2016-11-01

A Common Misconception in Multi-Label Learning

Michael Benjamin Brodie
Brigham Young University

A Common Misconception in Multi-Label Learning

Michael Benjamin Brodie

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Tony Martinez, Chair
Christophe Giraud-Carrier
Quinn Snell

Department of Computer Science

Brigham Young University

ABSTRACT

A Common Misconception in Multi-Label Learning

Michael Benjamin Brodie
Department of Computer Science, BYU
Master of Science

The majority of current multi-label classification research focuses on learning dependency structures among output labels. This paper provides a novel theoretical view on the purported assumption that effective multi-label classification models must exploit output dependencies. We submit that the flurry of recent dependency-exploiting, multi-label algorithms may stem from the deficiencies in existing datasets, rather than an inherent need to better model dependencies. We introduce a novel categorization of multi-label metrics, namely, evenly and unevenly weighted label metrics. We explore specific features that predispose datasets to improved classification by methods that model label dependence. Additionally, we provide an empirical analysis of 15 benchmark datasets, 1 real-life dataset, and a variety of synthetic datasets. We assert that binary relevance (BR) yields similar, if not better, results than dependency-exploiting models for metrics with evenly weighted label contributions. We qualify this claim with discussions on specific characteristics of datasets and models that render negligible the differences between BR and dependency-learning models.

Keywords: Multi-label classification, chain classifier, conditional dependence

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

In machine learning tasks, most models map a number of input variables, $\bar{x}$, to predict a single output class, y. In binary classification problems, this output takes on two possible values, 0 or 1. In multi-class problems, however, the output may have numerous possible values.

$$Binary : \bar{x} => y \in \{0, 1\}$$

$$Multi - Class : \bar{x} => y \in \{y_1, y_2, ..., y_n\}$$

These two types of outputs have corollaries in an increasingly common machine learning problem, multiple output classification. Rather than map inputs to a single output, multiple output classification models map inputs to several outputs - each with binary or several possible classes. The field of multi-label classification encompasses problems with multiple binary outputs. We describe this formally as

$$Multi - label : \bar{x} => \bar{y} \in \{0, 1\}^k$$

where k is the number of predicted output labels. Researchers have successfully used multi-label classification models for text-classification and scene labeling [1]. Multi-dimensional classification models handle those tasks that assign instances to a set of output classes, each with two or more possible values [29]. In our formal definition,

$$Multi - dimensional : \bar{x} => \bar{y}, y_k \in \{y_1, y_2, ..., y_n\}$$

each of the k output classes takes n possible values, where n is the number of values associated with the k^{th} output class.

Researchers often handle multi-dimensional output problems by simply converting them into multi-label problems [24]. For instance, an output variable, color, with three possible values $\{red, blue, yellow\}$ can be converted into three separate binary variables: $hasRed$ $\{0, 1\}$, $hasYellow$ $\{0, 1\}$, and $hasBlue$ $\{0, 1\}$. Admittedly, some output variables are best embodied as multi-dimensional variables. For instance, output variables representing the month and season are more understandably and efficiently represented as two multi-dimensional variables, rather than sixteen binary variables. However, we focus specifically on multi-label classification because the majority of previous research and current datasets focus on this method. Furthermore, while we can easily convert from multi-dimensional to multi-label output format, we often cannot perform a reverse conversion as easily.

As noted by [27], most multi-label classification approaches fall into two categories: Data transformation and algorithm adaptation. Regardless of the category, these methods generally introduce a new algorithm under the assumption that models will yield better predictions by exploiting dependencies among outputs. The authors of [2, 3, 5] challenge this assumption by providing theoretical and empirical evidence that multi-label algorithms often cannot simultaneously minimize multiple loss functions. They implore future researchers to create models that exploit a specific type of label dependency, rather than general models that seek to handle any form of multi-label datasets.

We similarly confront the controversial assumption that models can achieve improved results by learning and modeling dependencies among outputs. We note that [5] uses the statistical notation used in [8] for representing multivariate regression. As a result, [5] focuses its analysis on potential dependencies introduced by the stochastic error term, ϵ, used in the multivariate representation. We introduce our own notation, however, which in turn affects our analysis, conclusions, and recommendations for future work.

The novel contributions of this paper include a new categorization of multi-label metrics into evenly and unevenly weighted label metrics. Using this metric distinction, we provide a theoretical discussion of the purported claim of the importance of exploiting dependencies in multi-label classification. We also provide an empirical analysis of 15 benchmark datasets, 1 real-life dataset from InsideSales.com, and a variety of synthetic datasets. Finally, we synthesize the results of our experiments and assert that binary relevance (BR) yields similar, if not better, results than dependency-exploiting models for metrics with evenly weighted label contributions. We further note that with an appropriate submodel, relevant features, and a large number of instances, the differences between BR and dependency-learning models are negligible.

Of course, we recognize that some tasks require the learning and use of output dependency information for accurate classification. These tasks include, but are not limited to, multidimensional planning and scheduling tasks. One particularly promising area of future research involves classification problems with multiple correct multidimensional outputs. These types of tasks would require new approaches for learning, predicting, and measuring performance. We discuss this branch of classification, which we refer to as multidimensional output dependency (MOD) learning, in Chapter 5 and suggest paths for future work.

The remainder of this paper is outlined as follows. Chapter 2 describes previous approaches to to multi-label classification. We focus on the ubiquitous claim that effective models must exploit label dependencies. Chapter 3 discusses the distinction between conditional and unconditional dependencies in multi-label datasets. We then provide a novel theoretical analysis of dependencies in multi-label classification. Chapter 4 details an extensive empirical analysis of a variety of real-world and synthetic datasets and Chapter 5 describes future work.

Chapter 2: Related work

The binary relevance (BR) model is one of the most basic approaches to handling multi-label prediction problems [16, 23]. This method independently trains n models to predict n outputs, which are aggregated to produce a final prediction. Although BR models are quick to construct and easily parallelizable, many criticize the method because it ignores dependencies that may exist among outputs. This supposed shortcoming of BR led to the unverified, yet nearly universally accepted, assumption that the most probable path to improvements in multi-label classification will involve learning dependency structures among labels.

Along with BR, the label powerset (LP) method [6] serves as common baseline comparison in multi-label tasks. LP creates compound variables to model all possible combinations of the output variables. While this method implicitly accounts for label dependencies and correlations among outputs, the number of possible labels, $2^{|L|}$, where $|L|$ represents the number of labels, increases exponentially with each additional output. Because of this, LP is infeasible for tasks with a large number of output variables.

Although some methods, such as BR and LP, simply transform the data in order to handle multi-label tasks, most approaches adapt existing models or introduce new models to learn dependencies and concurrently predict outputs. One of the most popular approaches to model dependencies among outputs is the chain classifier (CC) [21]. This model mirrors BR by using n models for n outputs. Unlike BR, however, a CC cascades previous predictions along the chain of models. This means that each model in the chain learns dependencies among the previously predicted outputs.

The most common chain classifier models, including those found in multi-output prediction libraries such as MEKA [20] and MULAN [28], rely on a random chain ordering [7]. A number of papers employ empirical studies to argue that CC improves upon BR prediction. However, they acknowledge that randomly ordered CCs may or may not correctly capture dependencies among outputs. In efforts to overcome this limitation, Read et al. [21] introduced the Ensemble of Chain Classifiers (ECC), which increases prediction accuracies by employing a voting mechanism among several randomly ordered chain classifiers. Once again, however, this approach may fail to model the actual dependency structure among output variables. Furthermore, ECCs require additional resources to handle the computations for several linked CC models.

Zaragoza et al. [29] developed Bayesian Chain Classifiers (BCC) to more accurately learn output dependencies. The BCC uses a directed acyclic graph, or Bayesian Network, to first learn the dependencies among output variables. The BCC then uses that network to construct an ordering for the chain of models. Because of limits placed on the network to simplify the overall model, each link in the BCC only incorporates one additional class. This precludes the model from learning more complex dependency structures. Without the imposed simplifications, BCC models can learn more complex interactions among variables. However, the running time quickly becomes impractical since the BCC explores all 2^n possible chain orderings, where n is the number of outputs [2, 22]. Although ensembles of simplified BCC models can boost performance and avoid exponential running time, they still may not capture dependencies that involve two or more influencing output variables.

Read et al. [22] introduced a Super-Class (SC) model that clusters similar outputs to learn output dependencies with increased speed. This model uses simulated annealing to balance the process of exploiting discovered dependencies and exploring possible dependencies. In general, the SC approach reduces the enormous dependency search space and finds good local minima solutions. Much like randomly ordered chain classifiers, however, the SC model

may fall into a bad local minimum and produce poor prediction results. The approach may also experience problems similar to the LP method, such as overfitting.

One of the most recent models, the Classifier Trellis (CT) [24], places outputs into a predefined dependency structure, which the model gradually improves upon while training. This structure eliminates the need to solve the NP-hard problem of discovering a dependency network with no prior knowledge. The CT gradually improves upon this predefined network until reaching a satisfactory structure. Although the CT, on average, performs worse than an ECC, the predefined structure allows CT to scale to datasets with thousands of output variables. Nevertheless, because CT uses the pairwise matrix of mutual information to compute the dependence between random variables, CT cannot model dependencies that involve more than two variables, which limits the learning power of the model.

In addition to providing new approaches for multi-label classification, the papers for CC, BCC, SC, and CT reinforced the assumption that future improvements in the field would discover better representations of label dependencies. This belief resulted in a surge of papers with new schemes for ordering chain classifiers [7, 10, 12, 14, 15, 25, 29] and formulating new dependency structures. In harmony with [5], we question the value of creating additional dependency-based models without first validating the underlying assumption that models benefit by exploiting label dependencies. In an effort to answer this question, the next section details a theoretical analysis of the basic premise underlying recent multi-label prediction approaches.

2.1 Multi-label metrics

Existing research details numerous metrics for comparing and assessing the performance of new multi-label classifiers. Despite the varied forms of these metrics, they generally fall into two categories, bipartitions-based and rankings-based [17]. Multi-label ranking tasks deal with both multi-label classification and label ranking. Due to the added complexity

introduced by label ranking, we focus our attention on bipartitions-based metrics and leave rankings-based analysis for future work.

Bipartitions-based metrics often extend familiar binary classification metrics to appropriately handle multi-label data. Following the example of [17], we further divide bipartitions-based metrics into two categories: Label-based and example-based. Label-based metrics measure the performance of labels individually and report the averaged performance across all labels. Example-based metrics, on the other hand, assess the difference between the actual and predicted sets of labels. Table 2.1 contains two label-based and two example-based metrics that we use for our experiments in Section 4. Below we define each of these four metrics.

Table 2.1: Bipartitions-based metrics

Example-based	**Label-based**
Hamming loss	Micro F1
0/1 loss	Macro F1

2.1.1 Hamming loss

One of the most commonly reported multi-label metrics, hamming loss, measures the symmetric difference the predicted and actual label sets. We define this as

$$\text{Hamming Loss} = 1 - \frac{1}{|N||L|}\sum_{n=1}^{|N|}\sum_{i=1}^{|L|}(\hat{y}_i^n \oplus y_i^n)$$

where N is the total number of instances, L is the number of labels, and $\oplus$ returns the logical equality of $\hat{y}_i^n$ and y_i^n.

2.1.2 0/1 loss

Another much stricter, yet oft-reported measure is 0/1 loss, which corresponds to the exact match [7], subset accuracy [13], or example accuracy [22] scores in previous literature. This

metric reports the percentage of predicted label sets that contain an error. We represent this as

$$0/1 \text{ Loss} = 1 - \frac{1}{|N|} \sum_{n=1}^{|N|} \mathbf{1}(\hat{y}^n = y^n)$$

where $\mathbf{1}()$ returns 1 if the predicted $\hat{y}^n$ vector is identical to y^n.

2.1.3 F_1 score (macro-averaged)

The macro-averaged F_1 score is a familiar extension of the F_1 score for binary classification. To compute this metric, however, we average the F_1 scores of all label columns in the data. Similar to traditional F_1 scores, the macro-F_1 score for each label is simply the harmonic mean between precision, p_i, and recall, r_i, for the i^{th} label column:

$$F_{1_i} = \frac{2 \times p_i \times r_i}{p_i + r_i}$$

We then calculate the macro F_1 score by averaging the scores for each label. We denote this as

$$\text{Macro } F_1 = \frac{1}{L} \sum_{i=1}^{|L|} F_{1_i}$$

where L is the total number of label columns.

2.1.4 F_1 score (micro-averaged)

In contrast to Macro-F_1, Micro-F_1 computes statistics globally for all labels and instances. We express this mathematically as

$$\text{Micro } F_1 = \frac{\sum_{n=1}^{|N|} \sum_{i=1}^{|L|} \hat{y}_i^n \times y_i^n}{\sum_{n=1}^{|N|} \sum_{i=1}^{|L|} \hat{y}_i^n + \sum_{n=1}^{|N|} \sum_{i=1}^{|L|} y_i^n}$$

where $\hat{y}_i^n$ and y_i^n are the predicted and actual values, respectively, for label i and instance n. As aptly noted by [26], label columns with few examples strongly affect macro-F_1, whereas labels with many examples exert greater influence on micro-F_1.

We further distinguish the metrics in Table 2.1 by differentiating between even and uneven label weighting. We attribute even weighting to those measures that allot equal weight to each individual label. For example, hamming loss computes an average over both labels and instances, which gives all label predictions equal weighting in the metric calculation. Similarly, the micro-averaged F_1 score computes precision and recall statistics globally and equally weights the contributions of labels. At the other end of the spectrum, macro-averaged F_1 score and $0/1$ loss unevenly weight individual labels by allowing columns with fewer positive examples to more strongly influence these performance measures. Table 2.2 updates Table 2.1 with this added distinction.

Table 2.2: Bipartitions-based metrics differentiated by label weighting

	Example-based	**Label-based**
Even-weighting	Hamming loss	Micro F1
Uneven-weighting	$0/1$ loss	Macro F1

Chapter 3: Theoretical discussion

An increasing number of papers make efforts to provide a basic theoretical foundation for their proposed multi-label prediction models. In general, these theoretical discussions distinguish between two types of dependencies within multi-label data sets: Unconditional and conditional. Before introducing our own label dependency analysis, we define these terms:

Definition 1. *Unconditional Dependence*: Labels $Y = y_1, y_2 \dots y_k$ are considered unconditionally independent if $P(Y) = \prod_{i=1}^{k} P(y_i)$. If, however, the output label Y contains y_i and y_j such that $P(y_i, y_j) \neq P(y_i)P(y_j)$, we say that the labels contain an unconditional dependency.

Definition 2. *Conditional Dependence*: Labels $Y = y_1, y_2 \dots y_k$ are conditionally independent if $P(Y|x) = \prod_{i=1}^{k} P(y_i|x)$. Mirroring the previous definition, if the output label Y contains y_i and y_j such that $P(y_i, y_j|x) \neq P(y_i|x)P(y_j|x)$, we say that the labels are conditionally dependent.

Despite the similarities between these two types of dependencies, the presence of one form of dependency does not imply the other. In other words, a dataset can have unconditional dependencies without conditional dependencies, and vice versa. [4] stipulates that future models should address a specific type of dependence and minimize a clearly defined loss function. They cast doubt on the effectiveness of blanket models that claim to yield better results based on a wide variety of metrics. Both [18] and [5], after discussing conditional and unconditional dependencies, defend BR on the basis that it can perform well with a suitable base learner. To the best of our knowledge, [5] provides the most

thorough analysis of multi-label dependence, which they base on multivariate regression from [8]. [5] conjectures that the stochastic error terms in their multivariate setup can introduce conditional dependencies among labels. Their analysis and synthetic experiments suggest that BR performs best on datasets with none or some unconditional dependencies. We offer a simple, yet illuminating discussion that builds upon these previous theoretical analyses. Because our approach differs from former approaches, however, we gain new insights and resultant recommendations for future work.

For this analysis, we consider a hypothetical dataset, D, which contains inputs $\bar{X}$ and binary outputs $y_1 \dots y_k$. We assume that training and prediction use k models, which we denote as $m_1 \dots m_k$, to predict k outputs. In the case of BR, we construct $m_1 \dots m_k$ independently based on $\bar{X}$. For DE models, we assume that training can occur in parallel and the input to a particular model may include both $\bar{X}$ and other outputs. At prediction time, however, these DE models must execute in the order determined by the dependencies introduced in training. For instance, if model m_j predicts y_j and trains with both $\bar{X}$ and y_i as inputs, the model that predicts y_i, which we denote as m_i, must first predict y_i so the predicted value can be used as input to m_j. This requires at least one output to depend solely on $\bar{X}$ as input.

No Free Label Hypothesis. *Given an arbitrary multi-label dataset, D, that contains inputs $\bar{X}$ and labels $y_1 \dots y_k$, we construct DE models $m_1 \dots m_k$. At prediction time, all label predictions for an arbitrary m_i ultimately stem from the current input instance, $\bar{x}$. Therefore, prediction improvements of DE models for evenly weighted label metrics result from limitations of submodels, poor qualities of input features, and the number of output labels and training instances, rather than information learned from output dependencies.*

We next provide an informal discussion to detail the implications of this hypothesis. Consider two models, m_1 and m_2 that predict outputs y_1 and y_2, respectively. Let $d_{1,2}$ denote the level of dependence between these two outputs, where $d_{1,2} \in \{High, Low\}$. We use the notation $m_i > m_j$ to signify that m_i performs better than m_j. Likewise, $m_i \leq m_j$ denotes

that m_i performs as well or worse than m_j. If $d_{1,2}$ is Low, m_2 will not benefit from predicting with y_1 as an input, since y_1 will merely provide random noise. Similarly, if m_2 produces better results than m_1 using just $\bar{X}$ as input, using the prediction of y_1 as another input is likely to harm rather than help performance. However, if $d_{1,2}$ is $High$ and $m_2 \leq m_1$, m_2 may gain a performance boost by exploiting information provided by m_1. Table 3.1 displays these various scenarios.

Table 3.1: Benefit of exploiting dependencies based on correlation and model performance

		m_2	
		$> m_1$	$\leq m_1$
$d_{1,2}$	$High$	No	$Possibly$
	Low	No	No

This suggests that DE models do not necessarily have an advantage over simpler BR models. Nevertheless, numerous empirical studies provide evidence to the contrary, namely, that multi-label prediction models perform best by exploiting some form of dependence among outputs. In efforts to remove the discrepancy between our theoretical analysis and others' empirical results, we examine two scenarios in which BR models may fail to match the performance of DE models.

We first investigate a situation in which input features are not well-suited for predicting output labels. For instance, consider again the dataset involving $\bar{X}$ and y_1 and y_2. Assume that a BR model cannot perform better than random when mapping $\bar{X}$ to y_1 or y_2. Table 3.2 presents two toy datasets that demonstrate this situation. If D does not contain an unconditional dependency between y_1 and y_2, the probability of predicting the correct output vector is $P(y_1)P(y_2) = (0.5)(0.5) = 0.25$.

Table 3.2: Example datasets for which DE models could outperform BR

$\bar{X}$	y_1	y_2
0	0	0
0	1	1

Model	Prediction
BR	$P(y_1)P(y_2) = (0.5)(0.5) = 0.25$
DE	$P(y_1, y_2) = 0.5$

$\bar{X}$	y_1	y_2
0	0	1
0	1	0

If y_1 and y_2 are fully unconditionally dependent, however, then there exist two functions f and g such that $y_1 = f(y_2)$ and $y_2 = g(y_1)$. This means that the probability of predicting a correct output vector is $P(y_1, y_2) = 0.5$. By learning and exploiting this dependency, a DE model could show slight improvements over the independence assuming BR model in terms of $0/1$ loss, an unevenly weighted label metric. The empirical results of previous work as well as our synthetic data experiments in Section 4 support this claim. We readily admit that DE models can achieve marginal prediction improvements by finding additional ways to learn unconditional dependencies among labels. Nevertheless, this approach ignores the larger, underlying problems: The input data is not well fitted to the multi-label prediction task, and the submodels may be inappropriate for predicting certain outputs. By focusing on these problems, future models may yield more substantial improvements.

Chapter 4: Experiments

The flood of new dependency-exploiting, multi-label algorithms may originate from defects in existing benchmark datasets, rather than weaknesses of existing models. Many recent papers introduce complex schemes for exploiting multi-label dependencies and compare their new approaches against a variety of existing models over a range of metrics. These papers almost invariably report results using a subset of the datasets found in the MEKA [20] or MULAN [28] frameworks. Although these datasets provide an easy-access resource for multi-label experiments, we submit that these datasets may not reflect the true nature of multi-label classification problems.

We propose a number of experiments to examine the presumed benefit of using a dependency-exploiting model, rather than a simple BR model. This builds upon the experimental results of [9], which provides experimental evidence that dependency-exploiting algorithm adaptations do not generally perform better than BR. Instead of comparing BR to numerous other dependency-based models, we focus our attention on the CC model introduced by [21]. We use CCs because they can reportedly capture both unconditional and conditional dependencies. We note that various papers introduce more complicated models [7, 12, 29] that attempt to overcome the possible disadvantage of CCs randomly selected chain order. We handle this issue by reporting results of all $n!$ possible chain orders. This necessarily restricts our analysis to those datasets with $2 \leq n \leq 7$ labels. However, we further bolster our claims with additional experiments on large datasets with a subset of chain orders.

For all experiments in sections 4.1 and 4.2, we use MEKA to perform 10-fold cross-validation with the default settings for the following 20 different WEKA submodels models for BR: Logistic, Naïve Bayes, BayesNet, MLP, REPTree, Random Tree, Random Forest, LMT, J48, Decision Stump, ZeroR, PART, OneR, JRip, Decision Table, SMO, AdaBoost, LWL, IBk, and Simple Logistic. We report the best, median, and worst scores for hamming loss, 0/1 loss, and macro and micro F1-scores across the various BR models. We then run all possible chain order combinations of CCs using the underlying models that produce the best, median, or worst BR metrics. For each of the CC models, we report the best, median, and worst scores for its corresponding BR metric. For instance, if the MLP underlying model for BR achieves the best hamming loss, we report the best, median, and worst hamming losses for all possible chain orders of a CC with a MLP base model. When multiple underlying models achieve the same score for BR, we run each of the CC models and average the results.

4.1 Synthetic data

We first generate two 10,000-instance datasets, toy_1 and toy_2, that model the scenarios shown in Table 3.2. As expected from the theoretical discussion in Section 3, BR yields a hamming loss of 0.5 for both datasets. For the toy_2 dataset, the majority of BR experiments produced a 0/1 loss of 1.0, which is consistent with our earlier analysis. A CC, on the other hand, learns the unconditional dependency between outputs y_1 and y_2 and reduces the 0/1 loss to 0.5. This harmonizes with our hypothesis that DE models generally outperform BR in terms of unevenly weighted label metrics, such as 0/1 loss.

We next recreate two 10,000-instance datasets using the method from [2]. This method creates datasets with independent and conditionally dependent labels using the algorithms outlined in Figure 4.1. We note that we can convert the supposed conditionally dependent

labels, y_2 and y_3, to independent labels with composite function transformations:

$$y_1 = \phi(x)$$

$$y_2 = \phi(-x - 2\phi(x) + 1)$$

$$y_3 = \phi(x + 12\phi(x) - 2\phi(-x - 2\phi(x) + 1) - 11)$$

where

$$\phi(a) = \begin{cases} 1 & \text{if } \sigma(a) > u \\ 0 & \text{otherwise} \end{cases}$$

and σ represents the sigmoid squashing function, $1/(1 + exp(-x))$, and $u \sim U(0,1)$. The use of x vectors of size one and two for the dependent and independent datasets, respectively, may limit the flexibility of the possible output label decision boundaries. However, we expect that an underlying model with enough degrees of freedom to approximate the sigmoid function can also learn these functions. We admit that the element of randomness introduced by the sampled value u will prevent models from flawlessly estimating the true values of y_1, y_2, and y_3. Due to this condition, we conjecture that models will not benefit from passing along the values of predicted labels to subsequent functions. Rather than improving results, models may propagate prediction errors and worsen the overall model performance.

Algorithm 1: Independent labels
$x_1, x_2 \sim U(-0.5, 0.5)$;
$y_1 = \phi(x_1 + x_2)$;
$y_2 = \phi(-x_1 + x_2)$;
$y_3 = \phi(x_1 - x_2)$;

Algorithm 2: Conditionally dependent labels
$x_1 \sim U(-0.5, 0.5)$;
$y_1 = \phi(x_1)$;
$y_2 = \phi(-x_1 - 2y_1 + 1)$;
$y_3 = \phi(x_1 + 12y_1 - 2y_2 - 11)$;

Figure 4.1: Methods of [2] for synthetic data generation

Table 4.1 contains the results for the synthetic data generated with algorithms 1 and 2 from Figure 4.1. We note that the synthetic nature of these datasets precludes us from forming strong conclusions about the general behavior of multi-label classification. Nevertheless,

Table 4.1: Results for BR and the best (CB), median (CM), and worst (CW) CC orderings using the independent (left) and dependent (right) synthetic data generation processes

		BR	CB	CM	CW
BEST	Hamming Loss	0.423	0.422	0.423	0.423
	0/1 Loss	0.809	0.809	0.810	0.810
	F_1 Micro	0.583	0.590	0.587	0.579
	F_1 Macro	0.508	0.589	0.585	0.579
MEDIAN	Hamming Loss	0.430	0.427	0.431	0.433
	0/1 Loss	0.814	0.814	0.814	0.816
	F_1 Micro	0.563	0.565	0.563	0.562
	F_1 Macro	0.489	0.568	0.560	0.559
WORST	Hamming Loss	0.483	0.480	0.482	0.485
	0/1 Loss	0.860	0.858	0.863	0.863
	F_1 Micro	0.510	0.510	0.510	0.510
	F_1 Macro	0.414	0.510	0.510	0.510

		BR	CB	CM	CW
BEST	Hamming Loss	0.423	0.430	0.431	0.437
	0/1 Loss	0.809	0.616	0.633	0.677
	F_1 Micro	0.583	0.537	0.536	0.430
	F_1 Macro	0.508	0.532	0.532	0.221
MEDIAN	Hamming Loss	0.430	0.430	0.433	0.437
	0/1 Loss	0.814	0.612	0.617	0.633
	F_1 Micro	0.563	0.536	0.534	0.431
	F_1 Macro	0.489	0.534	0.529	0.221
WORST	Hamming Loss	0.483	0.430	0.436	0.437
	0/1 Loss	0.860	0.613	0.617	0.633
	F_1 Micro	0.510	0.534	0.489	0.430
	F_1 Macro	0.414	0.529	0.435	0.221

the results highlight several important points to consider when deciding between a BR and DE model. For example, the independent label results in Table 4.1 show that BR often yields lower hamming and 0/1 losses than CC. This likely occurs because CC learns spurious relationships among the unrelated labels and propagates errors along the chain of models. Thus, when relationships do not exist among labels, CC may harm rather than aid predictions for both evenly and unevenly weighted label metrics. Interestingly, the best, median, and worst CC models improve over BR in terms of F_1 macro score on the independent label dataset. This may occur due to slight correlations amongst labels introduced by sampling. We expect that the F_1 macro performance gap between BR and DE will shrink with larger datasets that more closely approximate the underlying distribution produced by Algorithm 1.

In the dependent label experiment results shown in Table 4.1, CC yields similar hamming losses to BR, but decreases 0/1 loss by nearly 20% regardless of chain order. This demonstrates the power of exploiting unconditional dependencies to optimize an unevenly weighted label metric in datasets with poor input features. In both tables, CC generally yields higher F_1 macro scores. Similar to the independent label results, CC generally yields higher F_1 macro scores. This occurs because F_1 macro allows label columns with fewer positive examples to more strongly influence the F_1 score. CC likely learns correlations between less frequent labels and produces a slight improvement in per-label precision.

4.2 Small benchmark datasets

For this section, we used the Emotions, Scene, and Flag datasets from [28], as well as the Image dataset from [11], using its proposed method to convert from multiple instance multi-label to traditional multi-label format. We also used a real-life, 3-label dataset with 10,000 instances from InsideSales.com. The results of these experiments show that the majority of CC models with different chain orderings perform similar to or worse than a base BR model.

Admittedly, Table 4.2 shows that the best CC orderings using the best BR submodel on average achieve better results for the four metrics. However, these top scores represent a minute fraction of the $n!$ CC models and may simply result from a favorable but fortuitous random initialization of the underlying model. We note that the performance of the median and worst CC orderings using the best BR model performed between 0.28% and 11.2% worse than BR, with the exception of 0/1 loss for the median CC ordering. We reiterate that CC can achieve lower 0/1 losses than BR even if submodels perform poorly overall, since CC models can learn and reproduce unconditional dependencies found among outputs.

Table 4.2: Percentage improvement of the best, median, and worst performing CC chain orderings over the best, median, and worst performing BR models, averaged across four small benchmark and InsideSales datasets

	Best BR Model			**Median BR Model**			**Worst BR Model**		
	Best	*Median*	*Worst*	*Best*	*Median*	*Worst*	*Best*	*Median*	*Worst*
Hamming Loss	3.04	-1.644	-11.217	2.735	-2.978	-8.508	4.058	0.88	-1.633
0/1 Loss	5.471	2.147	-3.4	10.574	6.985	3.491	4.614	3.192	2.672
F_1 Micro	1.497	-0.28	-2.439	1.831	-0.329	-3.29	17.242	17.066	-27.376
F_1 Macro	1.962	-0.325	-2.312	4.238	0.787	-3.382	13.8	13.8	-29.405

Table A.2 shows the average percentage increase or decrease in metric performance for CC on each of the five small benchmark datasets. These results again demonstrate the ability of CC to improve unevenly weighted label metric scores while performing poorly otherwise. For instance, the results for the Flags, Emotions, Image, and Scene datasets show that CC achieves improved 0/1 losses but significantly worse hamming losses. Furthermore,

CC produced smaller F_1 macro and micro scores than BR, with the exception of the emotions dataset. We attribute the improved performance of CC on all metrics for the ISDC dataset to numerous missing values in the training data. Additional experiments are needed to analyze the effects of missing values on multi-label prediction models. However, these missing values likely inhibited the training of simpler submodels and lowered the average BR performance. We note similar learning patterns with other submodels: Across all chain orderings, the Naive Bayes model performed better using CC than with BR. On the other hand, the Logistic model performed worse with CC on almost all different chain orderings than when using BR.

Table 4.3: Average percentage improvement of all best, median, and worst results of chain orderings over underlying Binary Relevance (BR) models

	Flags	*Emotions*	*Image*	*Scene*	*ISDC*
Hamming Loss	**-2.353**	**-4.593**	**-2.67**	**-0.299**	1.434
0/1 Loss	5.796	1.722	2.733	5.706	3.901
F_1 Micro	**-0.819**	0.439	**-4.965**	**-0.218**	11.614
F_1 Macro	**-1.606**	1.024	**-4.134**	**-0.141**	4.393

4.3 Large benchmark datasets

Due to memory and running time limitations, we modified our experimental setup for the following MEKA and MULAN large benchmark datasets: Delicious, Mediamill, IMDB, NUS-Wide-500, NUS-Wide-128, Reuters, Birds, Enron, Yeast, Medical, and Genbase. We note that we used the full datasets rather than the given train and test set splits. If a single BR model took longer than a week to complete on a particular dataset, we randomly sampled a 5,000 instance subset. For these reduced datasets, we performed 5-fold cross-validation with 16 underlying models for BR, removing those models from the original 20 which generally took longer than three days to complete. We then ran five randomly produced chain combinations of CCs using the underlying models that produced the best BR metrics. When multiple

underlying models achieved the same score for BR, we ran CC for the model that performed best across all metrics.

Table 4.4: Counts of percentage improvement of best and average chain orderings using the best performing underlying Binary Relevance (BR) model

	Best Chain Ordering								
	>10%	>5%	>1%	>0%	0%	<0%	<-1%	<5%	<-10%
Hamming Loss	0	0	1	0	5	0	1	1	3
0/1 Loss	1	0	5	4	0	1	0	0	0
F_1 Micro	0	0	0	4	0	1	2	0	4
F_1 Macro	1	1	4	1	0	2	1	1	0
	Average Chain Ordering								
	>10%	>5%	>1%	>0%	0%	<0%	<-1%	<5%	<-10%
Hamming Loss	0	0	0	1	2	1	3	1	3
0/1 Loss	0	0	2	5	0	2	2	0	0
F_1 Micro	0	0	0	2	0	2	2	1	4
F_1 Macro	1	1	2	1	0	2	3	1	0

As Table 4.4 demonstrates, CC does not provide a decisive improvement over BR. Out of the 44 runs reported in the table, only one CC trained on the Yeast dataset decreased hamming loss. We note that the best CCs run on benchmark datasets with more than 7 labels but fewer than 2,500 instances - namely, Yeast, Genbase, Enron, Medical, and Birds - generally produced slightly improved F_1 macro scores and 0/1 losses. However, this benefit is much less pronounced when averaged across all chain orderings. The worst hamming and F_1 micro scores come from the datasets with large numbers of labels or instances. For datasets with numerous labels, CC likely produces poorer hamming and F_1 micro scores because the model propagates erroneous information along longer dependency chains. When datasets have many instances and provide stronger features to label signals, BR performance improves and minimizes the possible benefits of using a dependency model to optimize evenly weighted label metrics.

Table 4.5: Average percentage improvement of the best CC chain orderings over the best BR models

	Hamming Loss	*0/1 Loss*	F_1 *Micro*	F_1 *Macro*	*Average*
Delicious	−10.526	0.101	−13.740	−2.564	−4.330
IMDB	−19.444	3.866	−3.196	0.826	−4.487
Mediamill	0.000	0.221	−2.178	−0.552	−0.627
NUS-Wide-500	−4.762	3.856	−86.486	−5.983	−23.344
NUS-Wide-128	−8.696	0.625	−80.994	11.940	−19.281
Reuters	0.000	1.250	−21.245	4.762	−3.808
Birds	0.000	−0.438	0.914	3.711	1.047
Enron	0.000	1.367	0.899	6.667	2.233
Yeast	1.047	0.380	0.935	−0.662	0.425
Genbase	0.000	13.043	0.202	2.273	3.880
Medical	−11.111	1.656	−0.602	1.980	−2.019

As the results in Tables 4.5 and 4.6 demonstrate, BR tends to yield noticeably improved hamming losses and F_1 micro scores. CC, on the other hand, produces less pronounced improvements in 0/1 losses and F_1 macro scores. We echo the findings of previous work that the choice between DE and BR depends heavily on the desired metric to optimize. Although we have focused primarily on the observed differences between evenly and unevenly spaced metrics, the size of the dataset, quality of the features, number of labels, and selected submodel also seem to affect the choice of whether or not to exploit label dependency information. Further experiments are needed to isolate and understand the actual effects of these issues on the relative performance of CC and BR.

Table 4.6: Average percentage improvement of the average CC chain orderings over the average BR models

	Hamming Loss	*0/1 Loss*	*F_1 Micro*	*F_1 Macro*	*Average*
Delicious	−10.526	0.060	−14.809	−4.444	−4.970
IMDB	−22.222	3.490	−5.205	−0.331	−6.067
Mediamill	−1.333	−0.022	−2.396	−3.315	−1.767
NUS-Wide-500	−4.762	3.856	−87.081	−5.983	−23.492
NUS-Wide-128	−8.696	0.625	−81.228	11.940	−19.340
Reuters	0.000	0.656	−22.060	2.491	−4.728
Birds	−2.051	−2.407	−0.512	0.371	−1.150
Enron	−0.426	0.592	−1.727	5.689	1.032
Yeast	0.314	0.025	0.685	−1.280	−0.064
Genbase	0.000	−2.609	0.040	1.866	−0.176
Medical	−11.111	−0.596	−0.915	−0.198	−3.205

Table 4.7: Percentage improvement from the best and average chain ordering over the best performing underlying Binary Relevance (BR) model. Results averaged over all large benchmark datasets

	Best	Average
Hamming Loss	**-4.863**	**-5.528**
0/1 Loss	2.357	0.334
F_1 Micro	**-18.681**	**-19.564**
F_1 Macro	2.036	0.619

Chapter 5: Implications and Conclusion

This work highlights several weaknesses of current efforts to improve multi-label classification models. Unlike previous work, we do not introduce a new model to handle multi-label prediction tasks. Rather, we have provided an analysis of the defects of existing studies and introduced a distinction between evenly and unevenly weighted label metrics. We have also given a theoretical discussion and in-depth empirical analysis of existing datasets, which shows that CC is not necessarily better than BR. In fact, our experiments demonstrate clearly that BR often outperforms a randomly ordered CC. We conclude that the choice between BR and a dependency-exploiting model depends largely on the chosen metric to optimize. In the case of unevenly weighted label metrics such as 0/1 loss and F_1 macro score, dependency-learning models clearly outperform BR. However, when optimizing evenly weighted label metrics, for instance, hamming loss and F_1 micro score, the benefits of dependency models is negligible. In addition, our experiments show that with an appropriate submodel, relevant features, and a large number of instances, BR will yield nearly as good, if not better, results than more expensive dependency-exploiting models.

These findings indicate paths for improvement across a variety of metrics in multi-label classification. We submit that future research will benefit by shifting the focus from developing new, dependency-exploiting models to creating additional benchmark datasets with better features and more instances. Such an adjustment will allow future studies to more closely identify the nature of multi-label learning. We likewise emphasize the need for increased metric and domain-specialization, rather than blanket models that seek to improve across all

multi-label metrics and datasets. This approach will naturally promote the development of techniques better suited to individual domains, such as image, audio, or text.

In future work, we will analyze specific submodels to determine those which perform best with BR for certain metrics. We will also revisit and convert our claims in Chapter 3 into a formalized theorem with an accompanying proof. As a related but new direction, we will explore the field of multidimensional output dependency (MOD) learning, in which an input vector can map to multiple correct output vectors. MOD learning partially resembles existing areas of machine learning, for example, multi-label learning, multi-task learning, or transfer learning. However, unlike these problems, MOD learning seeks to map each input vector, $\bar{x}$, to multiple, highly dependent output vectors. To the best of our knowledge, only [10, 19] have addressed and experimented with MOD learning. Although their work proves the existence of MOD problems and demonstrates the benefits of MOD-based approaches, the majority of their results come from synthetic and private datasets.

In addition to creating and making new MOD datasets publicly available, we will introduce methods for detecting MOD characteristics in existing datasets. This will provide the means for future researchers to investigate and find new ways to apply MOD learning. We also plan to extend or introduce metrics to assess and validate MOD models. We expect that MOD learning will find valuable application in automated planning, scheduling, robotics, and classification tasks with multiple correct labelings. For these tasks, we submit that that successful approaches will both benefit from and require the use of dependency exploiting models in order to provide satisfactory results.

Appendix A: Large Dataset Results

Table A.1: Average percentage improvement of the best CC chain orderings over the best BR models

	Hamming Loss	*0/1 Loss*	F_1 *Micro*	F_1 *Macro*	*Average*
Delicious	−10.526	0.101	−13.740	−2.564	−4.330
IMDB	−19.444	3.866	−3.196	0.826	−4.487
Mediamill	0.000	0.221	−2.178	−0.552	−0.627
NUS-Wide-500	−4.762	3.856	−86.486	−5.983	−23.344
NUS-Wide-128	−8.696	0.625	−80.994	11.940	−19.281
Reuters	0.000	1.250	−21.245	4.762	−3.808
Birds	0.000	−0.438	0.914	3.711	1.047
Enron	0.000	1.367	0.899	6.667	2.233
Yeast	1.047	0.380	0.935	−0.662	0.425
Genbase	0.000	13.043	0.202	2.273	3.880
Medical	−11.111	1.656	−0.602	1.980	−2.019

Table A.2: Average percentage improvement of the average CC chain orderings over the average BR models

	Hamming Loss	*0/1 Loss*	F_1 *Micro*	F_1 *Macro*	*Average*
Delicious	−10.526	0.060	−14.809	−4.444	−4.970
IMDB	−22.222	3.490	−5.205	−0.331	−6.067
Mediamill	−1.333	−0.022	−2.396	−3.315	−1.767
NUS-Wide-500	−4.762	3.856	−87.081	−5.983	−23.492
NUS-Wide-128	−8.696	0.625	−81.228	11.940	−19.340
Reuters	0.000	0.656	−22.060	2.491	−4.728
Birds	−2.051	−2.407	−0.512	0.371	−1.150
Enron	−0.426	0.592	−1.727	5.689	1.032
Yeast	0.314	0.025	0.685	−1.280	−0.064
Genbase	0.000	−2.609	0.040	1.866	−0.176
Medical	−11.111	−0.596	−0.915	−0.198	−3.205

References

[1] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[2] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In Johannes Fürnkranz and Thorsten Joachims, editors, *International Conference on Machine Learning*, pages 279–286. Omnipress, 2010.

[3] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence in multi-label classification. In *Workshop Proceedings of Learning from Multi-Label Data*, pages 5–12, Haifa, Israel, June 2010.

[4] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. Regret analysis for performance metrics in multi-label classification: the case of Hamming and subset zero-one loss. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Lecture Notes in Artificial Intelligence 6321 Machine Learning and Knowledge Discovery in Databases: The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery 2010*, pages 280–295, Barcelona, Spain, September 2010. Springer.

[5] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012. doi: 10.1007/s10994-012-5285-8.

[6] Luis Enrique Sucar, Concha Bielza, Eduardo Morales, Pablo Hernandez-Leal, Julio Zaragoza, and Pedro Larrañaga. Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41:14–22, May 2014.

[7] Evonnildo Costa Goncalves, Angelo Plastino, and Alex Freitas. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *Institute of Electrical and*

Electronics Engineers 25th International Conference on Tools with Artificial Intelligence, pages 469–476, Nov 2013.

[8] Trevor John Hastie, Robert John Tibshirani, and Jerome Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2009. Autres impressions : 2011 (corr.), 2013 (7e corr.).

[9] Derrall Heath, Andrew Zitzelberger, and Christophe Giraud-Carrier. A multiple domain comparison of multi-label classification methods. In *Working Notes of the 2nd International Workshop on Learning from Multi-label Data at International Conference on Machine Learning/Conference on Learning Theory*, pages 21–28, 2010.

[10] Joseph Ethan Heydorn. An improved classifier chain ensemble for multi-dimensional classification with conditional dependence. Master's thesis, Brigham Young University, 2015.

[11] Zhi hua Zhou and Min ling Zhang. Multi-instance multilabel learning with application to scene classification. In *Advances in Neural Information Processing Systems*, pages 1609–1616, 2006.

[12] Tomasz Kajdanowicz and Przemyslaw Kazienko. *Flexible Query Answering Systems: 10th International Conference, Flexible Query Answering Systems, Granada, Spain, September 18-20. Proceedings*, chapter Heuristic Classifier Chains for Multi-label Classification, pages 555–566. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[13] Cheng-Xian Li. Exploiting label correlations for multi-label classification. Master's thesis, University of California, San Diego, 2011.

[14] Weiwei Liu and Ivor Tsang. On the optimality of classifier chain for multi-label classification. In Corinna Cortes, Neil Lawrence, Daniel Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 712–720. Curran Associates, Inc., 2015.

[15] Xi Liu, Zhiping Shi, Zhixin Li, Xishun Wang, and Zhongzhi Shi. Sorted label classifier chains for learning images with multi-label. In Alberto Del Bimbo, Shih-Fu Chang, and Arnold W. M. Smeulders, editors, *ACM Multimedia*, pages 951–954. ACM, 2010.

[16] Oscar Luaces, Jorge Díez, José Barranquero, Juan José del Coz, and Antonio Bahamonde. Binary relevance efficacy for multilabel classification. *Progress in AI*, 1(4):303–313, 2012.

[17] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45 (9):3084–3104, 2012.

[18] Elena Montañés, José Ramón Quevedo, and Juan José del Coz. Aggregating independent and dependent models to learn multi-label classifiers. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*, pages 484–500, 2011.

[19] Richard Glenn Morris. A hierarchical multi-output nearest neighbor model for multi-output dependence learning. Master's thesis, Brigham Young University, 2013.

[20] Jesse Read. Meka: A multi-label extension to weka. Web, 2012. URL `http://meka.sourceforge.net/`.

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

[22] Jesse Read, Concha. Bielza, and Pedro Larranaga. Multi-dimensional classification with super-classes. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1720–1733, July 2014.

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

[24] Jesse Read, Luca Martino, Pablo M. Olmos, and David Luengo. Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition*, 48(6): 2096–2109, 2015.

[25] Pablo Nascimento Silva, Eduardo Corrêa Gonçalves, Alexandre Plastino, and Alex Freitas. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery*, chapter Distinct Chains for Different Instances: An Effective Strategy for Multi-label Classifier Chains, pages 453–468. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[26] Mohammad Shahed Sorower. A literature survey on algorithms for multi-label learning. Technical report, Oregon State University, Corvallis, 2010.

[27] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine*

Learning, European Conference on Machine Learning, pages 406–417, Berlin, Heidelberg, 2007. Springer-Verlag.

[28] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[29] Julio Zaragoza, Luis Enrique Sucar, Eduardo Morales, Concha Bielza, and Pedro Larrañaga. Bayesian chain classifiers for multidimensional classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Three*, International Joint Conference on Artificial Intelligence, pages 2192–2197. AAAI Press, 2011. ISBN 978-1-57735-515-1.