of this work is to provide a review of classifier chains, a survey of the techniques and extensions provided in the literature, as well as perspectives for this approach in the domain of multi-label classification in the future. We conclude positively, with a number of recommendations for researchers and practitioners, as well as outlining key issues for future research.

1. Introduction

Interest in multi-label classification has grown at an explosive pace in the last 10 years, from only a few dozen explicit mentions in the scientific literature to hundreds of new papers per year, a significant collection of benchmark datasets, and a number of dedicated software frameworks. Applications are as diverse as those found in multi-class classification, and several families of methods have emerged. Reviews of the area are given by, for example, Zhang and Zhou (2014), and in the broader context of multi-output learning, by Waegeman et al. (2019).

The defining aspect of multi-label learning is the association of multiple binary class labels to a single instance. A multi-label dataset can be denoted $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N}$, consisting of N examples. Each i-th instance $\boldsymbol{x}^{(i)}$ is associated with a label vector $\boldsymbol{y}^{(i)} = [y_1^{(i)}, \ldots, y_L^{(i)}]$; there are L elements corresponding to L label concepts, and each element $y_j^{(i)} \in \{0, 1\}$ indicates the relevance (if 1) or not (if 0) of the j-th concept to this instance. A multi-label model is tasked with providing predictions $\widehat{\boldsymbol{y}} = [\widehat{y}_1, \ldots, \widehat{y}_L]$ for any given test instance $\tilde{\boldsymbol{x}}$. Note that traditional multi-*class* learning also considers L concepts but, in contrast, only a single concept can be relevant to any particular instance. In both types of learning, each instance is typically multi-dimensional, described by D features (also referred to as attributes).

Figure 1a shows an example of a multi-label dataset. Figure 1b shows how this dataset can be naturally transformed, or reduced, into L binary problems that are solved independently. This approach, of applying independent binary classifiers, is known as the *binary relevance* method, which has become a typical baseline in multi-label studies.



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

(a) A multi-label dataset, with test instance $\tilde{\boldsymbol{x}}$.

$\boldsymbol{X}$	Y_1
$\boldsymbol{x}^{(1)}$	0
$\boldsymbol{x}^{(2)}$	1
$\boldsymbol{x}^{(3)}$	0
$\boldsymbol{x}^{(4)}$	1
$\boldsymbol{x}^{(5)}$	0
$\tilde{\boldsymbol{x}}$	$\widehat{y}_1$

$\boldsymbol{X}$	Y_2
$\boldsymbol{x}^{(1)}$	1
$\boldsymbol{x}^{(2)}$	0
$\boldsymbol{x}^{(3)}$	1
$\boldsymbol{x}^{(4)}$	0
$\boldsymbol{x}^{(5)}$	0
$\tilde{\boldsymbol{x}}$	$\widehat{y}_2$

$\boldsymbol{X}$	Y_3
$\boldsymbol{x}^{(1)}$	1
$\boldsymbol{x}^{(2)}$	0
$\boldsymbol{x}^{(3)}$	0
$\boldsymbol{x}^{(4)}$	0
$\boldsymbol{x}^{(5)}$	0
$\tilde{\boldsymbol{x}}$	$\widehat{y}_3$

$\boldsymbol{X}$	Y_4
$\boldsymbol{x}^{(1)}$	0
$\boldsymbol{x}^{(2)}$	0
$\boldsymbol{x}^{(3)}$	0
$\boldsymbol{x}^{(4)}$	1
$\boldsymbol{x}^{(5)}$	1
$\tilde{\boldsymbol{x}}$	$\widehat{y}_4$

(b) A transformation into L two-class datasets to which independent binary classifiers can be applied.

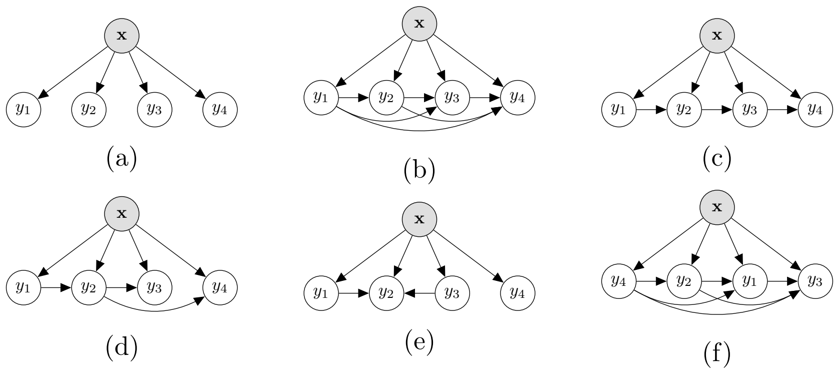
Figure 1: Illustration of how independent classifiers can be applied to a multi-label classification problem by transformation (often explicitly called a reduction) into separate datasets. Each instance is also a vector, not expanded for notational simplicity. The goal of a model is to predict all labels $\widehat{y}_j | j = 1, \ldots, L$ for a given test instance $\tilde{\boldsymbol{x}}$.

The method of *classifier chains* was described by Read et al. (2009) (later, with an extended analysis, by Read et al., 2011), and also proposed contemporaneously by Kajdanowicz and Kazienko (2009). The idea is simple: connect binary classifiers in a 'chain', such that the output prediction of one classifier is appended as an additional feature to the input of all subsequent classifiers. This method is one of many approaches that seeks to model relationships between labels, thus obtaining improved performance over the binary relevance approach. There are now dozens of variants and analyses of classifier chains, and the method has been involved in at least a hundred empirical evaluations, often proving among the most competitive, even in relatively recent comparisons (Moyano, Galindo, Cios, & Ventura, 2018). On the grounds of such interest, much of which is still ongoing, the goal of this paper is to provide an overview of landmark developments and analyses, and also to discuss perspectives exemplified by various methods that have been proposed.

For the purposes of our investigation, we define classifier chains under the following two properties: 1) one classifier per label, considered as a node in a *chain*, where 2) the chain is any directed acyclic structure in which the output of one classifier becomes input to the subsequent classifiers to which it is connected in that structure. This is a broader definition than in the initial work proposing classifier chains, which explicitly considered a fully-connected cascade; to afford us greater flexibility to follow more recent developments in the same context. Arguably, we could speak of "directed acyclic graphs of classifiers" but we retain the terminology of a 'chain' in line with the bulk of the related literature. Indeed, the flexibility of this method is certainly one of the main factors behind its popularity, and the large number of variants offered in the literature.

If we consider chains which are not connected at all, then we recover the binary relevance method. We remark that "binary relevance", although typically denoting independent classifiers, can be considered itself a family of methods that encompass the full spectrum of classifier chains (Burkhardt & Kramer, 2015; Zhang, Li, Liu, & Geng, 2018) from independent classifiers to fully-connected chains, without conflict of terms. An important concept is the hyperparametrization of *base classifiers*; here, any suitable binary classifier (e.g., logistic regression, decision trees, support vector machines) can be considered. In the deterministic sense, the j-th binary classifier, given an input instance $\boldsymbol{x}$, produces predictions $\widehat{y}_j \in \{0, 1\}$ indicating the relevance of each of the j-th labels as it pertains to that instance. For probabilistic varieties of chains, discussed in Section 3.1, we should additionally assume that a base classifier provides a probabilistic interpretation of this decision. It is worth noting that the classifier chain then corresponds to an appropriately structured Bayesian network in which the base classifiers define the conditional probability distributions at each node of the network.

Graphically, the original formulation of classifier chains can be drawn as in Figure 2b, as a fully connected chain. One could also refer to the fully connected structure as a *cascade* or a *fan* to distinguish from a Markov chain where each node is connected only to the previous node and following node – which is also a possible configuration in this context (see Figure 2c, and, for example, Read, Martino, & Hollmén, 2017). Other variants are also possible, such as trees (for example, Figure 2d, and Enrique Sucar et al., 2014), and arbitrary directed acyclic graphs (DAGs) (for example, Figure 2e, and Zhang & Zhang, 2010). The mechanism of all these configurations is the same: the incoming edges to the j-th node represent features to the j-th classifier, and the outgoing edge its prediction. Figure 2a is equivalent to binary relevance (and thus, Figure 1b). Even though undirected graphical models are related (further discussion in Section 5), we consider those to belong to a separate class of model due to the different inference strategies involved.



(a) (b) (c)

(d) (e) (f)

Figure 2: Different classifier chain structures for a problem with 4 labels. Note the difference between (b) and (f) is the order of labels.

Figure 3 shows the dataset transformation corresponding to Figure 2b (which can be contrasted with that of independent models; Figure 2a and Figure 1b); one dataset per node/label. It is straightforward to make transformations for the other cases exemplified in Figure 2 in a related fashion. Any binary classifier can be employed on the transformed datasets.

$\boldsymbol{X}$	Y_1
$\boldsymbol{x}^{(1)}$	0
$\boldsymbol{x}^{(2)}$	1
$\boldsymbol{x}^{(3)}$	0
$\boldsymbol{x}^{(4)}$	1
$\boldsymbol{x}^{(5)}$	0
$\tilde{\boldsymbol{x}}$	$\widehat{y}_1$

$\boldsymbol{X}$	Y_1	Y_2
$\boldsymbol{x}^{(1)}$	0	1
$\boldsymbol{x}^{(2)}$	1	0
$\boldsymbol{x}^{(3)}$	0	1
$\boldsymbol{x}^{(4)}$	1	0
$\boldsymbol{x}^{(5)}$	0	0
$\tilde{\boldsymbol{x}}$	$\widehat{y}_1$	$\widehat{y}_2$

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

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

Figure 3: The transformation of a dataset (that of Figure 1a) for the application of classifier chains (as shown in Figure 2b). It is worth highlighting that (as in Figure 1) the 0/1 values shown in the tables correspond to the values $y_j^{(i)}$ found in the training data; whereas $\widehat{y}_1, \ldots, \widehat{y}_L$ (in the final row) are all predicted values *not* available at training time.

Classifier chains have obtained state-of-the-art performance in many empirical evaluations, including a variety of datasets and evaluation metrics (see, for example, Madjarov et al., 2012; Moyano et al., 2018, and references therein). This strong off-the-shelf performance, their simplicity of implementation, and the open choice of base classifiers to fit many preferences and suitability to different domains, has led to their wide usage and ongoing development. A search in the academic literature reveals they have been used in diverse applications ranging from vision and natural language domains, to bioinformatics, health analytics, time series, and route forecasting. Although we do not explicitly tackle this setting in this manuscript, the method of classifier chains (unlike many other multi-label methods) generalizes easily to the case of having multiple (more than two) values per label, a task sometimes called multi-label multi-class, multi-dimensional, or multi-objective classification, see, for example, the work by Kocev et al. (2007), Read et al. (2014).

In addition to attracting interest from practitioners on account of their performance, classifier chains have also raised many questions of a theoretical nature: How can their efficacy be explained? And what do the learning algorithms for these chains optimize? (We look at these issues in Section 3). Tied in with this are further questions that have driven much related work: Is there an optimal chain order and, if so, how to find it? (This is covered in Section 4). Can chains be seen as a special case of other methods (or vice versa)? (See our discussion of related work in Section 5). Following an attempt at addressing these questions, we provide perspectives and discuss open issues in Section 6, and give recommendations to practitioners in Section 7, before concluding (Section 8).

2. Preliminaries: Multi-label Learning

The general concept of multi-label classification and classifier chains has been introduced in Section 1 in terms of mapping instances $\boldsymbol{x}$ to label-vectors $\boldsymbol{y}$. Here we provide additional notation and concepts which are relevant to a more in-depth and theoretical discussion.

Beginning with a binary *single-label* problem, we seek a classifier h to map instances $\boldsymbol{x}$ to $y \in \{0, 1\}$. It should carry out this mapping, i.e., make a decision/classification, according to the loss function of interest. A common objective is to minimize 0/1 loss

$$\ell_{0/1}(y,\widehat{y}) = [\![y \neq \widehat{y}]\!] = \begin{cases} 0 & y = \widehat{y} \text{ (inputs match exactly)} \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

which is equivalent to maximizing classification accuracy. In this paper we use $\mathcal{J}$ to denote a payoff function such as accuracy, where higher is better; note that $\ell_{0/1}(y,\widehat{y}) = 1 - \mathcal{J}_{\mathsf{acc.}}(y,\widehat{y})$. Minimizing the 0/1 loss implies that in a probabilistic setting

$$\widehat{y} = h(\boldsymbol{x}) = \operatorname*{argmax}_{y \in \{0,1\}} P(y|\boldsymbol{x}) \tag{2}$$

i.e., the mode of the posterior distribution (maximum a-posteriori, or MAP estimate).

In multi-label classification, there are 2^L possible classifications (corresponding to distinct combinations/subsets of L labels), rather than just 2, hence:

$$\widehat{\boldsymbol{y}} = [\widehat{y}_1, \ldots, \widehat{y}_L] = h(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \{0,1\}^L} P(\boldsymbol{y}|\boldsymbol{x}) \tag{3}$$

We use the notation $P(\boldsymbol{y}|\boldsymbol{x}) \equiv P(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{X} = \boldsymbol{x})$, corresponding to a single number, i.e., $P(\boldsymbol{y}|\boldsymbol{x}) \in [0,1]$, to represent the probability that $y_1, \ldots, y_L$ are the label relevance indicators, given test instance $\boldsymbol{x}$.

The 0/1 loss can be normalized, providing a value $\in [0,1]$ for N training examples as

$$\ell_{0/1}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \ell_{0/1}(\boldsymbol{y}^{(i)}, \widehat{\boldsymbol{y}}^{(i)}) \tag{4}$$

with training set $\mathcal{D}$, where $\ell_{0/1}(\boldsymbol{y}^{(i)}, \widehat{\boldsymbol{y}}^{(i)})$ – referring to Eq. (1) – returns 0 only if each single bit matches exactly. (thus often referred to as 'exact match' in payoff form; $\ell_{0/1}(\mathcal{D}) = 1 - \mathcal{J}_{\mathsf{exact\text{-}match}}(\mathcal{D})$). We may rewrite explicitly

$$\ell_{0/1}(\boldsymbol{y},\widehat{\boldsymbol{y}}) = [\![\boldsymbol{y} \neq \widehat{\boldsymbol{y}}]\!] = \begin{cases} 0 & \sum_{j=1}^{L} [\![y_j = \widehat{y}_j]\!] = L \\ 1 & \text{otherwise} \end{cases}$$

The sum over indices $i = 1, \ldots, N$ is allowed by the assumption of independent and identically distributed samples in the training set.

The choice of this loss coincides with the general intuition that we should model labels together, since an incorrect combination of labels by any degree gives worst possible loss for that instance. As such it has motivated and justified many methods. It also implies exponential complexity (if all 2^L label combinations are to be modeled explicitly), therefore providing a major challenge to overcome. Classifier chains form one family of such methods.

There are numerous (more than a dozen) often-used multi-label loss metrics; some of them are drastically different from each other. For example, another loss frequently referred to in the multi-label literature is Hamming loss:

$$\ell_H(\mathcal{D}) = \frac{1}{NL} \sum_{i=1}^{N} \sum_{j=1}^{L} \ell_{0/1}(y_j^{(i)}, \widehat{y}_j^{(i)}) = \frac{1}{L} \sum_{j=1}^{L} \left(\frac{1}{N} \sum_{i=1}^{N} \ell_{0/1}(y_j^{(i)}, \widehat{y}_j^{(i)}) \right) \tag{5}$$

Even if expressed here in terms of (single-label) 0/1 loss, Hamming loss can be considered on the opposite end of the spectrum to 0/1 loss as a multi-label metric, and indeed indicates that we can model all labels individually and independently, as we have explicitly expressed on the right hand side; i.e., it can be tackled by independent binary classifiers – via minimizing Eq. (1) on a per-label basis.

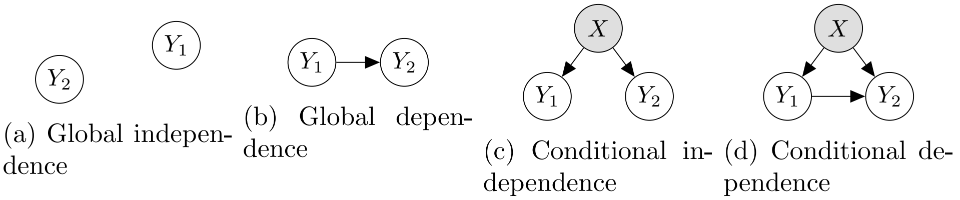
What does it mean to model labels together (or not)? Statistically, we can speak of modeling label dependence. We may speak of global dependence (or marginal/unconditional dependence) between two label variables Y_1 and Y_2 when

$$P(\boldsymbol{Y}) = P(Y_1)P(Y_2|Y_1) = P(Y_2)P(Y_1|Y_2) \neq P(Y_1)P(Y_2)$$

and conditional dependence (conditioned on input $\boldsymbol{x}$) implies that

$$P(\boldsymbol{Y}|\boldsymbol{x}) = P(Y_1|\boldsymbol{x})P(Y_2|Y_1,\boldsymbol{x}) = P(Y_2|\boldsymbol{x})P(Y_1|Y_2,\boldsymbol{x}) \neq P(Y_1|\boldsymbol{x})P(Y_2|\boldsymbol{x}) \tag{6}$$

(we use upper case $\boldsymbol{Y} = [Y_1, Y_2]$ here to make explicit that these inequivalences can hold for any combinations $[y_1, y_2] \in \{0,1\}^2$). Figure 4 provides visual intuition in the form of graphical models. Also note that an in-depth introduction to these concepts is given in the context of multi-label learning by Dembczyński et al. (2012a), and we refer the reader there for a thorough treatment.



(a) Global independence (b) Global dependence (c) Conditional independence (d) Conditional dependence

Figure 4: Different types of dependence w.r.t. Y_1 and Y_2; shown here as Bayesian networks.

Note that in Eq. (6) the second and third terms are equivalent expansions of the joint full conditional (left term). This means that Figure 4d (also Figure 4b) showing dependence can be equivalently drawn with the arrow pointing from Y_2 to Y_1. More generally, for L labels, conditional dependence implies

$$\begin{aligned} P(\boldsymbol{Y}|\boldsymbol{x}) &= P(Y_1|\boldsymbol{x})\prod_{j=2}^{L} P(Y_j|\boldsymbol{x}, Y_1, \ldots, Y_{j-1}) \qquad (7) \\ &\neq \prod_{j=1}^{L} P(Y_j|\boldsymbol{x}) \end{aligned}$$

and again here, any permutation of $j = 1, \ldots, L$ is valid.

The expansion into conditional distributions is convenient for expression as a Bayesian network, such as those drawn in Figure 2, where directed edges show the conditional dependence. Eq. (7) specifically captures full dependence corresponding to either Figure 2b or Figure 2f (among other possible label orders). Sparser structures can model sparser dependence relations, by replacing $Y_1, \ldots, Y_{j-1}$ with the corresponding parents of node Y_j.

Thus far, we have provided the groundwork for motivating classifier chains: performance metrics such as 0/1 loss which imply modeling labels together (i.e., linking them in the chain structure). Further, we have briefly reviewed label dependence, a theoretical mechanism upon which to view such an approach. Although Hamming loss entails the goal of high performance on each label individually and independently (rather than jointly together) – such a goal should also be a goal of a multi-label method – and typically will correlate with it; sometimes the minimizer will even coincide with that of 0/1 loss (Dembczyński et al., 2012a); an obvious example is where classification can be carried out perfectly and all losses are 0.

Table 1: A toy training dataset with empirical distributions. In the conditional distributions specifically exemplified here, test instance $\tilde{\boldsymbol{x}} = [1,1]$, and $\widehat{y}_1 = 1$. Notice that either $\widehat{\boldsymbol{y}} = [0,1]$ or $\widehat{\boldsymbol{y}} = [1,0]$ fullfil Eq. (3) for minimizing 0/1 loss, yet minimizing Hamming loss suggests $[0,0]$ according to Eq. (2) per label. By comparing $P(\cdot)$ and $P(\cdot|\tilde{\boldsymbol{x}})$ in tables (b), (c) and (d), we find both global dependence and conditional dependence under Eq. (6); but we must recall that these P are only empirical estimations via counting from training dataset (a) ($\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N}$ where $L = 2, N = 8$); specifically $P(\boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N}[\![\boldsymbol{y}^{(i)} = \boldsymbol{y}]\!]$ and $P(\boldsymbol{y}|\tilde{\boldsymbol{x}}) = P(\boldsymbol{y}, \tilde{\boldsymbol{x}})/P(\tilde{\boldsymbol{x}})$ where $P(\boldsymbol{y}, \tilde{\boldsymbol{x}}) = \frac{1}{N}\sum_{i=1}^{N}[\![\boldsymbol{y}^{(i)} = \boldsymbol{y}]\!][\![\boldsymbol{x}^{(i)} = \tilde{\boldsymbol{x}}]\!]$; e.g., $P([1,0]|\tilde{\boldsymbol{x}}) = (\frac{1}{8}\cdot 2)/\frac{1}{2} = 0.5$; and, i.e., not necessarily equivalent to either underlying ground truth or test data distributions. Note that the tables can be understood in conjunction with Figure 4.

$\boldsymbol{X}$	Y_1	Y_2
$[0,1]$	1	1
$[0,1]$	0	0
$[1,0]$	0	0
$[0,0]$	0	0
$[1,1]$	1	0
$[1,1]$	1	0
$[1,1]$	0	1
$[1,1]$	0	1

(a) Training dataset

$\boldsymbol{y}$	$P(\boldsymbol{y})$	$P(\boldsymbol{y}|\tilde{\boldsymbol{x}})$
$[0,0]$	0.375	0.
$[0,1]$	0.25	0.5
$[1,0]$	0.25	0.5
$[1,1]$	0.125	0.

(b) Estimated distributions over $\boldsymbol{y}$

y_1	$P(y_1)$	$P(y_1|\tilde{\boldsymbol{x}})$
0	0.625	0.5
1	0.375	0.5

y_2	$P(y_2)$	$P(y_2|\tilde{\boldsymbol{x}})$
0	0.625	0.5
1	0.375	0.5

(c) Estimated distributions over y_j

y_2	$P(y_2|\widehat{y}_1)$	$P(y_2|\tilde{\boldsymbol{x}}, \widehat{y}_1)$
0	0.67	1.
1	0.33	0.

(d) Estimated distributions over $y_2|\widehat{y}_1$

At this point, however, it is crucial to emphasise that the equivalence in Eq. (6), denoting label dependence, holds only for ground-truth P. In real-world problems, we do not have true underlying distributions P and instead have to approximate them, typically in a data-driven fashion (learning from the training data). Table 1 shows example empirical distributions obtained by counting of co-occurrence frequencies from training data. Different base classifiers will embody different approximations, but whichever model class we choose, it is unlikely that these estimated P will be the ground truth. This departure (from an idealized ground-truth model) is furthermore exacerbated by approximate inference (which is increasingly needed for larger numbers of labels, given the exponential increase in label combinations). This issue has major implications regarding the design of classifier chains; it raises important questions regarding their performance and choice of chain structure, and lays the basis for a discussion of the complex relationship between chain structure and performance under different loss functions. We address these questions in the following sections of this article.

3. How Classifier Chains Work

In this section, we explore the mechanism of classifier chains, looking to explain reasons for their efficacy on multi-label problems.

As with most machine learning methods, there are many angles from which to approach and view classifier chains. We elaborate on two of these in the following subsections (similarities to other methods will be approached in Section 5). We choose these two because they illustrate two aspects leading to the success of chaining. In particular, the first (Section 3.1) on account of its appealing probabilistic interpretation and how this relates to strong predictive performance particularly under 0/1 loss. The second (Section 3.2) is mainly about explaining strong performance of chains under Hamming loss and improving performance in general, by producing a stronger feature representation via chained labels. Indeed, if a multi-label concept is well modeled, predictive performance will be strong across many loss metrics or even – as mentioned in Section 2 – equivalent. Metrics may also be blended or used as surrogates for each other (Park & Read, 2018).

In this section, we consider the fully-cascaded chain where each j-th base classifier provides prediction

$$\widehat{y}_j = h_j(\boldsymbol{x}, \widehat{y}_1, \ldots, \widehat{y}_{j-1}) \tag{8}$$

but this can easily be generalized to any directed acyclic structure by replacing indices $1, \ldots, j-1$ with the indices corresponding to the parent nodes of y_j.

To build h_j, one uses instances $\{[\boldsymbol{x}^{(i)}, y_1^{(i)}, \ldots, y_{j-1}^{(i)}]\}_{i=1}^N$ formed from the training data as input, and corresponding $\{y_j^{(i)}\}_{i=1}^N$ as output. From this perspective, binary base classifiers can then be trained in an off-the-shelf manner. We remind the reader that, conversely, predictions $\widehat{y}_j$ (or their estimated probabilities) are *not* used (or known) at training time (refer to Figure 3 for illustration) – at prediction time they are imputed recursively by the model via Eq. (8).

It has been noticed by Senge et al. (2013), and is worth remarking here, that this common way of training classifier chains, using the training labels as inputs, breaks a common assumption of statistical learning, namely that the distribution of the training data and test data should be identical. This is because at test time, the *predicted* labels are used as inputs, yet these are not from the same distribution as the training data, as that distribution is not known, and can only be approximated by the predictive distribution used on test data. Further discussion on this special case is deferred to Section 5.3.

3.1 Classifier Chains as Probabilistic Models

The formalization of *probabilistic classifier chains* was proposed by Dembczyński et al. (2010). The training process is identical to the 'standard' formulation, but an additional requirement of the base classifiers h_j is that they have a probabilistic interpretation (at least in the loose sense of a prediction $\in [0, 1]$ that could be understood as a confidence):

$$\widehat{y}_j = h_j(\boldsymbol{x}) := \operatorname*{argmax}_{y_j \in \{0,1\}} P(y_j|\boldsymbol{x}, y_1, \ldots, y_{j-1}) \tag{9}$$

With these models (or, more precisely, their probabilistic components), inference can be phrased as a maximum a-priori (MAP) estimate, where Eq. (3) is developed as follows:

$$\widehat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \{0,1\}^L} P(y_1|\boldsymbol{x}) \prod_{j=2}^{L} P(y_j|\boldsymbol{x}, y_1, \ldots, y_{j-1}) \tag{10}$$

This corresponds to a minimization of the 0/1 loss (Eq. (4)), since the MAP estimate Eq. (10) is the minimizer for this loss. Note, particularly, the expansion of the joint probability from Eq. (7) in correspondence with Figure 2b.

Figure 5 illustrates this idea as a probability tree where each path from root to leaf represents one combination of labels $\boldsymbol{y} \in \{0, 1\}^L$, associated with probability $P(\boldsymbol{y}|\boldsymbol{x})$, factored across branches as per Eq. (10). Under this view, deterministic inference in classifier chains as originally proposed by Read et al. (2009), follows a single path greedily through the tree. This is an inexpensive and arguably crude approximation of Eq. (10) to the extent that one can only talk of the method being a mode-*seeker* (Dembczyński et al., 2010), that is to say it seeks out the MAP estimate[1] in a greedy way, but is not guaranteed to find it. On the other extreme, exploring all 2^L paths provides a Bayes-optimal inference solution, corresponding to the highest payoff (minimal 0/1 loss). While being a faithful evaluation of Eq. (10), this exhaustive search is generally intractable, provoking the application of tree search methods to efficiently trial a subset of paths, thereby approximating the best solution at reduced cost: for example, Monte Carlo search, ϵ-approximate, beam search, or even A* search (the work by Mena et al., 2016 provides a comprehensive survey; with detailed references and empirical results). Note also that, although typically set up to minimize 0/1 loss, probabilistic classifier chains can be used with any loss function (Dembczyński, Waegeman, & Hüllermeier, 2012b) including Hamming loss, unlike the original greedy inference which seeks out the 0/1 minimizer in a non-optimal way. We can also see that goodness-of-fit depends on chain structure, since if it is reduced to that of Figure 2a, chains will generally not be able to minimize 0/1 because label dependencies cannot be modeled. We defer a deeper discussion on chain structure to Section 4.

Although classifier chains may be configured to optimize different losses, 'by default' when making a prediction (supposing adequate connectivity and no steps taken to the contrary) they are seeking a minimization of 0/1 loss (greedily, exhaustively, or on some trade-off) for that prediction, using the given structure and test instance. This explains the typically high performance under this and related metrics. Recall that, although the base models are induced independently (i.e., the training process for h_j is not dependent on the outcome of training $h_1, \ldots, h_{j-1}$ and can be carried out separately and even in parallel – including the probabilistic chains), this is *not* the case at test time, where a model's inference procedure is explicitly dependent on the inference of the previous labels. Moreover, training is done iteratively over labels, typically leading to a 0/1-loss minimization

[1] One may remark that, with regard to the first label in the chain y_1 (but only that one) it can be said to be minimizing Hamming loss

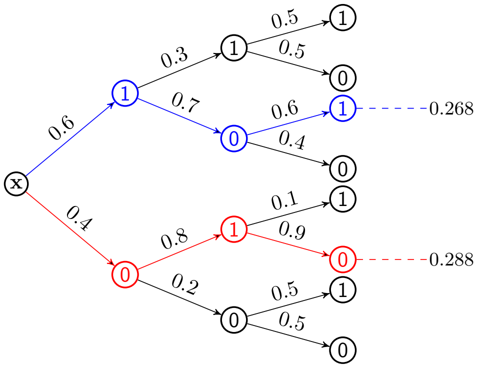


Figure 5: A probability tree corresponding to a fully-connected classifier chain (like Figure 2b) over three labels. The marginal $p(y_j|\boldsymbol{x},\ldots)$ is shown on branches from level $j-1$ to j, whereas $P(\boldsymbol{y}|\boldsymbol{x})$ is given for each full path. Note that in this example the nodes (labels) of the best path, $P([0,1,0]|\boldsymbol{x}) = 0.288$, are not the same as those taken by 'standard' (greedy) classifier chains, which picks $P([1,0,1]|\boldsymbol{x}) = 0.268$. Note also that there are 2^L paths in total ($2^3 = 8$ in this case).

or some approximation thereof. Given a particularly useful/powerful (and possibly large) set of input features $\boldsymbol{x}$, the additional predictive benefit of the extra label features may be negligible – this is an informal way of stating that labels are approximately *conditionally* independent of each other. In such a scenario, the minimizer for 0/1-loss and Hamming loss is equivalent, and the 'multi-label nature' of the problem ceases to exist. However, it also implies that the problem is 'easy' – studies invariably measure some form of conditional label dependence; as we commented above at the beginning of this section.

3.2 Classifier Chains as Neural Networks

One might assume that if (probabilistic) classifier chains are set up to minimize 0/1 loss, they will not show statistically significant improvement compared to independent classifiers in cases where all label concepts are being evaluated independently, such as under Hamming loss. However, it is widely shown that classifier chains *do* in fact often outperform independent classifiers when labels are evaluated independently, even when the base model class is identical (see, for example, empirical studies by Madjarov et al., 2012; Moyano et al., 2018; Rivolli et al., 2020).

This apparent contradiction is resolved under a different conceptualisation of the base classifiers. If we consider that models $h_1, \ldots, h_{j-1}$ are *part of* the j-th model h_j, then the performance gain can be explained in terms of earlier labels offering themselves as a feature space expansion for later labels in the chain (Dembczyński et al., 2012b; Read & Hollmén, 2017). Figure 6 makes this explicit by showing classifier chains as a feed-forward multi-layer neural network (where base classifiers take the place of activation functions).

The result is comparable to the way latent nodes behave inside a standard multi-layer neural network. One can recall the case of modeling the xor function, which is very well known in the neural network community (see, for example, Goodfellow, Bengio, & Courville, 2016) for requiring a hidden layer for correct modeling. Figure 7 contrasts a typical failed

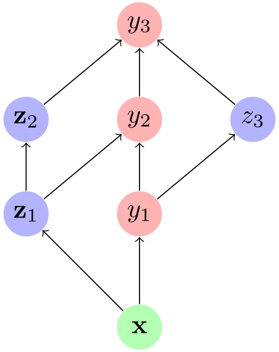


Figure 6: Greedy inference in a classifier chain as a forward pass in a neural network. Base classifiers take the place of activation functions into y_j and identity functions to z_k, e.g., $z_3 = f(y_1) = y_1$ (can also be seen as delay nodes). Note that z-variables are thus either vectors or scalars depending on input. There is no implication of back propagation in the training process. Note y_1 and y_2 can be viewed as part of the model for predicting y_3.

attempt of a linear classifier to achieve separability of points w.r.t. their XOR class labels vs. a successful solution in a classifier chain where separation is achieved via leveraging of an earlier label prediction (OR, in this example). We further point out that even a base classifier with a linear decision boundary provides a non-linearity via its decision function. For example, logistic regression has a sigmoid function, and uses a threshold to map into discrete class output, $\widehat{y}_j \in \{0, 1\}$.

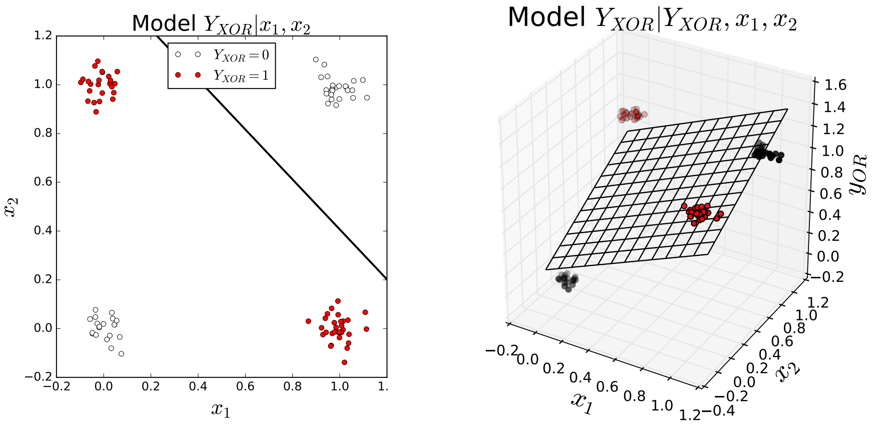


Figure 7: The XOR label cannot be separated by a linear decision boundary in $\boldsymbol{x} \in \mathbb{R}^2$ space (left), but can be separated with an expansion to the feature space in the form of the XOR label (right; vertical axis). Note that jitter has been added to the points $\boldsymbol{x} \in \{0, 1\}^2$ for purposes of illustration.

If classifier chains may be considered as a deep neural network of $L + 1$ layers, this is only in respect of the forward pass. A full analogy to neural networks is limited by the fact that the nodes of a classifier chain are not latent nodes in the true sense; they are already exemplified in the training set as training labels. We cannot claim deep *learning*, as we do not back-propagate through the base classifiers. Hence, one may argue that classifier chains should be considered as 'feature-space expanders' rather than neural networks, however an analogy with elementary basis-function expansion is also limited since the formation of these functions (in chains) is data driven and not expert driven. A deep network has been leveraged and extended with synthetic/artificial labels and traditional hidden layers, for example, by Read and Hollmén (2017), and Cisse, Al-Shedivat, and Bengio (2016); which can be seen as developing a network deep in the *label space* (as opposed to in the input space). Particularly the idea of adding "synthetic" labels to a chain blurs the line with the concept of basis expansions (arbitrary non-linear functions, often polynomials or radial functions). We refer again to this discussion later in Table 3.

It is also possible to pass probabilistic information from the marginal posterior on labels down the chain instead of a hard classification, where each node emits real-valued probabilities $P(Y_j = 1|\boldsymbol{x}, \ldots) \in [0, 1]$ instead of $\widehat{y}_j = h_j(\boldsymbol{x}) = \text{argmax}_{y_j \in \{0,1\}} P(Y_j = y_j|\boldsymbol{x}, \ldots))$, according to the argument that this additional information on prediction uncertainty/confidence is more expressive than a binary value. Under the view as a forward pass through a neural network, this is a question of activation function/non-linearity as represented by each base classifier.

Activation functions in a neural network are inherently simplistic, since it is the greater network that embodies the necessary complexity. In a related way, a classifier chain (i.e., a network of base classifiers) is relatively less effective if we select a powerful non-linear model class for these classifiers such as decision trees or even ensemble models (Read & Hollmén, 2017); the stronger base learners make much of the connectivity in the chain redundant. This is particularly so under Hamming loss, where the gain of chains compared to independent models may in theory be reduced to zero. However, in practice the effect varies greatly depending on the dataset and base classifier parametrization (Rivolli et al., 2020). This is an important issue in classifier-chain design. In any case, this question of connectivity leads us to the general question of how specifically to order or structure the chain around its label nodes.

4. The Question of Chain Structure

Chain structure is a major issue, because it affects major aspects of the classification model including, notably, predictive performance, computational scalability, and interpretation. Note that we consider a variation in the order of nodes a specific variation of structure (no need to consider them separately, as in some work).

The full-chain factorization (as in Eq. (10)) is in fact valid and equivalent for any order of labels (see also Eq. (6) in Section 2) however, as explained earlier, this refers only to the case where P is the ground-truth distribution and when the inference is exhaustive. In practice (where these conditions do not hold) it is found that chain order has significant effect. This has been empirically found in the literature (for example, by Read et al., 2014, among others) and we additionally illustrate this here with the results in Figure 8: the variability of performance among different chains is clearly exhibited.

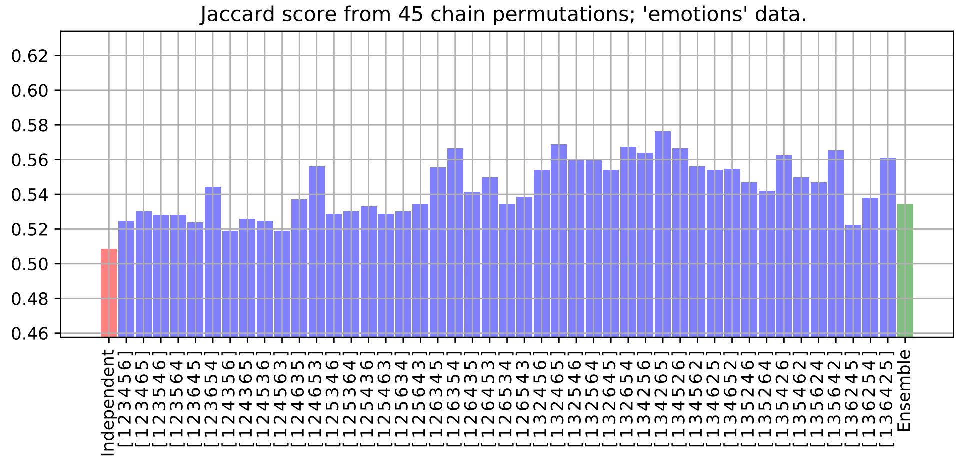


Figure 8: Jaccard score on the Emotions data for classifier chains (base classifier: logistic regression) for the first 45 of the 720 possible chain orders compared. Also included are the Jaccard scores obtained using independent classifiers and an ensemble of all 45 chains, where the latter predicts the majority labels across all 45 chains.

Discussions of chain order frequently relate to the issue of *error propagation* (Senge, del Coz, & Hüllermeier, 2014); a phenomenon related to the challenge faced in hidden Markov models known as the label-bias problem (Dietterich, 2002) where it denotes the effect of high uncertainty at some part of the chain leading to an error, which in turn propagates down the chain causing further errors. There are numerous factors implicated in this uncertainty. One obvious factor is the length of the chain, and here there are several aspects to notice: a longer chain is inherently more susceptible to propagating error (simply because of greater length for an error to propagate over); yet conversely – if this chain is heavily cascaded/connected – it then also offers a potentially more powerful expansion of feature space (by offering to leverage an increasing number of input nodes; recall, Section 3.2). Naturally, a longer chain is much more computationally challenging to manage, with the number of possible predictions increasing exponentially and the number of possible chain structures increasing hyper-exponentially (w.r.t. length/the number of labels L).

A sparse DAG is attractive because it simplifies the model, making it more efficient and scalable, and it can even improve predictive performance (because it may reduce error propagation by reducing propagation in general). Enforcing sparsity may also yield a more interpretable model. Unfortunately, although such a hypothetical DAG is efficient to use, finding it (as well as chain order) is a question of combinatorial complexity, and the possible structures cannot be exhaustively trialed for more than a dozen or so labels: just 6 labels of the benchmark emotions data (considered in Figure 8 w.r.t. Jaccard score[2]) imply a total of 720 different chain orders, and 32 768 unique DAGs.

Due to the different factors involved (accuracy, efficiency, interpretation, . . .) the motivation for structure search is often not clearly linked to any particular factor. In the fol-

[2] Also known as Jaccard index, or Jaccard similarity coefficient, considered alongside Hamming and 0/1-metrics by, for example, Park and Read (2018); higher is better

lowing, we focus on the predictive performance (which arguably has motivated the largest number of related studies in classifier chain order) and additionally treat efficiency and interpretation with regard to this as a question of tradeoff.

In the following we discuss different options for structure search. In order to facilitate this discussion and the different issues that arise, we consider the example of Figure 9, featuring a toy example of three labels, each representing a logical operation on the two binary inputs (recall also the related Figure 7).

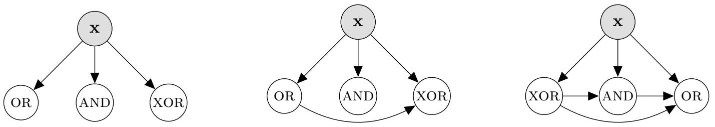


Figure 9: A toy problem, with two bits as input, $\boldsymbol{x} \in \{0,1\}^2$, and three labels that represent the logical operations on these bits. Shown are 3 of the 8 possible directed structures among the three nodes (note that the order changes). The suitability of each depends on the base classifier, where a non-linear classifier like a decision tree will solve all labels on any structure, and logistic regression is only suitable on the middle with greedy inference; also on the fully-connected structure (right) with exhaustive inference.

4.1 Random Ensembles

Using ensembles of randomly ordered chains was an early approach (Read et al., 2011) that proved effective in a similar sense to other methods that induce diversity among ensemble members and then combine predictions, such as bagging (Breiman, 1996), and a similar bias-tradeoff analysis applies. Bagging is typically carried out using an unstable learning algorithm such as a decision tree inducer. However, even if base models are not necessarily unstable, imposing a random order on each chain model helps to achieve this effect. This is related to, for example, imposing random structures of neural networks to make them more diverse when used in an ensemble. The averaging effect of the ensemble thus reduces the variance component of the error caused by this randomness. Note however that this strategy essentially mitigates potentially poor chain orders rather than identifying a good order (as seen in Figure 8 where the ensemble of chains is an average performer compared to the best and worst chain orders).

4.2 Using a Label Hierarchy

Hierarchical classification has long been of interest in the multi-label community (Kiritchenko, Matwin, Nock, & Famili, 2006; Ramrez-Corona, Sucar, & Morales, 2014). A hierarchy designed by a domain expert will almost certainly incorporate some form of label dependence, and is available for use by an algorithm even prior to seeing the data. However, such hand-built hierarchies and other structures (if they are available) are usually designed for human interpretation of the data rather than to increase classifier performance. Indeed, a model class has typically not even been considered at the time of the hierarchy design. Top performance on hierarchical classification problems can be obtained with chain-based methods that completely ignore the hierarchy (Puurula et al., 2014), indicating that such a consideration is not always necessary.

We do emphasise that results may differ if the hierarchy explicitly forms part of the loss metric, but this goes beyond the scope of this work. In other cases, making use of a hand-built/domain-expert hierarchy can be considered a special case of generating structure from label dependence, which we discuss in the following, Section 4.3 (from the context of learning the dependence from the data). Later, in Section 5.5, we discuss hierarchical tree-based methods in the context of related work.

4.3 Structure from Label Dependence

A large part of the classifier-chains literature promotes structuring the chain according to label dependencies discovered in the data. This idea is well founded; it would be inefficient to place a chain structure over a set of labels that are independent of each other. And it is also attractive in the sense that measuring dependence between variables is a highly studied problem for which many statistical tools exist. A common recipe for classifier chain-based methods (and other methods in the wider multi-label literature) has been to 1) measure label dependence, and 2) use the dependence measurements to create a structure (often a sparse one for efficiency reasons), and 3) deploy base classifiers and an inference method of choice. This approach is taken by Zhang and Zhang (2010), Read et al. (2015), Kajdanowicz and Kazienko (2013), Jun et al. (2019) and dozens more (many others cited by those authors).

However, although it may be intuitively obvious that dependence is connected to structure, exploiting this intuition successfully requires further thought.

A simple approach is to measure global label dependence by counting label co-occurrences, and to derive a structure from this, where high dependence favors connectivity. An initial inconvenience is the difficulty to obtain any representative sparse structure, due to the almost ubiquitous inter-dependence of multi-label data. A second issue is that any global dependence detected among labels in the dataset does not imply a corresponding advantage connecting these in a chain structure at testing time where input instances are observed (the reader may recall the discussion on global versus conditional dependence in Section 2, and in particular Table 1).

Measuring conditional label dependence (dependence under observation of input $\boldsymbol{x}$) implies building models. Probabilistic classifiers h_j (recall; Eq. (9)) incorporate a mechanism for obtaining approximation $P(\cdot|\boldsymbol{x},\cdots)$ that can be used to check for conditional dependence against Eq. (7). This immediately raises two questions: firstly, of the computational complexity of building these models (as required for each measurement), and secondly, of the effectiveness of the chosen model class for the particular problem; different base classifiers will provide different approximations of ground-truth P.

An efficient way of estimating conditional label dependence was described by Zhang and Zhang (2010). It requires building only one classifier per label (and is thus linear with L) and thereafter only incurs the computational overhead in computing pairwise statistical measurements of dependence between their *errors*. This is because the individual errors $\epsilon_j = \ell(y_j, \widehat{y}_j)$ (of each j-th label, under loss function ℓ) are conditioned on the instance via the prediction: $\widehat{y}_j = h_j(\boldsymbol{x})$. This approach is therefore relatively affordable.

The model class of base classifiers has a huge effect on the ideal chain structure. Consider, for example, the toy problem in Figure 9 having logical operations as labels (relating also to Figure 7, as discussed in Section 3.2): if classifiers h_j learn to predict their labels perfectly given any pair of input bits $\boldsymbol{x}$ then there is no point in connecting any labels in a chain (conditional independence holds). But of course, a perfect model cannot usually be obtained in practice. For example, logistic regression fails to model the Y_{XOR} concept correctly on its own, which may show up in measurements of dependence, and thus according to the argument of structure from dependence, chain connectivity should be increased around this label.

However, we remark that the *order* of labels in the chain (its *directionality*) cannot be determined from measurements of statistical dependence. In Figure 9 it is apparent that the order in which OR and XOR are given may be crucial to the success of the problem (particularly when greedy inference is employed). Figure 8 also illustrates numerous cases where a minor change in label order has a significant effect on accuracy, for example, notice the considerable drop in accuracy from swapping labels 4 and 5 at the end of the chain.

To take this aspect into account, Zaragoza et al. (2011) create a connected structure, obtained from measurements of label dependence, and then convert it into a number of trees with different directionality across each one (namely, one tree per label, each label is used once as a root node). As an ensemble method, it incurs the relative advantages and disadvantages of such an approach (see Section 4.1). Particularly, having so many models can be unsatisfactory in terms of computational complexity and interpretability (notably, it does not shed any light onto causality).

Overall, label dependence measurements can potentially provide interesting insights into interpreting the data (just as feature dependence is often measured in single-label data to obtain this information) from a correlation point of view. Conditional dependence can hint towards structures that might be useful for prediction, but this is ultimately dependent on the selection of base classifiers and inference schemes – both of which are often plugged in after a structure is set. As such, dependence cannot reveal causality, and it cannot not unveil the most powerful directionality (order of labels) across the chain.

4.4 Order via Base Classifier Accuracy

Heuristics can also be designed around the expected predictive performance of base classifiers, i.e., their *accuracy*: labels that are easier to model (i.e., have relatively higher predictive performance) can be placed near the beginning of the chain, supposing that these models are less likely to generate errors (which would be propagated down a relatively longer part of the chain). Therefore, under this view, weaker classifiers are placed nearer the end of the chain where they have less influence. Studies have been carried out by Enrique Sucar et al. (2014), Kajdanowicz and Kazienko (2013); and discussed also by Senge et al. (2013). One should note that although this approach suggests an order across labels (unlike label-dependence), it does not lead to a general solution for structure. Neither is there any consensus in the literature over whether this approach is effective. Indeed, very poor predictions for a particular label may potentially serve as excellent features for other predictions. In view of this, such labels should be placed nearer the beginning of the chain to serve as good feature expansions (as implied in Section 3.2) even if their own predictions are poor.

4.5 General Search for Structure

Although there are good arguments given in the literature to the above-discussed factors behind chain-order performance, in real-world data, the different factors are convoluted and may interact and counteract each other. This is apparent when we take a closer look at per-label accuracy; as in Figure 2. Therefore a single heuristic may not be globally effective.

m	Chain Order	Jaccard	A1	A2	A3	A4	A5	A6
5	[1 2 3 6 4 5]	0.52	0.81	0.70	0.73	0.93	0.82	0.74
6	[1 2 3 6 5 4]	0.54	0.81	0.70	0.73	0.93	0.86	0.74
17	[1 2 5 4 6 3]	0.53	0.81	0.70	0.73	0.94	0.81	0.73
18	[1 2 5 6 3 4]	0.53	0.81	0.70	0.71	0.91	0.81	0.77
20	[1 2 6 3 5 4]	0.57	0.81	0.70	0.75	0.95	0.84	0.77
42	[1 3 5 6 4 2]	0.57	0.81	0.72	0.74	0.94	0.83	0.76

Table 2: A selection of models $\{m\}$ (indexed from left to right in Figure 8), associated Jaccard score (also as in Figure 8) alongside respective per-label accuracy (Aj) for all labels $j = 1, \ldots, 6$. Moving a label forward or backwards in the chain may have either a positive or negative outcome for the labels involved or make no difference at all; and per-label changes may counteract each other to be invisible under the overall score associated with a structure.

If our aim is to find an optimal chain structure under a particular loss function, given a particular base classifier and inference configuration, at any cost, then we may approach the task directly as a trial-and-error search through the space of all possible structures. This task is already relevant in Bayesian network structure learning (Gasse, 2017). Finding the optimal structure this way is an NP-hard problem, but many options already exist for tackling it, for example: local search, simulated annealing, and other hill-climbing and evolutionary methods; and many of these have been adapted specifically to classifier chains, for example, by Kumar et al. (2013), Read et al. (2014), Goncalves et al. (2013). And Gasse (2017) gives a thorough treatment in the general application to multi-label learning. Figure 10 illustrates a simple example using tree search.

Formally, letting θ represent the parametrization of the structure of a chain classifier (for which base model class is pre-selected), and $\mathcal{J}(\theta)$ be the payoff (accuracy) incurred by this model, then we are searching for

$$\theta^* = \operatorname*{argmax}_{\theta \in \Theta} \mathbb{E}[\mathcal{J}(\theta)] \tag{11}$$

where space Θ contains all possible structures, of hyper-exponential size w.r.t. L, namely $2^{L(L-1)/2}$ DAGs (or $L!$ orders given fixed structure). Note that, of course this is an argmin when considering loss metrics ℓ.

Aside from the size of the space, we note that each proposed step/trial $\theta' \in \Theta$ in the search requires the approximation of an expectation, because we want parametrization θ to

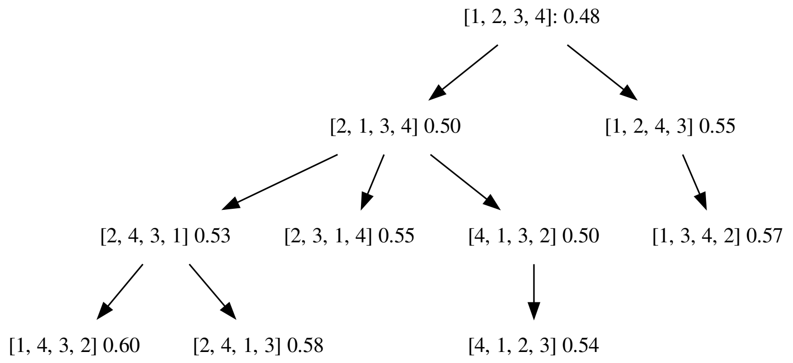


Figure 10: An example of searching for chain order via tree search, over $L = 4$ labels. At each node the chain order θ is shown with its associated expected payoff $\mathbb{E}[\mathcal{J}(\theta)]$ (as in Eq. (11)). Each branch is associated with a measurement of this estimated payoff; implying a slow expansion. Nevertheless, as is typical, a small search can render an important improvement in predictive performance, even if no clear pattern emerges with regard to chain order.

perform well in general on unseen test data. An average over internal k-fold cross validation results using the training data can be used in its place – a task which requires building k copies of a classifier chain (which itself consists of L base classifiers) at each step; and implies a difficult tradeoff: More cross validation (i.e., more folds) may better approximate the expectation (and facilitate a more precise search), but using k folds implies a factor of k in running time. On the other hand, using fewer folds implies less computational expenditure, but higher variance in their estimated performance. This challenging scenario is known in other tasks, and many tools are available; Powell (2019), for example, provides a synthesis of techniques and applications.

Even with state-of-the-art tools and computational resources, the optimal θ^* (satisfying Eq. (11)) may never be found and we will have to make do with some approximation $\hat{\theta}$, such that (we hope) $\mathcal{J}(\hat{\theta}) \approx \mathcal{J}(\theta^*)$. Luckily, in practice, the surface of the function $\mathcal{J}(\theta)$ over θ is often undulating with many local maxima that yield good results (Read et al., 2014) and therefore off-the-shelf searches yield a good local maximum $\hat{\theta}$ (corresponding to an effective chain) relatively quickly. Figure 8 also shows an indication of this. Performing multiple searches in parallel (from different points in the space) additionally yields an effective ensemble (where all members rest on or close to some local maxima).

In addition to off-the-shelf textbook methods, one can make particular adaptations of a search that are useful in the context of classifier chains. Namely, one can take advantage of the fact that under greedy inference a search in chain space can be made increasingly faster by freezing the chain along its directionality. In other words, a simulated-annealing type scheme where proposed structures differ from the ones trialled already only near the end of the chain, and increasingly so over time. This speed-up is on account of only having to retrain classifiers which are ‘down chain’ from the first modified node and the first part

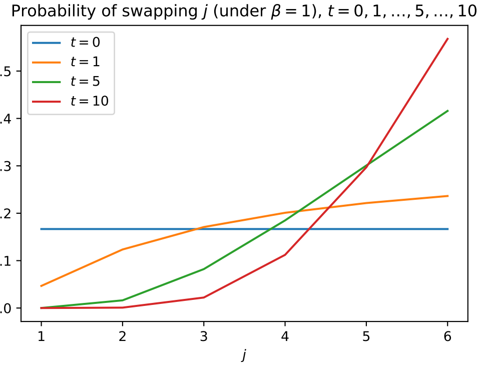


Figure 11: Searching for a chain structure or order involves trialing a series of proposals. For example, two labels can be swapped in their positions to propose a new order. This figure shows a distribution for selecting such labels, which is modified over time steps t of the search, at speed proportional to β. The search becomes cheaper over time, since altering the chain from point j only involves retraining base models $j + 1, \ldots, L$. Based on Read et al. (2014).

of the chain can be reused from earlier trials (Read et al., 2014). An illustration is given in Figure 11.

By imposing a fixed structure, the search space is reduced to simply the order in which labels appear within that structure. Read et al. (2015), for example, use a relatively-sparse trellis structure. Teisseyre (2017) employs L_1 regularization to prune a fully-connected directed structure to a sparse one. The latter approach is particularly efficient as the structure 'search' is carried out as an integral part of the learning algorithm; albeit with a loss of flexibility since this method does not work generally for all types of regularization and base classifiers.

The search for chain structure is often approached exclusively in the model-building (training) phase, prior to inference on test instances. After all, the structure is an integral part of a classifier chain model and one can obtain a single structure which works globally well on the test set. On the other hand, it must be emphasised that even an exhaustive search will not necessarily uncover a single ground-truth representation θ^* (in Eq. (11)). In any case θ^* is not necessarily unique, and moreover, may differ among evaluation metrics and types of instances. Since metrics $\mathcal{J}$, and particularly, test instances, are not always observed at training time, it can also be appropriate to move the structure search to inference time, as follows, w.r.t. the MAP estimate, extended from Eq. (10):

$$\widehat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \{0,1\}^L, \theta^* \in \Theta} P(\boldsymbol{y} | \boldsymbol{x}; \theta^*) \tag{12}$$

where $P(\cdot|\cdot;\theta)$ implies traversing the probabilistic representation specified by the structure θ (a particular classifier chain arrangement).

Naturally, conducting the full search indicated by Eq. (12) is impractical even for modestly sized label sets since it requires training multiple models per test instance. However, if we have already carried out a search at training time for a globally-performing structure, then we have already trialed a set of strutures $\mathcal{S} = \{\theta_1, \ldots, \theta_M\}$. Replacing Θ with $\mathcal{S}$ in Eq. (12) yields a dynamic search that incurs only an additional factor M of running time per test instance, the same as an ensemble of M models, yet with often better results (Read et al., 2014; da Silva, Gon calves, Plastino, & Freitas, 2014; Narassiguin, Elghazel, & Aussem, 2017; Pinto, Soares, & Mendes-Moreira, 2016).

If each individual $\theta \in \mathcal{S}$ is obtained by starting from a different point in the search space, there will typically be considerable variety among these locally-optimum structures, despite the fact that all are rendered in the search for a globally optimum structure. On small problems, where it is feasible to generate a large part of the structure space, it is possible to observe many disconnections between local changes in structure and global changes in accuracy. For example in Figure 8 (and, respectively, Table 2), the small difference in chain order from swapping labels [1 2 3 6 4 5] and [1 2 3 6 5 4] corresponds to a notable jump in predictive performance (particularly surprising as this swap comes at the end of the chain). Conversely, the orders [1 2 6 3 5 4] and [1 3 6 5 4 2] appear objectively dissimilar from each other, but obtain very similar (and relatively high) Jaccard score. Table 2 provides particular insight.

4.6 Summary and Alternatives

The question of the 'best' chain structure can only be answered respective of evaluation metric, base classifier(s), parametrization, inference method, and test instances – and none of these items are necessarily attached to the available real-world training data. The structure of models is an important issue across many areas of machine learning, including probabilistic models and neural networks. Therefore research is ongoing in many areas (a recent survey is provided by Scanagatta, Salmerón, & Stella, 2019).

Even though investing in structure search can pay off in terms of global and local predictive performance (regarding test set, and individual test instances, respectively), the challenges and instability associated with this search has inspired researchers to avoid this issue altogether. Indeed, increasingly-promising efforts have given rise to effective methods which are similar in approach to classifier chains, but avoid the question of chain structure. We look at some of these in the following section, and discuss the relative disadvantages that this approach incurs.

5. Related Methods

Having elaborated prediction with classifier chains as probabilistic inference (Section 3.1) and a feed-forward pass of a neural network (Section 3.2), it is inevitable to turn up close connections to other methods in these areas, and other related areas.

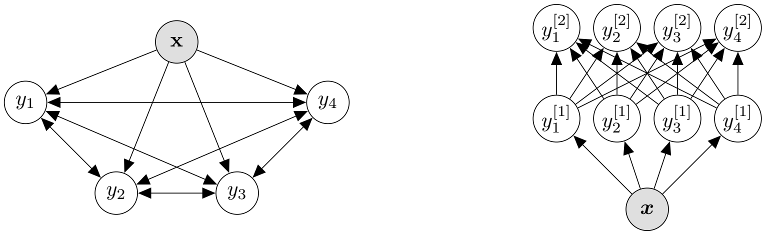
5.1 Probabilistic Graphical Models

The probabilistic view of classifier chains revealed that a classifier chain is in fact a type of probabilistic graphical model. It can be seen as a particular case of a conditional random field (Dembczyński et al., 2012b), or a maximum entropy Markov model or a hidden Markov Model (Read et al., 2017), or another variety of graphical model, depending on the chain structure and inference method chosen. Note that Markov models specifically imply the Markov assumption which is not a typical restriction in chains. Also, greedy inference is considered the standard option for classifier chains, i.e., a single directed pass, but typically more rigorous (even if still approximate) inference is carried out in, e.g., conditional random fields (CRFs), and to specifically minimize log loss.

A main point of departure from classifier chains, at least as we have defined them in this work, is when directionality is removed from (or equivalently bi-directionality is implied upon) the edges connecting label nodes. Such a graph of binary classifiers (see Figure 12a), was proposed by Guo and Gu (2011). Compared to a directional chain, the training procedure is simplified: each binary classifier takes the output of all other classifiers as additional input; so the question of label order is no longer in consideration, neither the computational expenditure required for it. However, the prediction/inference phase is significantly more intense: single-pass greedy inference is not possible or at least not effective, and rather, hundreds or thousands (or more) iterations of Gibbs sampling may be required for each test instance. For larger labelsets, the question of structure (even if not directionality) is still relevant, since sparsity becomes necessary for tractability. Essentially, for each test instance $\boldsymbol{x}$, this approach relies on taking samples

$$\tilde{y}_j^{[t]} \sim p(y_j|\boldsymbol{x}, \tilde{y}_1^{[t]}, \ldots, \tilde{y}_{j-1}^{[t]}, \tilde{y}_{j+1}^{[t-1]}, \ldots, \tilde{y}_L^{[t-1]}) \tag{13}$$

over iterations $t = 1, 2, \ldots$ until convergence. Then, a marginal mean or joint mode estimation (to minimize Hamming loss or 0/1 loss, respectively) can be obtained simply by averaging or taking the mode over samples.



(a) CRF/Undirected network.

(b) Stacked binary relevance.

Figure 12: Multi-label models related to classifier chains, exemplified for $L = 4$ labels: (a) an undirected model (shown as bi-directional) among labels, from Guo and Gu (2011), implying iterations at inference time; and (b) a stacking approach such as used by Cheng and Hüllermeier (2009) among others (note that often the second level of labels in (b) also takes the input $\boldsymbol{x}$, but this is omitted for brevity).

5.2 Neural Networks

The view of classifier chains as neural networks (as we discussed in Section 3.2) raises obvious connections, not just to standard multi-layer perceptron architectures (which, as Nam et al., 2014 demonstrate, can be competitive multi-label models) but also relatively recent architectures, for example, residual neural networks (He, Zhang, Ren, & Sun, 2016); ResNets. In Figure 13, we illustrate a simple ResNet for single-label prediction, in a way that draws obvious comparison to the cascade of a classifier chain as shown in Figure 6; only the type of inner layer variables differ, with ResNets having been introduced to cascade inputs across hidden layers rather than label-space.

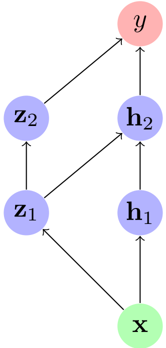


Figure 13: A ResNet for single-label prediction, using node notation similar to Figure 6 w.r.t. $\mathbf{z}_l$. In this case, $\mathbf{h}_l$ are 'standard' hidden layers in the network. Note the structural similarity to that of the classifier chain in Figure 6.

Further connections can be seen to variants of recurrent neural networks (RNNs) such as long short-term memory networks (LSTMs). Indeed, LSTMs have already found some success in application to the multi-label problem (Nam, Loza Mencía, Kim, & Fürnkranz, 2017). The main interest in this application is in the context of an extremely large number of labels, since potentially substantially fewer parameters are required; in fact possibly fewer than the number of output labels, since with a fixed number of parameters (as chosen by the user as a hyperparameter) an RNN will continue producing labels until an end-of-sequence symbol is output. We remark that unlike usual applications of sequential models like RNNs, the order that labels are output is not important for multi-label classification tasks, rather they are considered as an unordered set.

5.3 Stacking

As noticed by Kajdanowicz and Kazienko (2013), Senge et al. (2013) and others, chaining can be viewed as a particular case of binary relevance stacking and vice versa. Stacking approaches for multi-label classification have been presented and studied independently, for example, by Cheng and Hüllermeier (2009), Loza Mencía and Janssen (2016), Kajdanowicz and Kazienko (2009). These methods typically use the predictions of independent binary classifiers as inputs to a second set of classifiers. The vanilla version of this approach is shown in Figure 12b, noting that often the input is additionally directed to the second layer of classifiers. The understanding is that the second layer 'corrects' the predictions of the first layer in taking into account dependence among labels.

A stacked prediction is made as follows for the j-th label:

$$\widehat{y}_j = \widehat{y}_j^{[2]} = h_j^{[2]}(\boldsymbol{x}, \widehat{y}_1^{[1]}, \ldots, \widehat{y}_L^{[1]}) \tag{14}$$

where $[l]$ denotes the l-th layer, i.e., $h_j^{[2]}$ is the base classifier on layer 2 responsible for predicting the relevance of the j-th label. The similarities with greedy inference (Eq. (8))

Table 3: A learning problem with regard to label y_2, shown for (a) training, and (b) testing, under related schemes, including a classifier chain. Here, ϕ is a basis function (e.g., $\phi_1 = x^2$) and $\hat{z}_1$ is a latent variable (not known at training time). All values with hats, e.g., $\hat{z}$, require a learning procedure to obtain; otherwise are available as (or can be calculated directly from) the dataset prior to learning. We have omitted superscripts of training examples ($x \equiv x^{(i)}, y_2 \equiv y_2^{(i)}$) for brevity; $\tilde{x}$ is explicitly a test example (with no known associated labels).

	X_1	X_2	X_3	Y_2
Basis expansion	x	ϕ_1	ϕ_2	y_2
Classifier chain	x	y_1		y_2
Stacking	x	$\widehat{y}_1$	$\widehat{y}_2$	y_2
Hidden layer	x			y_2

(a) Training

	X_1	X_2	X_3	Y_2
Basis expansion	$\tilde{x}$	ϕ_1	ϕ_2	$\widehat{y}_2$
Classifier chain	$\tilde{x}$	$\widehat{y}_1$		$\widehat{y}_2$
Stacking	$\tilde{x}$	$\widehat{y}_1^{[1]}$	$\widehat{y}_2^{[1]}$	$\widehat{y}_2^{[2]}$
Hidden layer	$\tilde{x}$	$\hat{z}_1$	$\hat{z}_2$	$\widehat{y}_2$

(b) Inference

in a standard classifier chain is apparent: with a difference that in a classifier chain the j-th label is only predicted once. It is easy to imagine simple changes to cross from one form to another. For example, by passing over a chain twice, we obtain

$$\widehat{y}_j = \widehat{y}_j^{[2]} = h_j^{[2]}(\boldsymbol{x}, \widehat{y}_1^{[1]}, \ldots, \widehat{y}_L^{[1]}, \widehat{y}_1^{[2]}, \ldots, \widehat{y}_{j-1}^{[2]}) \tag{15}$$

for the prediction of the j-th label in the second pass/iteration; a general combination of chaining and stacking of which both a standard chain and stacked set of classifiers are special cases. As we add a third and fourth pass over the chain, and so on, we find close ties to inference for undirected chains (recall Figure 12a and, particularly, Eq. (13)) – except in that case nodes are chosen (i.e., samples are taken) stochastically, rather than taking the mode as prediction.

However, if we look at the training phase, there is an important distinction to make with regard to stacking. If each model $h_j^{[l]}$ is trained using the true labels $\{y_j^{(i)}\}$ as they are presented in the training dataset (as additional inputs alongside each $\boldsymbol{x}^{(i)}$), this can indeed be seen as a particular case of classifier chains. On the other hand, if $h_j^{[l]}$ is trained using some predictions $\{\widehat{y}_j^{(i)}\}$ as inputs, where $\widehat{y}_j^{(i)} = h_j^{[l-1]}(\boldsymbol{x}^{(i)}, \ldots)$ from a previous layer, then there is a qualitative difference; this form of training is typical of stacking. When classifier chains uses this procedure of training models using predictions rather than the training set labels, this is called *nested stacking* by Senge et al. (2013); and those authors showed interesting results: nested stacking performs better under Hamming loss (presumably by leveraging predictions as feature-space expansion), whereas standard classifier chains perform better under 0/1 loss (as they are able to model conditional label dependence). Nested stacking has also been recognized elsewhere (for example, by Li & Lin, 2014) under different names. Table 3 provides an explicit example and comparison among four methods including these, which can provide further intuition.

5.4 RAkEL

The large family of methods stemming from the so-called *label powerset* approach provides a well-known alternative to classifier chains. The RAkEL (*ra*ndom *k*-lab*el* subsets) method (Tsoumakas, Katakis, & Vlahavas, 2011) has been heavily cited and extended in the multi-label literature. Rather than binary base classifiers, label powerset methods consider multi-class base classifiers, where each label vector/combination is considered as a class value; thus the total number of possible classes is 2^L. The method gets its name from set notation, where all combinations of labels from a set of labels, e.g., $\{1, \ldots, L\}$, is indeed the powerset $\mathcal{P}$, of cardinality 2^L (we can denote this by $\mathcal{P} = \{0, 1\}^L$). However, in practice, the number of classes is much less than 2^L because it is limited to the labelsets actually observed in the training data (of size N), and because only sets of $k < L$ labels are considered (as in RAkEL), and possibly also because the set is explicitly reduced to only the most frequently observed labelsets as performed by Read et al. (2014). Although these methods were originally seen as a different approach to classifier chains, in a probabilistic setting, they can all be observed as approximating the same optimization problem, namely minimizing 0/1 loss as $k \to L$. Suppose indeed that $k = L$ and that a set of labelsets $\mathcal{V} \subset \mathcal{P}$ has been selected by one of these methods, where $|\mathcal{V}| \ll 2^L$, then the inference task is to select which labelset maximizes the posterior joint conditional:

$$\widehat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{V}} P(\boldsymbol{y}|\boldsymbol{x}) \tag{16}$$

i.e., a MAP estimate. In comparison to Eq. (10), one sees that, whereas classifier chains provide an efficient (probabilistic tree-)search over the space of all 2^L possible predictions, label-powerset methods restrict the search space itself.

5.5 Tree-based Methods

In Section 4.2 above we discussed hierarchies, where a tree is modeled around a pre-defined dataset hierarchy. It is also clear that a tree, rather than a cascaded chain, is one kind of generalization that can be learned for classifier chains, i.e., yielding classifier trees. This approach has been studied by Dembczyński et al. (2016) in a probabilistic context.

In this scenario, of trees built around class labels, we find close connections to other tree-based approaches, such as nested dichotomies (Frank & Kramer, 2004; Leathart et al., 2019); even if these are not necessarily for multi-label classification – having binary class probability estimates facilitates efficient tree search techniques for efficient inference in many contexts. The same can be said with regard to hierarchical softmax (Wydmuch, Jasinska, Kuznetsov, Busa-Fekete, & Dembczynski, 2018).

Condensed filter trees (Li & Lin, 2014) are a tree-based method for multi-label learning related to classifier chains, particularly in regard to having 2^L possible paths from root to leaf (see, particularly, Figure 5). The novelty in these trees is based on the fact that, since many paths are seldom taken, focus can be placed on important ones, hence ‘condensing’ the structure.

5.6 Structured-Output Prediction

Multi-label learning is a particular type of structured-output learning, for which typical examples involve sequence to sequence mapping or image segmentation (i.e., outputs with inherent local dependence) but structured output learning also considers problems that require output of more complex structures such as trees and graphs. CRFs and Markov models already mentioned above are well-known models to this kind of learning.

The interesting connection to classifier chains is in methods where the aspect of search is combined with learning, as well as the idea of a transformation (or reduction) to simpler problems. A notable example is the SEARN algorithm (Daumé, Langford, & Marcu, 2009) which reduces structured output to binary classification via cost sensitive classification. These authors use the language of reinforcement learning, and talk of a *policy* h that is built for each training example, and taking *action* $\widehat{y}_j$ (making a prediction). However one can get close to this view from Section 4.5, where the search and the learning becomes inextricably linked. Furthermore, as with classifier chains, a main advantage (compared to many other structured-output methods) is being able to tackle different loss functions using any model class (i.e., what we refer to as a base classifier). In Section 3.1, we mentioned how probabilistic chains can be flexible with regard to loss function, and this was further generalized by the hyper-parametrization of the loss function with regard to chain order search (Eq. (11)). In this sense, in their most generic formulations, SEARN and classifier chains are closely related, and could be formulated as special cases of each other, not forgetting that the original respective targeted areas (structured-output learning vs. multi-label classification for chains) do indeed lead to different formulations.

Whereas SEARN (as most varieties of classifier chain) uses classifiers to produce structured outputs through a single sequence of greedy decisions, HC Search (Doppa, Fern, & Tadepalli, 2014a) makes use of recurrent classifiers. This approach has been shown effective vs. SEARN, as well as on multi-label problems (Doppa, Yu, Ma, Fern, & Tadepalli, 2014b). In this context, a heuristic (H) provides a search (e.g., a tree search, as mentioned above) across possible labelsets $\boldsymbol{y}$, and then it is a cost function (C) which evaluates these candidates. A novelty separating this method from chaining as we have discussed it is the idea that the search may be learned separately under a different loss, as opposed to employing an off-the-shelf textbook search method. The idea of evaluating (albeit not necessarily explicitly ranking) candidates $\boldsymbol{y}$ produced by a structure is identical in the probabilistic chain formulation.

5.7 Summary

A full elaboration of all related methods is beyond the scope of this paper. Instead, we can emphasise again the particular niche of classifier chains: a flexible hyper-parametrization of binary base classifiers trained on a transformed multi-label dataset with off-the-shelf tools and models; a fast approximate inference over general DAG structures, leading to their strong out-of-the-box performance without the need for hand-crafted feature functions or hidden units.

6. Perspectives and Open Issues for Classifier Chain Methods

Given that over a decade has passed since the initial formulation of classifier chains as a method of multi-label classification, it is well worth asking whether they are still relevant and competitive, in the rapidly evolving area of multi-label learning.

In general, the constant flow of modifications and developments building on classifier chains that appear in the scientific literature suggests that interest is strong and ongoing. Of course, developments are driven by the general interest in multi-label learning. Several new implementations of classifier chains have appeared in major open-source software frameworks in recent years, including Scikit-Learn (Pedregosa et al., 2011) and derivatives (Szymański & Kajdanowicz, 2017; Montiel et al., 2018).

On the other hand, there are several limitations that are becoming increasingly apparent, for example computational complexity. A single fully-cascaded chain implies quadratic complexity in terms of feature space expansion. This is negligible on datasets with only tens of labels, but in recent years the multi-label community has approached ever larger datasets, eventually including a class of "extreme multi-label" problems, for example by Liu et al. (2017), with tens or hundreds of thousands of label concepts.

Many strategies can be taken to extend usability and scalability of chains, for example ensemble subspaces have been used successfully in datasets with many thousands of labels (Puurula et al., 2014), where chains are built on a subset of the labels and their votes are combined. This label-subspace methodology draws heavily from other approaches, such as the RAkEL method (Tsoumakas et al., 2011).

Particularly, as data sets grow larger and as computational power becomes cheaper and more widely available (especially GPUs and TPUs, etc.) it becomes increasingly difficult for classifier chain approaches to out-compete neural network architectures (such as those by Liu et al., 2017; but there are many such examples), for which maturing frameworks exist. Earlier, the chaining mechanism replaced to some extent the need for hidden nodes and learning their associated weights/parameters (see Section 3.2) providing an off-the-block advantage against data-hungry neural networks. But now data is increasingly available, and it is possible to build networks of millions of parameters, and train those parameters, with only a few lines of code and a few hours of GPU time.

On the other hand, even though the largest multi-label datasets are becoming larger, creating a new trend in extreme classification, there is no shortage of new real-world applications associated with only modest numbers of labels in smaller tabular datasets. And this is likely to maintain interest in and development of chain methods. Besides that, we emphasise that neural and chain architectures are not by any means mutually exclusive (as already seen in Section 3.2) and neural architectures can further benefit from aspects found in classifier chains, as explored by Read and Hollmén (2017) and Cisse et al. (2016) among others.

Still, in cases of 'small data', where deep networks of latent nodes are not needed or suitable, there are often particular challenges for classifier chains that need further attention. For example, as a set of binary classifiers, chain methods are particularly vulnerable to problems of *class imbalance*, stemming from the sparsity of the label matrix of most multi-label datasets. There have been proposals to address this, for example, by Liu and Tsoumakas (2018), Lin and Xu (2016). In some cases the sparsity may be linked additionally to the phenomenon of *weak* labels – a type of noise due to 'lazy' human labeling where some relevant label values are missing in the training data.

Although interpretability has not been a motivating factor behind most work on classifier chains to date (or on multi-label learning in general) it can be seen as a key advantage offered by chaining. Unlike multi-layer neural networks or other multi-label methods where inter-label dependence is 'hidden' in (i.e., modeled by) inner layers of the network, classifier chains explicitly models this dependence in a structure in the output space – among the labels. In the multi-label literature, one can find different visualizations of label dependence, usually in the form of graphs or heatmaps, but the associated work for the most part does not verify such dependence relations with a domain expert, or show them to be useful or offering insight to any real-world problems, or even demonstrate stability of the relations from one test set to another. In addition, further development is needed to establish if and in what contexts dependence is useful and reliable for interpretation – particularly taking into account base models and chain structure. It seems that this is a clear path requiring attention, especially with growing interest in interpretable machine learning (Molnar, 2019).

There have been increasingly advanced efforts to integrate feature selection into the chain, for example, by (Tenenboim-Chekina, Rokach, & Shapira, 2013; Teisseyre, 2017). Such an approach makes sense, as any conceptual boundary between feature and label variables is already inherently blurred by the chaining mechanism.

Using so-called automatic machine learning (Auto-ML) could prove useful as a general means for electing and calibrating base models, chain order, and so on. There is some recent and early work in this direction involving classifier chains (Wever, Tornede, Mohr, & Hüllermeier, 2020).

The choice of SEARN (mentioned in Section 5.6) to use the terminology of a *policy* is particularly interesting. If transferred to the area of classifier chains, one can easily imagine using many of the large variety of methods from the area of reinforcement learning, such as Monte Carlo tree search, for finding a classification in a non-greedy way.

Another interesting perspective of classifier chains is via transfer learning and concept drift adaptation. In a sense, building classifier chains *is* transfer learning. In machine learning, if we become interested in a new class concept, we may want to adapt from (i.e., transfer) knowledge representations of an existing similar concept. The labels in a multi-label dataset are almost always related in some way (after all, they are part of the same dataset). Therefore adding a new label to a classifier chain could be considered as transferring knowledge from existing concepts (i.e., labels) to learn the new one. Of course, in typical applications of transfer learning, such as adaptation to concept drift, older concepts are no longer useful and can eventually be discarded, unlike in the typical multi-label case, where all labels are, in most cases, considered equally relevant.

One may then even-more easily consider the adaptation of chaining mechanisms to multi-task learning – learning L tasks together. Multi-label learning is clearly a special case of multi-task learning. In the general case, however, there is more freedom with regard to different types and sizes of input spaces per concept in multi-task learning. Furthermore, multi-task learning inherently includes (or at least does not exclude) regression tasks among the L tasks to be learned – which will take us to the final point of this section.

The development of chains in a regression context, where labels take on continuous values $y_j \in \mathbb{R}$, appears natural at first approach, yet it meets significant challenges. Despite early demonstrations of direct application (Borchani, Varando, Bielza, & Larrañaga, 2015), getting strong off-the-shelf performance (vs. independent models) from these so-called *regressor chains* has proven more difficult. This is mainly due to different loss functions targeted in regression, and the lack of inherent non-linearity (for example, consider: logistic regression is linear in terms of decision boundary, but classification via logistic regression is not, because each input is cast to either 0 or 1). A recent analysis is provided by Read and Martino (2020). Overall, regressor chains appear to be an interesting avenue for future research, but they behave and require a treatment so different from their classifier homologues that we can avoid analysis of them in this paper.

7. Summary and Recommendations

In this work, we have catalogued the evolution of the family of methods of classifier chains across many different analyses, and synthesized many of these methods and their respective advantages and disadvantages.

We have not provided a large-scale empirical comparison of different methods, since the inherent flexibility of classifier chains makes it difficult to set up a fair but concise evaluation. Almost all varieties target some point on the spectrum of possible tradeoffs between predictive performance and computational expenditure, or address a particular challenge, and are therefore interesting for specific combinations of dataset and metric. However, instead of such an evaluation, we make some general recommendations.

Figure 14 and Table 4 outline the main varieties to be chosen from and (in the table) their respective computational complexity, with regard to both training and testing phases. The complexity is considered relative to the size of the label set, L (thus we do not deal specifically with subsampling strategies that may affect the size of the input instance space). Clearly, as L becomes larger, more consideration must be made toward computationally tractable training and inference. However, aside from this spectrum, there are other important aspects worth highlighting. If the metric of predictive performance evaluates labels independently of each other, as for example Hamming loss does, then less chain structure is necessary in general, but the base classifier should be sufficiently powerful and non-linear. On the other hand, a weak linear base classifier will almost always benefit from increased connectivity, and more rigorous inference.

A number of classifier chain 'recipes' are suggested in Table 5. These suggested configurations still leave room for finer-grained parametrization such as the ϵ or beam-width of the search, hyper-parametrization of base classifiers, and so on. Indeed, each recipe does not necessarily correspond to a particular paper from the literature, although some specific example references are given (also in in Table 4 and throughout the text of this paper). Rather, these can be seen as a way to roll the review material of this paper into a toolbox comprising a number of high-level recommendations suitable for many real-world problems.

Even though particular configurations of classifier chains scale up to fairly large datasets, as discussed in Section 6, many large multi-label problems, especially of the 'extreme' variety, are increasingly better served by neural network architectures, which may, of course, incorporate elements of classifier chains.

Table 4: Complexity of (a) inference and (b) structuring strategies for L labels and M trained ensemble members ($M = 1$ corresponds to a single model), supposing fixed input space and base classifier. In (a), S is dependent on a chosen search strategy; and can be anywhere in $1 \leq S \leq 2^L$, i.e., ranging between greedy and brute-force complexity. Recall that, although measuring marginal dependence does not require training classifiers, computation is nevertheless required.

(a) Inference complexity (number of passes down the chain)

Inference	Iterations	Example reference
Greedy	$O(M)$	Read et al., 2009
Search	$O(M \cdot S)$	Mena et al., 2016
Exhaustive	$O(M \cdot 2^L)$	Dembczyński et al., 2010

(b) Training complexity, which involves making a number of marginal-dependence measurements (complexity denoted in $O_{\mathcal{M}}$ in terms of the number of measurements) and/or inducing models (complexity denoted in O in terms of the number of models). Note that we include the cost of the final models built for prediction (in addition to those built for measuring conditional dependence) in O. Three variations are listed for conditional dependence.

Structuring strategy	Complexity	Example reference
Random (default)	$O(M \cdot L)$	Read et al., 2011
Marginal Dependence	$O_{\mathcal{M}}(L^2) + O(M \cdot L)$	
Cond. Dependence	$O(L^2) + O(M \cdot L)$	
Cond. Dependence	$O(L) + O_{\mathcal{M}}(L^2) + O(M \cdot L)$	Zhang & Zhang, 2010
Cond. Dependence	$O(L)$	Teisseyre, 2017
Based on Accuracy	$O(L) + O(M \cdot L)$	See Section 4.4
Search‡ (fixed structure)	$O(M \cdot L!)$	Read et al., 2015
Search‡ (free structure)	$O(2^{L^2})$	Gasse, 2017

‡ Actual complexity depends on chosen search algorithm

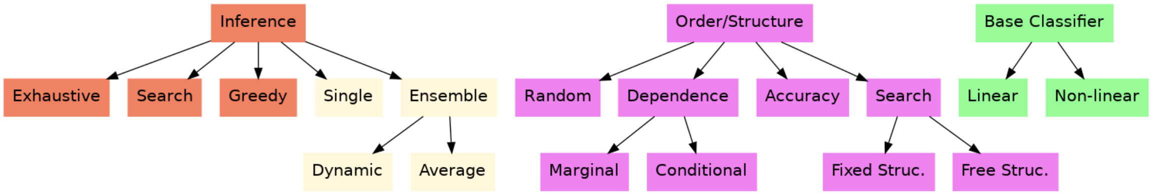


Figure 14: Configurations of classifier chains in terms of inference and single model vs. ensembles (left), chain order/structure (mid.), and base classifier (right). In addition, the quality of posterior probability/confidence of the base classifier should also be considered for non-greedy inference (not shown).

8. Conclusion

Over a decade after initial interest in *classifier chains* as a method for multi-label classification, novel developments and analyses and fresh applications continue to appear in

Table 5: Some suggested classifier chain recipes combining the results of numerous papers (references given in Table 4). Obviously a plethora of other options are also possible. Refer to Section 3 and Section 4 for details on Inference and Structure, respectively. If not specified, the base classifier is assumed selected and parametrized according to practitioner preference, and problem domain and dimensions.

Name	Inference		Order/Structure	Base Classifier
The Baseline	Greedy	Ensemble	Random	SVM or trees
As by Read et al. (2009)				
The Kaggler	Greedy	Ensemble	Random/Sparse	A mixture
A large subspace-ensemble of random diverse sparse chains; with a variety of base models.				
A Good Order	Search	Single		Logistic regression
Good probability estimates help with a search for one powerful fully-cascaded model.				
Neural Net	Greedy	Single		L_2 logistic reg.
A full cascade for predictive power, with regularization. Essentially a neural network.				
Neural Net Sparse	Greedy			L_1 logistic reg.
Single model, L_1 regularization takes care of pruning, as in Teisseyre (2017).				
Sparse & interpretable	Greedy	Single		Decision trees
Use a conditional dependence heuristic (Zhang & Zhang, 2010), for sparse structure.				
Expensive & effective	Search			Linear, or mix
A multiple-start structure search. Reuse models in a dynamic ensemble at inference time.				

the literature. Particular variations of chaining continue to attain competitive and often state-of-the-art performance on many multi-label datasets. New mechanisms for training and inference have been developed and have now also been adapted to other areas, such as multi-output regression.

The rise of ubiquitous access to neural network frameworks and associated hardware acceleration has begun to overshadow the option of off-the-shelf classifier chains for very large datasets. Nevertheless, as is also the case in relation to many other areas, there can be mutual benefit and shared development between deep neural and chaining approaches. In addition to this, one should keep in mind that only a subset of newly emerging datasets can be considered better suited to treatment under deep neural architectures, and therefore we can expect classifier chaining to continue to be relevant, thereby justifying the review of the methodology which we have carried out in this paper.

Furthermore, we may remark that there are many issues found in multi-label contexts that directly relate to classifier chains, such as weak labels, class imbalance and interpretability of label relations discovered (and how they relate to and can provide insight on the underlying application domain). These thematics are far from considered solved, and new issues are coming to the forefront. We speculate that numerous papers will continue to appear to confront them.

Acknowledgements

Thanks to Tomasz Kajdanowicz and Willem Waegeman who pointed out useful references and provided insightful discussion on classifier chains during the preparation of this paper. Thanks also to all five anonymous reviewers, each of whom provided important discussion and suggestions, and pointed out relevant material and references, which contributed towards the development of this paper.

References

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, *5*(5), 216–233.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Burkhardt, S., & Kramer, S. (2015). On the spectrum between binary relevance and classifier chains in multi-label classification. In *SAC 2015: 30th ACM Symposium on Applied Computing*, pp. 885–892. ACM.

Cheng, W., & Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, *76*(2-3), 211–225.

Cisse, M., Al-Shedivat, M., & Bengio, S. (2016). Adios: Architectures deep in output space. In *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48, pp. 2770–2779, New York, New York, USA. PMLR.

da Silva, P. N., Gon calves, E. C., Plastino, A., & Freitas, A. A. (2014). Distinct chains for different instances: An effective strategy for multi-label classifier chains. In Calders, T., Esposito, F., Hüllermeier, E., & Meo, R. (Eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 453–468, Berlin, Heidelberg. Springer Berlin Heidelberg.

Daumé, III, H., Langford, J., & Marcu, D. (2009). Search-based structured prediction. *Machine Learning*, *75*(3), 297–325.

Dembczyński, K., Cheng, W., & Hüllermeier, E. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML '10: 27th International Conference on Machine Learning*, pp. 279–286, Haifa, Israel. Omnipress.

Dembczyński, K., Kotlowski, W., Waegeman, W., Busa-Fekete, R., & Hüllermeier, E. (2016). Consistency of probabilistic classifier trees. In *ECML-PKDD 2016 : machine learning and knowledge discovery in databases*, Vol. 9852, pp. 511–526. Springer.

Dembczyński, K., Waegeman, W., Cheng, W., & Hüllermeier, E. (2012a). On label dependence and loss minimization in multi-label classification. *Machine Learning*, *88*(1-2), 5–45.

Dembczyński, K., Waegeman, W., & Hüllermeier, E. (2012b). An analysis of chaining in multi-label classification. In *ECAI: European Conference of Artificial Intelligence*, Vol. 242, pp. 294–299. IOS Press.

Dietterich, T. G. (2002). Machine learning for sequential data: A review. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pp. 15–30, London, U.K. Springer-Verlag.

Doppa, J. R., Fern, A., & Tadepalli, P. (2014a). HC-search: A learning framework for search-based structured prediction. *Journal of Artificial Intelligence Research*, *50*, 369–407.

Doppa, J. R., Yu, J., Ma, C., Fern, A., & Tadepalli, P. (2014b). HC-search for multi-label prediction: An empirical study. In Brodley, C. E., & Stone, P. (Eds.), *AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pp. 1795–1801. AAAI Press.

Enrique Sucar, L., Bielza, C., Morales, E. F., Hernandez-Leal, P., Zaragoza, J. H., & Larrañaga, P. (2014). Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters*, *41*(C), 14–22.

Frank, E., & Kramer, S. (2004). Ensembles of nested dichotomies for multi-class problems. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML 04, p. 39, New York, NY, USA. Association for Computing Machinery.

Gasse, M. (2017). *Probabilistic Graphical Model Structure Learning : Application to Multi-Label Classification.* Theses, Université de Lyon.

Goncalves, E. C., Plastino, A., & Freitas, A. A. (2013). A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pp. 469–476.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* The MIT Press.

Guo, Y., & Gu, S. (2011). Multi-label classification using conditional dependency networks. In *IJCAI '11: 24th International Conference on Artificial Intelligence*, pp. 1300–1305. IJCAI/AAAI.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Jun, X., Lu, Y., Lei, Z., & Guolun, D. (2019). Conditional entropy based classifier chains for multi-label classification. *Neurocomputing*, *335*, 185 – 194.

Kajdanowicz, T., & Kazienko, P. (2009). Hybrid repayment prediction for debt portfolio. In Nguyen, N. T., Kowalczyk, R., & Chen, S.-M. (Eds.), *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, pp. 850–857, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kajdanowicz, T., & Kazienko, P. (2013). Heuristic classifier chains for multi-label classification. In Larsen, H. L., Martin-Bautista, M. J., Vila, M. A., Andreasen, T., & Christiansen, H. (Eds.), *Flexible Query Answering Systems*, pp. 555–566, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kiritchenko, S., Matwin, S., Nock, R., & Famili, F. (2006). Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Proc. of the 19th Canadian Conference on Artificial Intelligence*, pp. 395–406.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *Proc. of the 18th European Conf. on Machine Learning*, ECML '07, pp. 624–631, Berlin, Heidelberg. Springer-Verlag.

Kumar, A., Vembu, S., Menon, A., & Elkan, C. (2013). Beam search algorithms for multilabel learning. *Machine Learning*, *92*(1), 65–89.

Leathart, T., Frank, E., Pfahringer, B., & Holmes, G. (2019). On calibration of nested dichotomies. In *Advances in Knowledge Discovery and Data Mining - 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, April 14-17, 2019, Proceedings, Part I*, pp. 69–80.

Li, C.-L., & Lin, H.-T. (2014). Condensed filter tree for cost-sensitive multi-label classification. In Xing, E. P., & Jebara, T. (Eds.), *Proceedings of the 31st International Conference on Machine Learning*, Vol. 32 of *Proceedings of Machine Learning Research*, pp. 423–431, Bejing, China. PMLR.

Lin, W., & Xu, D. (2016). Imbalanced multi-label learning for identifying antimicrobial peptides and their functional types. *Bioinformatics*, *32 24*, 3745–3752.

Liu, B., & Tsoumakas, G. (2018). Making classifier chains resilient to class imbalance. In Zhu, J., & Takeuchi, I. (Eds.), *Proceedings of The 10th Asian Conference on Machine Learning*, Vol. 95 of *Proceedings of Machine Learning Research*, pp. 280–295. PMLR.

Liu, J., Chang, W.-C., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pp. 115–124, New York, NY, USA. ACM.

Loza Mencía, E., & Janssen, F. (2016). Learning rules for multi-label classification: a stacking and a separate-and-conquer approach. *Machine Learning*, *105*(1), 77–126.

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*(9), 3084–3104.

Mena, D., Montañés, E., Quevedo, J. R., & Coz, J. J. (2016). An overview of inference methods in probabilistic classifier chains for multilabel classification. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, *6*(6), 215–230.

Molnar, C. (2019). *Interpretable Machine Learning.* https://christophm.github.io/interpretable-ml-book/.

Montiel, J., Read, J., Bifet, A., & Abdessalem, T. (2018). Scikit-MultiFlow: A multi-output streaming framework. *Journal of Machine Learning Research*, *19*(72), 1–5.

Moyano, J. M., Galindo, E. L. G., Cios, K. J., & Ventura, S. (2018). Review of ensembles of multi-label classifiers: Models, experimental study and prospects. *Information Fusion*, *44*, 33–45.

Nam, J., Kim, J., Mencía, E. L., Gurevych, I., & Fürnkranz, J. (2014). Large-scale multilabel text classification - revisiting neural networks. In *ECML-PKDD '14: 25th European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 437–452.

Nam, J., Loza Mencía, E., Kim, H. J., & Fürnkranz, J. (2017). Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in Neural Information Processing Systems 30*, pp. 5413–5423.

Narassiguin, A., Elghazel, H., & Aussem, A. (2017). Dynamic ensemble selection with probabilistic classifier chains. In Ceci, M., Hollmén, J., Todorovski, L., Vens, C., & Džeroski, S. (Eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 169–186, Cham. Springer International Publishing.

Park, L. A. F., & Read, J. (2018). A blended metric for multi-label optimisation and evaluation. In *ECML-PKDD 2018: 29th European Conference on Machine Learning*, pp. 719–734.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pinto, F., Soares, C., & Mendes-Moreira, J. (2016). Chade: Metalearning with classifier chains for dynamic combination of classifiers. In Frasconi, P., Landwehr, N., Manco, G., & Vreeken, J. (Eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 410–425, Cham. Springer International Publishing.

Powell, W. B. (2019). A unified framework for stochastic optimization. *European Journal of Operational Research*, *275*(3), 795 – 821.

Puurula, A., Read, J., & Bifet, A. (2014). Kaggle LSHTC4 winning solution. Tech. rep., Kaggle LSHTC4 Winning Solution. Report on our winning solution to the LSHTC4 Kaggle Competition.

Ramrez-Corona, M., Sucar, L. E., & Morales, E. F. (2014). Chained path evaluation for hierarchical multi-label classification..

Read, J., & Hollmén, J. (2017). Multi-label classification using labels as hidden nodes. Tech. rep. 1503.09022v3, ArXiv.org. ArXiv.

Read, J., & Martino, L. (2020). Probabilistic regressor chains with Monte-Carlo methods. *Neurocomputing*, *In Press*, 1–26.

Read, J., Martino, L., & Hollmén, J. (2017). Multi-label methods for prediction with sequential data. *Pattern Recognition*, *63*(March), 45–55.

Read, J., Martino, L., & Luengo, D. (2014). Efficient Monte Carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, *47*(3), 1535–1546.

Read, J., Martino, L., Olmos, P. M., & Luengo, D. (2015). Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition*, *48*(6), 2096–2109.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier chains for multi-label classification. In *ECML 2009: 20th European Conference on Machine Learning*, pp. 254–269. Springer.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359.

Read, J., Puurula, A., & Bifet, A. (2014). Multi-label classification with meta labels. In *ICDM'14: IEEE International Conference on Data Mining*, pp. 941–946. IEEE.

Rivolli, A., Read, J., Soares, C., Pfahringer, B., & de Carvalho, A. C. P. L. F. (2020). An empirical analysis of binary transformation strategies and base algorithms for multi-label learning. *Machine Learning*, *In Press*(1573-0565), 1–55.

Scanagatta, M., Salmerón, A., & Stella, F. (2019). A survey on bayesian network structure learning from data. *Progress in Artificial Intelligence*, pp. 425–439.

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

Senge, R., del Coz, J. J., & Hüllermeier, E. (2014). On the problem of error propagation in classifier chains for multi-label classification. In Spiliopoulou, M., Schmidt-Thieme, L., & Janning, R. (Eds.), *Data Analysis, Machine Learning and Knowledge Discovery*, pp. 163–170, Cham. Springer International Publishing.

Szymański, P., & Kajdanowicz, T. (2017). A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints*.

Teisseyre, P. (2017). CCnet: Joint multi-label classification and feature selection using classifier chains and elastic net regularization. *Neurocomputing*, *235*, 98 – 111.

Tenenboim-Chekina, L., Rokach, L., & Shapira, B. (2013). Ensemble of feature chains for anomaly detection. In *Multiple Classifier Systems*, Vol. 7872 of *Lecture Notes in Computer Science*, pp. 295–306. Springer Berlin Heidelberg.

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

Waegeman, W., Dembczyński, K., & Hüllermeier, E. (2019). Multi-target prediction: a unifying view on problems and methods. *Data Mining and Knowledge Discovery*, *33*(2), 293–324.

Wever, M., Tornede, A., Mohr, F., & Hüllermeier, E. (2020). Libre: Label-wise selection of base learners in binary relevance for multi-label classification. In Berthold, M. R., Feelders, A., & Krempl, G. (Eds.), *Advances in Intelligent Data Analysis XVIII*, pp. 561–573, Cham. Springer International Publishing.

Wydmuch, M., Jasinska, K., Kuznetsov, M., Busa-Fekete, R., & Dembczynski, K. (2018). A no-regret generalization of hierarchical softmax to extreme multi-label classification. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 31*, pp. 6355–6366. Curran Associates, Inc.

Zaragoza, J. H., Sucar, L. E., Morales, E. F., Bielza, C., & Larrañaga, P. (2011). Bayesian chain classifiers for multidimensional classification. In *24th International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 2192–2197.

Zhang, M.-L., Li, Y.-K., Liu, X.-Y., & Geng, X. (2018). Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, *12*(2), 191–202.

Zhang, M.-L., & Zhang, K. (2010). Multi-label learning by exploiting label dependency. In *KDD '10: 16th ACM SIGKDD International conference on Knowledge Discovery and Data mining*, pp. 999–1008. ACM.

Zhang, M.-L., & Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837.