Distinct Chains for Different Instances: an Effective Strategy for Multi-label Classifier Chains

Pablo Nascimento da Silva[1], Eduardo Corrêa Gonçalves[1], Alexandre Plastino[1], and Alex A. Freitas[2]

[1] Fluminense Federal University, Institute of Computing, Niteroi RJ, Brazil
{psilva,egoncalves,plastino}@ic.uff.br
[2] University of Kent, School of Computing, UK
a.a.freitas@kent.ac.uk

Abstract. Multi-label classification (MLC) is a predictive problem in which an object may be associated with multiple labels. One of the most prominent MLC methods is the classifier chains (CC). This method induces q binary classifiers, where q represents the number of labels. Each one is responsible for predicting a specific label. These q classifiers are linked in a chain, such that at classification time each classifier considers the labels predicted by the previous ones as additional information. Although the performance of CC is largely influenced by the chain ordering, the original method uses a random ordering. To cope with this problem, in this paper we propose a novel method which is capable of finding a specific and more effective chain for each new instance to be classified. Experiments have shown that the proposed method obtained, overall, higher predictive accuracies than the well-established binary relevance, CC and CC ensemble methods.

Keywords: Multi-Label Classification; Classifier Chains; Classification

1 Introduction

Multi-label classification (MLC) is the supervised learning problem of automatically assigning multiple labels to objects based on the features of these objects. An example of practical application is semantic scene classification [1], where the goal is to assign concepts to images. For instance, a photograph of the sun rising taken from a beach can be classified as belonging to the concepts "sky", "sunrise", and "ocean" at the same time. Other examples of important applications of MLC include text classification [16] (associating documents to various subjects), music categorization [17] (labeling songs with music genres or concepts) and functional genomics [3] (predicting the multiple biological functions of genes and proteins), just to name a few.

The multi-label classification problem can be formally defined as follows. Let $L = \{l_1, ..., l_q\}$ be a set of q class labels, where $q \geq 2$. Given a training set $D = \{(x_1, Y_1), (x_2, Y_2), ..., (x_N, Y_N)\}$ where each instance i is associated with a

feature vector $x_i = \{(x_1, ..., x_d)\}$ and a subset of labels $Y_i \subseteq L$, the goal in the multi-label classification task is to learn a classifier $h(X) \rightarrow Y$ from D that, given an unlabeled instance $E = (x, ?)$, is capable of predicting its labelset Y.

MLC problems tend to be more challenging than traditional single-label classification problems (SLC), where objects can be associated with only a single target class label. This mainly occurs due to the existence of label correlations in most MLC problems. For instance, in scene classification, an image labeled as "ocean" is more likely to also be associated to labels such as "ship" or "beach", since these concepts are positively correlated. Similarly, an image associated to the label "desert" is less likely to also be associated to the label "snow", as these concepts are negatively correlated. Therefore, intuitively, it is expected that MLC methods which are able to identify and model label correlations should be more accurate. A large body of recent work [2, 9, 14, 15, 20, 24, 26, 27] has primarily concentrated efforts to tackle this problem by using a wide range of different heuristics and statistical techniques.

Proposed in [14, 15], the classifier chains method (CC) is one of the simplest and most prominent of such methods. The CC method involves the training of q single-label binary classifiers $y_1, y_2, ..., y_q$ where each one will be responsible for predicting a specific label in $\{l_1, l_2, ..., l_q\}$. These q classifiers are linked in a randomly-ordered chain $\{y_1 \rightarrow y_2 \rightarrow ... \rightarrow y_q\}$, such that, at classification time, each binary classifier y_j incorporates the labels predicted by the previous $y_1, ..., y_{j-1}$ classifiers as additional information. This is accomplished using a simple trick: in the training phase, the feature vector associated to each classifier y_j is extended with the binary values of the labels $l_1, ..., l_{j-1}$. Although it employs a simple approach to deal with label dependencies, CC has proved to be one of the best methods for multi-label learning in terms of both efficiency and predictive performance, having become a recommended benchmark algorithm [12, 28].

However, there is a drawback in the CC approach even noted by its authors in [14, 15]: the fact that the label ordering is decided at random. It is intuitive that an inadequate label ordering can potentially decrease accuracy, as the first binary classifiers could frequently output wrong predictions at classification time, thus resulting in significant error propagation along the chain. However, finding an optimized label sequence is a difficult problem because of the enormous search space of $q!$ different existing label permutations. In order to cope with this issue, the authors of CC suggest combining random orders via an ensemble of classifier chains (ECC) with the expectation that the effect of poorly ordered chains in predictive accuracy could be mitigated. Recently, other variations of the CC basic approach have been proposed in the literature [6, 10, 13, 25], which are not based on ensembles. These novel techniques rely on the use of either statistical tests of correlation or heuristic search techniques (such as genetic algorithms and beam search) with the goal of finding a single sequence that leads to an improvement on the predictive accuracy of the CC model (i.e., an optimized label sequence). After being determined, this unique optimized chain should be used in the training and classification steps of the multi-label classifier chain model.

Nevertheless, none of the proposed CC variations have yet explored the idea of using a distinct label sequence for each new instance to be classified. In this concept, the aim is to construct a model which uses a specific label sequence tailored to each new instance at classification time. The main contribution of this paper is to demonstrate that this approach leads to a significant improvement in the predictive accuracy of the CC model. We propose a novel method called OOCC (One-to-One Classifier Chains) that addresses this problem by assigning a label sequence to a new instance in the test set based on the label sequences that perform well in training instances similar to the new instance, where such similar training instances are found using a conventional nearest neighbor (lazy learning) method. As a secondary contribution, this paper also aims at improving the fundamental understanding of the CC model. In this regard, we report the results of an experiment that, for the first time, investigated in depth the effect of different label sequences on the effectiveness of the CC method.

The remainder of this paper is organized as follows. Section 2 presents a brief overview on multi-label classification and discusses the original CC conceptual model, highlighting its main advantages and disadvantages. Section 3 presents an experiment that investigated the influence of the label sequence in the predictive accuracy of CC models. Section 4 is the main section of this work, where the OOCC algorithm is formalized and explained. Section 5 revises the related work. In Section 6, experimental results of OOCC and other MLC algorithms are presented. Conclusions and research directions are given in Section 7.

2 Multi-Label Classification

2.1 Evaluation Measures

Several evaluation metrics have been proposed to evaluate multi-label classifiers [12, 18, 28]. This subsection presents the ones used in this paper. In the definitions throughout the text we adopted the following notation: n is number of test instances; q is the number of labels; Y_i and Z_i represents, respectively, the actual and the predicted labelset of the i^{th} test instance.

The Exact Match (EM) measure, defined in Equation 1, assesses the proportion of instances that were fully correctly predicted in the test set. In Equation 1, $I(true) = 1$ and $I(false) = 0$.

$$EM = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Zi) \tag{1}$$

The Accuracy (ACC) and F-Measure (FM) measures, respectively defined in Equations 2 and 3, are less strict than EM, providing the user with information about the proportion of correct predictions, meanwhile taking into consideration results that are partially correct.

$$ACC = \frac{1}{n}\sum_{i=1}^{n} \frac{|Z_i \cap Y_i|}{|Z_i \cup Y_i|} \tag{2}$$

$$FM = \frac{1}{n}\sum_{i=1}^{n}\frac{2 \times |Z_i \cap Y_i|}{|Z_i| + |Y_i|} \tag{3}$$

The Hamming Loss (HL) measure, defined in Equation 4, gives the average percentage of wrong label predictions to the total number of labels. The expression $Y_i \triangle Z_i$ represents the symmetric difference between Y_i and Z_i. Since HL is a loss function, its optimal value is zero.

$$HL = \frac{1}{n}\sum_{i=1}^{n}\frac{|Z_i \triangle Y_i|}{q} \tag{4}$$

2.2 Basic Approaches for Multi-label Learning

Existing methods for MLC can be divided into two main categories: algorithm dependent or independent [18, 28]. Algorithm dependent methods extend or adapt an existing single-label algorithm for the task of MLC. E.g., in [22] the authors developed a special topology and a new inference procedure for Bayesian networks so as to allow their use in multi-label problems.

By contrast, algorithm independent methods transform the multi-label problem into one or more single-label classification (SLC) problems. Then, any existing SLC algorithm can be directly applied by simply mapping its single label predictions into multi-label predictions. This enables abstraction from the underlying base algorithm, which is an important advantage since different classifiers achieve better performance in different application domains. There are a few distinct strategies to perform the transformation, being the binary relevance (BR) [11, 15] approach the most commonly adopted. This method works by decomposing the multi-label problem into q independent single-label binary problems. In the training phase, one binary classifier is independently learned for each label. The labels of new instances are predicted by combining the outputs produced by each classifier.

The BR strategy presents some advantages: (i) it is simple and algorithm independent; (ii) it scales linearly with q; (iii) it can be easily parallelized. However, a serious disadvantage lies in that it is based on the assumption that all labels are independent. Each classifier works independently, disregarding the possible occurrence of relationships among labels. As a consequence, potentially important predictive information may be ignored.

2.3 Classifier Chains

The classifier chains model [14, 15], briefly introduced in Section 1, represents a direct extension to the BR approach which is able to exploit label correlations. As with BR, the CC method involves the training of q single-label binary classifiers $y_1, y_2, ..., y_q$ where each one will be solely and respectively responsible for predicting a specific label in $\{l_1, l_2, ..., l_q\}$. The difference is that, in CC, these q classifiers are linked in a chain $\{y_1 \rightarrow y_2 \rightarrow ... \rightarrow y_q\}$. The first binary classifier

in the chain, y_1, is trained using solely the attributes that compose the feature set X as its input attributes to predict the first label l_1. The second binary classifier, y_2, is trained using X augmented with l_1, which corresponds to the label associated to the classifier y_1. Each subsequent classifier y_j is trained using X augmented with the information of $j-1$ labels (the labels associated to the previous $j-1$ classifiers in the chain). Once the model is trained, the classification step should also be performed in a chained way. To predict the labelset of a new object, q binary classifications are needed, with the process beginning at y_1 and going along the chain. In this procedure, the classifier y_j predicts the relevance of label l_j, given the feature space augmented by the predictions carried out by the previous $j-1$ classifiers.

The CC conceptual model has many appealing properties. First, it is theoretically simple. While most MLC methods invest in complex probabilistic approaches to model label dependencies, CC adopts a quite straightforward strategy: it just passes label information between classifiers. It is also relatively efficient, since it scales linearly with q. Finally and more importantly, the method has proved to be highly effective. A comprehensive recent empirical study [12] comparing several state-of-the-art methods for MLC reported that CC is among the top best performing algorithms in terms of predictive performance. However, there is an important drawback in the basic CC approach: the label ordering is decided at random instead of being selected in a data-driven fashion. This issue is carefully investigated in the next section.

3 The Label Sequence Issue

In the basic CC model, the label sequence is decided at random. This has often been considered a major drawback, even noted by the authors of CC themselves, which deemed that if the first members of the chain have low accuracy (i.e., if they output many wrong predictions), error propagation will occur along the chain causing a significant decrease in predictive accuracy [14, 15]. In a similar vein, [10, 13] argued that different label orderings can lead to different results in terms of predictive accuracy mainly due to noisy data and finite sample effects. For example, if a label l_j is rare, then its binary model may be misestimated depending on the position of l_j in the chain. Nonetheless, the authors of [19] have a completely different belief. They consider that the effect of the chain order will often be very small when the number of features in the dataset is much higher than the number of labels (which corresponds to the most typical situation).

Nevertheless, [2] realized that "the effect of different orders on the prediction performance of the (CC) algorithm has not yet been studied in depth". Motivated by this consideration and by the conflicting views of [19] and [10, 13–15], we decided to carry out an experiment to investigate the following questions:

1. Does finding a single optimized label sequence for the entire dataset indeed significantly improve the effectiveness of CC?
2. Does finding different optimized label sequences for distinct instances improve even more the effectiveness of CC?

The experiment consisted in assessing the predictive accuracy of CC considering all $q!$ label permutations of three benchmark datasets using the following single-label base algorithms: k-NN, C4.5, Naïve Bayes, and SMO [23]. The main goal is to observe the differences in predictive accuracy between the best (most accurate) chain and the worst chain for each of those base classifiers. If most of the differences are large, then there is evidence that the label sequence is actually important. In the experiment the predictive performance is determined in terms of the Accuracy measure (a brief note on results for other measures will be mentioned later).

The experiment was implemented in Java, within the MULAN tool [21], an open source platform for the evaluation of multi-label algorithms that works on the top of the WEKA API [7]. The datasets "emotions" ($q = 6$), "scene"($q = 6$) and "flags" ($q = 7$), obtained from the MULAN repository were used in this experiment. Since they have a small number of labels, it became feasible to build and test CC models for all possible label permutations. These models were evaluated by applying the holdout method with the use of the training and test parts supplied with the datasets.

Tables 1, 2 and 3, respectively present the results for the datasets "emotions", "scene" and "flags" in terms of Accuracy. These tables are divided into four main columns. The first indicates the name of the base algorithm (the acronym "x-NN" is used to refer to the k-NN algorithm configured with $k = x$). The second main column indicates the obtained Accuracy values when a unique chain is selected for the training and testing of all instances. It is divided into sub-columns {1},{2} and {3}, which respectively present the performance of the best label sequence, the performance of the worst label sequence and the difference in the Accuracy value between the best and the worst sequences. The third main column indicates the obtained Accuracy values when different label sequences are used for different instances. It is also divided into three sub-columns, labeled as {4}, {5} and {6}. Sub-column {4} presents the Accuracy value that is obtained when the best label sequence associated with each instance is selected. Sub-column {5} presents the computed Accuracy value when the worst sequence associated with each instance is selected. Sub-column {6} simply shows the difference between the values in {4} and {5}. Finally, the fourth main column presents the improvement in performance obtained when the best chain for each instance is selected in relation to the use of a unique best chain for the entire dataset (the best chain on average). In Sub-columns {3}, {6}, and {7}, the numbers between brackets in each cell denote the rank of the corresponding difference values.

The results revealed that: (i) using a single optimized label sequence indeed has a strong effect on predictive accuracy; and (ii) finding different optimized label sequences for distinct instances is even more effective. For example, consider the performance of the C4.5 algorithm in Table 3. Note that the difference in Accuracy between the model built with the best single chain (i.e., the best chain on average considering the entire dataset) and the model built with the worst single chain reached 12.65% (Sub-column {3}). However, the difference in the predictive performance between selecting the best chain for each instance and

Table 1: Results of the exhaustive experiment in terms of Accuracy values for the *emotions* dataset.

Classifier	One chain for the dataset			One chain for each instance			Improvement
	Best {1}	Worst {2}	Diff {3}	Best {4}	Worst {5}	Diff {6}	{7}={4}−{1}
1-NN	0.4926	0.4926	0.0000 (7)	0.4926	0.4926	0.0000 (7)	0.0000 (7)
3-NN	0.5837	0.4983	0.0854 (5)	0.6885	0.3879	0.3006 (3)	0.1048 (5)
5-NN	0.5957	0.5314	0.0643 (3)	0.7227	0.3982	0.3245 (5)	0.1270 (4)
7-NN	0.6021	0.5307	0.0714 (4)	0.7244	0.3916	0.3328 (4)	0.1223 (3)
C4.5	0.5380	0.4059	0.1321 (1)	0.9059	0.0724	0.8335 (1)	0.3679 (1)
NB	0.5436	0.5184	0.0252 (6)	0.5840	0.4656	0.1184 (6)	0.0404 (6)
SMO	0.6167	0.4864	0.1303 (2)	0.7805	0.3426	0.4380 (2)	0.1638 (2)

Table 2: Results of the exhaustive experiment in terms of Accuracy values for the *scene* dataset.

Classifier	One chain for the dataset			One chain for each instance			Improvement
	Best {1}	Worst {2}	Diff {3}	Best {4}	Worst {5}	Diff {6}	{7}={4}−{1}
1-NN	0.6368	0.6368	0.0000 (7)	0.6368	0.6368	0.0000 (7)	0.0000 (7)
3-NN	0.6785	0.6575	0.0210 (5)	0.7103	0.6315	0.0787 (5)	0.0318 (5)
5-NN	0.6898	0.6522	0.0376 (3)	0.7429	0.6196	0.1233 (3)	0.0531 (3)
7-NN	0.6819	0.6487	0.0332 (4)	0.7277	0.6116	0.1161 (4)	0.0458 (4)
C4.5	0.5993	0.5376	0.0617 (2)	0.8822	0.1564	0.7259 (1)	0.2829 (1)
NB	0.4415	0.4358	0.0057 (6)	0.4473	0.4309	0.0164 (6)	0.0058 (6)
SMO	0.6915	0.6069	0.0846 (1)	0.9034	0.3537	0.5497 (2)	0.2119 (2)

selecting the worst chain for each instance is 54.24% (Sub-column {6}). More interestingly, note that the choice of the most accurate chain for each instance lead to an Accuracy value 18.67% higher than that achieved by the one obtained by the model built with the best single chain for the entire dataset (Sub-column {7}). The same characteristic can be also observed for the two other datasets (Tables 1 and 2) and nearly all base algorithms evaluated in the experiments, with the exception of 1-NN (for which there is no improvement in Sub-column {7} across Tables 1, 2 and 3).

Additionally, it is also evident that the different base (single-label) algorithms, due to their own characteristics, are affected to different degrees by the label ordering. The effect tends to be very large when the base algorithms are C.45 and SMO, but it can be rather small for Naïve Bayes. The k-NN presented large differences for some configurations of k and small differences for others.

We also ran the same experiment using the measures of Exact Match and Hamming Loss and the results were similar: they evidenced that building a model which uses a specific and more effective label sequence for each new instance at classification time can largely improve the predictive performance of CC. Motivated by this empirical finding, in the next section we propose a novel

Table 3: Results of the exhaustive experiment in terms of Accuracy values for the *flags* dataset.

Classifier	One chain for the dataset			One chain for each instance			Improvement
	Best {1}	Worst {2}	Diff {3}	Best {4}	Worst {5}	Diff {6}	{7}={4}−{1}
1-NN	0.5305	0.5305	0.0000 (7)	0.5305	0.5305	0.0000 (7)	0.0000 (7)
3-NN	0.6223	0.5159	0.1064 (5)	0.6964	0.4154	0.2810 (2)	0.0741 (5)
5-NN	0.6277	0.5343	0.0934 (4)	0.7487	0.4225	0.3262 (4)	0.1210 (4)
7-NN	0.6143	0.5102	0.1041 (3)	0.7394	0.4104	0.3290 (3)	0.1251 (3)
C4.5	0.6222	0.4957	0.1265 (1)	0.8089	0.2665	0.5424 (1)	0.1867 (1)
NB	0.5759	0.4873	0.0886 (6)	0.6291	0.4179	0.2112 (5)	0.0532 (6)
SMO	0.6068	0.5220	0.0848 (2)	0.7712	0.3751	0.3962 (6)	0.1644 (2)

method that addresses this problem by assigning a label sequence to a new instance based on the label sequences that perform well in the training instances that are most similar to the new instance being classified.

4 One-to-One Classifier Chains (OOCC)

In this section we present a novel method called One-To-One Classifier Chains (OOCC), which assigns a label sequence to each new instance t in the test set based on the label sequences that perform well in training instances similar to t. The basic ideas of our OOCC method are as follows. First, we find the one or more label sequences that perform well for each training instance (see Subsection 4.1). Then we use a k-NN (k-nearest neighbors) algorithm to retrieve the k training instances that are most similar to the instance t being classified, and assign, to t, the label sequence that was found to perform best for the training instances. Due to the similarity between testing instance t and its nearest training instances, it is expected that an effective label sequence for instance t's nearest neighbors will also be an effective label sequence for instance t.

In order to measure the predictive accuracy associated with each candidate label sequence for a given training instance, we compute the quality function in Equation 5. This function (originally proposed in [6]) determines the quality of prediction performed by a CC model with regard to the instance t, by taking into account the measures of Exact Match, Accuracy and Hamming Loss.

$$Quality(t, CC) = \frac{(1 - HL) + ACC + EM}{3} \tag{5}$$

The OOCC method modifies both the training and the classification steps of the original CC method. These changes are explained in the next subsections.

4.1 OOCC's Training Procedure

Algorithm 1 describes the algorithm used in the OOCC's training step. This algorithm produces as output an array named *bestChains*, which is responsible

Algorithm 1 OOCC's training procedure

Input : D (training set), m (number of data partitions), r (number of label sequences)
Output: $bestChains$ (an array containing the best chains for each training instance)

```
 1: Divide the training set D into m folds {D_1, D_2, ..., D_m}
 2: bestChains → new Array(N)
 3: for all folds D_v ∈ D do
 4:    CCModels ← ∅
 5:    D_st ← {D − D_v}, where D_v = validation set, D_st = sub-training set
 6:    RS ← generateRandomSequences(r)
 7:    for all label sequences l_s ∈ RS do
 8:       CC ← buildCC(D_st, l_s)
 9:       CCModels ← CCModels ∪ CC
10:    end for
11:    for all instances I ∈ D_v do
12:       bestQuality ← −1
13:       for all classifier chain models CC ∈ CCModels do
14:          l_s ← label sequence associated to the model CC
15:          curQuality ← Quality(I, CC)
16:          if curQuality > bestQuality then
17:             bestQuality ← curQuality
18:             bestChains(I) ← {l_s, curQuality}
19:          else if curQuality = bestQuality then
20:             bestChains(I) ← bestChains(I) ∪ {l_s, curQuality}
21:          end if
22:       end for
23:    end for
24: end for
25: return bestChains
```

for storing the best label sequence(s) associated to each training instance at the end of processing. First, the training dataset is partitioned into m distinct subsets (line 1), where m is a user-provided parameter. Each of the m subsets represents a different validation set (denoted as D_v within the algorithm specification). These validation sets are processed in turn in the FOR loop that encompasses lines 3 to 24. This FOR loop is divided into two phases: building CC models with random chains (lines 4 to 10) and selection of the best label sequences for each instance (lines 11 to 23).

The first phase works as follows. During each iteration, the data partition D_v is assigned r distinct random label sequences, where r is specified by the user. Next (lines 7 to 10), r CC models are induced, one for each distinct sequence, using a sub-training set D_{st} formed by the remainder $m-1$ data partitions (i.e., all data partitions except D_v). These models are stored in the array $CCModels$.

Once the models are built, it becomes possible to identify the best label sequences associated to each instance I of the data partition D_v. This is done in the second phase of the OOCC's training procedure. In this phase, all r trained

Algorithm 2 OOCC's classification procedure

Input : D (training set), t (instance to be classified), k (number of neighbors)
Output: Z (the predicted labelset for instance t)

```
1: NN ← find the k closest neighbors to t in D.
2: S ← ∅
3: bestQuality ← −1
4: for all neighbors I ∈ NN do
5:    chains ← label sequences stored in bestChains(I)
6:    curQuality ← Quality of the label sequences stored in bestChains(I)
7:    if curQuality > bestQuality then
8:       bestQuality ← curQuality
9:       S ← chains
10:   else if curQuality = bestQuality then
11:      S ← S ∪ chains
12:   end if
13: end for
14: if S.size = 1 then
15:   l_s ← the label sequence stored in S
16: else
17:   l_s ← randomly-choose a label sequence from S
18: end if
19: CC ← buildCC(D, l_s)
20: Z ← classify(t,CC)
21: return Z
```

models contained in $CCModels$ are used to evaluate the Quality of each instance I from D_v with the use of the function defined in Equation 5 (lines 14-15). The label sequence which achieves the highest value of Quality for an instance I must be stored in the output array $bestChains$, along with their associated Quality value (lines 16-21). Since for some instances, more than one label sequence may achieve the same best value of Quality, it is possible to store more than one label sequence for I.

4.2 OOCC's Classification Procedure

The OOCC method employs a lazy procedure to classify a new test instance t, which is described in the algorithm shown in Algorithm 2. This procedure can be divided into three phases which are explained below.

Phase 1 (line 1) consists in finding the k instances more similar to t in the training set, where k is a user-specified parameter. In Phase 2 (lines 3 to 13), the algorithm examines the best label sequences associated to each neighbor instance (which were found in the OOCC's training step and are stored in the $bestChains$ set). At the end of this phase, the highest-quality sequence(s) will be stored in the S set. Phase 3 (lines 14 to 21) actually performs the classification of t. First, an optimized label sequence l_s is selected from S. A CC model is induced using the training set D and l_s. This model is then used to classify t.

5 Related Work

The authors of the original CC method were the first to propose a method to address the label ordering issue. They suggested the use of an ensemble of classifier chains (ECC) [14, 15] in order to cope with that issue. In this approach the individual classifiers vote and the output labelset for a new instance is determined based on the collection of votes.

The techniques proposed in [6, 10, 13, 25] are based on the search for a single optimized label sequence rather than using an ensemble approach. In [25], the authors present the Bayesian Chain Classifier (BCC) algorithm. In this approach, the first step is to induce a maximum weighted spanning tree, according to the mutual dependence measure between each pair of labels. Then, different optimized sequences may be generated according to the selection of a distinct node as the root node. The algorithm presented in [10] tackles the label sequence optimization problem by performing a beam search over a tree in which every distinct path represents a different label permutation. Since the construction of a tree with $q!$ paths is infeasible even for moderate sizes of q, the algorithm employs a user adjustable input parameter b (beam width) to reduce the number of paths (at each level, only the top-b vertices in terms of predictive accuracy are maintained in the tree). The M2CC algorithm, described in [13], employs a double-Monte Carlo optimization technique to efficiently generate and evaluate a small population of distinct label sequences. The algorithm starts with a randomly chosen sequence, s_0. During the algorithm execution this sequence is modified with the aim of finding, at least, a local maximum of some payoff function (e.g.: Exact Match). Finally, the work of [6] proposes GACC – a genetic algorithm to solve the label sequence optimization problem. In this strategy, each chromosome represents a different label sequence and the fitness function is the same defined in Equation 5. The crossover operation works by transferring subchains of random length between pairs of individuals. The proposed GA follows the wrapper approach [5], evaluating the quality of an individual (candidate label sequence) by using the target MLC algorithm (i.e. the CC algorithm). All these proposals aim at finding a unique label sequence that is used to train a CC model for all instances in the training dataset. Differently, the OOCC method proposed in this work is capable of selecting a distinct and more effective chain for each instance of the training dataset.

The PCC algorithm, introduced in [4], represents a technique to improve CC through the use of inference optimization. The PCC's training step is identical to the CC's one: a label sequence is randomly chosen and used to train a CC model. However, its classification step works differently. According to the label sequence used in the training step, the PCC classifier aims at maximizing the posterior probability of the predicted labelset for each test instance. However, differently of our approach, the PCC requires a probabilistic single-label base classifier. Moreover, it has the disadvantage of employing an exhaustive search in the space of 2^q possible label combinations. Thus, its practical applications are restricted to problems where q is small.

6 Experiments

We implemented our OOCC method within the MULAN platform [21]. The proposed method was evaluated on nine distinct benchmark datasets, which were obtained from the Mulan repository. A holdout evaluation was performed to assess the predictive performance of the multi-label methods, by using the training and test parts that come with these datasets. We compared OOCC to the algorithms BR, CC and ECC. The WEKA's SMO implementation with default parameters was used as the base single-label classification algorithm for all evaluated methods, although other algorithms could have been used. The parameter values used in OOCC were $k = 5$, $m = 5$ and $r = 15$. For ECC, the number of members in the ensemble was set to 10.

The predictive performance of the algorithms was evaluated in terms of Accuracy, F-Measure, Hamming Loss and Exact Match. To determine whether the differences in performance for each measure are statistically significant, we ran the Friedman test and the Nemenyi post-hoc test, following the approach described in [8]. First, the Friedman test is executed with the null hypothesis that the performances of all methods are equivalent. Whenever the null hypothesis is rejected at the 95% confidence level, we ran the Nemenyi post-hoc multiple comparison test, which assesses if there is a statistically significant difference in the performances of each pair of methods.

The results for the measures of Accuracy, F-Measure, Hamming Loss and Exact Match are respectively presented in Tables 4, 5, 6 and 7. In these tables, N, d and q represent, respectively, the number of instances, attributes and labels for each dataset. The best results for each dataset are highlighted in bold type. The obtained rank for each method in each dataset is presented in parenthesis. In the lines right below Tables 4, 5 and 6, the symbol $\succ$ represents a significant difference between one or more methods, such that $\{a\} \succ \{b, c\}$ shows that the method a is significantly better than b and c.

Table 4: Performance of BR, CC, ECC and OOCC in terms of Accuracy.

Dataset (N, d, q)	Accuracy			
	BR	CC	ECC	OOCC
flags (194, 19, 7)	**0.5938 (1.0)**	0.5560 (4.0)	0.5748 (2.5)	0.5748 (2.5)
cal500 (502, 68, 174)	0.2017 (2.0)	0.1765 (4.0)	0.2007 (3.0)	**0.2113 (1.0)**
emotions (593, 72, 6)	0.4835 (4.0)	0.5202 (3.0)	0.5653 (2.0)	**0.5866 (1.0)**
birds (645, 300, 19)	0.5669 (3.0)	0.5623 (4.0)	**0.5682 (1.0)**	0.5672 (2.0)
genbase (662, 1186, 27)	**0.9908 (2.5)**	**0.9908 (2.5)**	**0.9908 (2.5)**	**0.9908 (2.5)**
medical (978, 1449, 45)	0.6990 (4.0)	0.7134 (3.0)	0.7161 (2.0)	**0.7220 (1.0)**
enron (1702, 1001, 53)	0.4063 (3.0)	0.4053 (4.0)	**0.4501 (1.0)**	0.4129 (2.0)
scene (2407, 294, 6)	0.5711 (4.0)	0.6598 (3.0)	0.6654 (2.0)	**0.6702 (1.0)**
yeast (2417, 103, 14)	0.5018 (3.0)	0.4892 (4.0)	0.5333 (2.0)	**0.5429 (1.0)**
rank sums	26.5	31.5	18.0	**14.0**

$\{OOCC\} \succ \{BR, CC, ECC\}, \{ECC\} \succ \{BR, CC\}, \{BR\} \succ \{CC\}$

Table 5: Performance of BR, CC, ECC and OOCC in terms of F-Measure.

Dataset (N, d, q)	F-Measure			
	BR	CC	ECC	OOCC
flags (194, 19, 7)	**0.7139 (1.0)**	0.6764 (4.0)	0.7020 (2.0)	0.6927 (3.0)
cal500 (502, 68, 174)	0.3297 (2.0)	0.2919 (4.0)	0.3251 (3.0)	**0.3375 (1.0)**
emotions (593, 72, 6)	0.5556 (4.0)	0.5979 (3.0)	0.6429 (2.0)	**0.6627 (1.0)**
birds (645, 300, 19)	**0.6061 (1.0)**	0.5976 (4.0)	0.6039 (2.5)	0.6039 (2.5)
genbase (662, 1186, 27)	**0.9940 (2.5)**	**0.9940 (2.5)**	**0.9940 (2.5)**	**0.9940 (2.5)**
medical (978, 1449, 45)	0.7273 (4.0)	0.7409 (3.0)	0.7436 (2.0)	**0.7472 (1.0)**
enron (1702, 1001, 53)	0.5152 (4.0)	0.5110 (3.0)	**0.5575 (1.0)**	0.5197 (2.0)
scene (2407, 294, 6)	0.5985 (4.0)	0.6761 (3.0)	0.6870 (2.0)	**0.6883 (1.0)**
yeast (2417, 103, 14)	0.6101 (3.0)	0.5904 (4.0)	0.6361 (2.0)	**0.6436 (1.0)**
rank sums	25.5	30.5	19.0	**15.0**

$\{OOCC\} \succ \{BR, CC, ECC\}, \{ECC\} \succ \{BR, CC\}, \{BR\} \succ \{CC\}$

The results presented in Tables 4 and 5 show that the OOCC performance is, in the majority of the datasets, superior to all other methods with respect to the Accuracy and F-Measure metrics. Note that the obtained rank sums are always smaller (indicating a better result) for the OOCC method. Actually, the Friedman test reported a significant difference between the methods. The Nemenyi post-hoc test indicated that OOCC is significantly better than CC, BR and ECC for both Accuracy and F-Measure, at the 95% confidence level.

The results in Table 6 indicate that ECC obtained the best results in terms of Hamming Loss, being significantly superior to all other methods. For this measure, the OOCC method performed fairly well, as it is significantly better than CC and equivalent to BR. Finally, Table 7 presents the results regarding the Exact Match measure. Although the Friedman and Nemenyi tests indicated that no statistically significant differences exist between the Exact Match values achieved by the four methods, it is possible to observe that the OOCC algorithm

Table 6: Performance of BR, CC, ECC and OOCC in terms of Hamming Loss.

Dataset (N, d, q)	Hamming Loss			
	BR	CC	ECC	OOCC
flags (194, 19, 7)	**0.2637 (1.0)**	0.3011 (4.0)	0.2813 (2.0)	0.2835 (3.0)
cal500 (502, 68, 174)	**0.1375 (1.0)**	0.1527 (3.0)	0.1458 (2.0)	0.1635 (4.0)
emotions (593, 72, 6)	0.2145 (3.0)	0.2376 (4.0)	0.2137 (2.0)	**0.2063 (1.0)**
birds (645, 300, 19)	0.0658 (3.0)	0.0668 (4.0)	**0.0595 (1.0)**	0.0619 (2.0)
genbase (662, 1186, 27)	**0.0007 (2.5)**	**0.0007 (2.5)**	**0.0007 (2.5)**	**0.0007 (2.5)**
medical (978, 1449, 45)	0.0117 (4.0)	0.0115 (3.0)	**0.0111 (1.5)**	**0.0111 (1.5)**
enron (1702, 1001, 53)	0.0572 (2.0)	0.0585 (4.0)	**0.0512 (1.0)**	0.0573 (3.0)
scene (2407, 294, 6)	0.1144 (3.0)	0.1154 (4.0)	**0.1026 (1.0)**	0.1116 (2.0)
yeast (2417, 103, 14)	**0.1997 (1.0)**	0.2109 (4.0)	0.2024 (3.0)	0.2014 (2.0)
rank sums	20.5	32.5	**16.0**	21.0

$\{ECC\} \succ \{BR, CC, OOCC\}, \{OOCC\} \succ \{CC\}, \{BR\} \succ \{CC\}$

has the smallest rank sum (i.e., the best overall result), having obtained the best results for five of the nine evaluated datasets.

Our empirical results indicate that the proposed OOCC method exhibits a very competitive performance, obtaining results significantly superior to the other evaluated methods, according to two of the four evaluated measures of predictive accuracy. It is also worth noting that the original CC method performed rather poorly in terms of Accuracy, F-Measure and Hamming Loss, presenting a performance significantly inferior to ECC, OOCC and even to the baseline BR method (the CC method performed better than BR only in terms of Exact Match, however without statistical significance). This confirms that the use of a single randomly-generated label sequence seems to be an ineffective approach for multi-label chain classifiers, reinforcing the importance of either using an ensemble or searching for an optimized label sequence.

Table 7: Performance of BR, CC, ECC and OOCC in terms of Exact Match.

Dataset (N, d, q)	Exact Match			
	BR	CC	ECC	OOCC
flags (194, 19, 7)	**0.1538 (1.5)**	0.1231 (3.5)	0.1231 (3.5)	**0.1538 (1.5)**
cal500 (502, 68, 174)	0.0000 (2.5)	0.0000 (2.5)	0.0000 (2.5)	0.0000 (2.5)
emotions (593, 72, 6)	0.2525 (4.0)	0.2822 (3.0)	0.3267 (2.0)	**0.3465 (1.0)**
birds (645, 300, 19)	0.4630 (3.5)	**0.4722 (1.5)**	**0.4722 (1.5)**	0.4630 (3.5)
genbase (662, 1186, 27)	**0.9799 (2.5)**	**0.9799 (2.5)**	**0.9799 (2.5)**	**0.9799 (2.5)**
medical (978, 1449, 45)	0.6140 (4.0)	0.6326 (3.0)	0.6357 (2.0)	**0.6465 (1.0)**
enron (1702, 1001, 53)	0.1209 (4.0)	0.1313 (2.0)	**0.1503 (1.0)**	0.1295 (3.0)
scene (2407, 294, 6)	0.4908 (4.0)	0.6112 (2.0)	0.6012 (3.0)	**0.6162 (1.0)**
yeast (2417, 103, 14)	0.1603 (4.0)	0.1952 (3.0)	0.2148 (2.0)	**0.2399 (1.0)**
rank sums	30.0	23.0	20.0	**17.0**

No significance differences according to the Friedman test

7 Conclusions and Future Work

The classifier chains approach has become one of the most influential methods for multi-label classification. It is distinguished from other methods by its simple and effective approach to exploit label dependencies. However, the basic CC model suffers from an important drawback: it decides the label sequence at random. The main contribution of this paper was the proposal of a novel multi-label classifier chain method called One-to-One Classifier Chains (OOCC), which is capable of finding, at classification time, a specific and more accurate label sequence for each new instance in the test set. The OOCC method was compared against the well-established BR, CC and ECC methods. The obtained results show that OOCC significantly outperformed all these three methods in terms of Accuracy and F-Measure. In terms of Hamming Loss, OOCC significantly outperformed CC and was significantly outperformed by ECC. There was no significant difference among the four methods in terms of the Exact Match measure.

Additionally, we contributed to a better understanding of the underlying principles of the CC method by reporting the results of a study that evidenced that: (i) finding a single optimized label sequence has a strong effect on predictive accuracy; (ii) finding different optimized label sequences for distinct instances is even more effective; and (iii) the different base (single-label) algorithms, due to their own characteristics, are affected to different degrees by the label ordering.

For future research, we first intend to perform a detailed analysis on the sensivity of the results to the parameters r, m and k. The main goal is to determine the best set of parameters for the OOCC algorithm, using the training set to optimize the parameters. We also intend to evaluate other approaches to determine the label sequence of a new instance to be classified (which may not be necessarily based on the Quality measure). Finally, we plan to compare OOCC against some of the methods described in Section 5 and to develop an ensemble version of the proposed OOCC method.

Acknowledgments. This work was supported by CAPES research grant BEX 1642/14-6 (Eduardo Corrêa Gonçalves), CNPq and FAPERJ research grants (Alexandre Plastino) and CAPES DS scholarship (Pablo Nascimento da Silva).

References

1. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning Multi-Label Scene Classification. Pattern Recognition, 37(9), 1757–1771 (2004).
2. Cherman, E.A., Metz, J., Monard, M.C.: Incorporating Label Dependency into the Binary Relevance Framework for Multi-label Classification. Expert Systems with Applications, 39(2), 1647–1655 (2012).
3. Clare, A., King, R.: Knowledge Discovery in Multi-label Phenotype Data. In: 5th European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD 2001), pp. 42-53, Freiburg, 2001.
4. Dembczynski, K., Cheng, W., Hüllermeier, E.: Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In: 27th Intl. Conf. on Machine Learning (ICML'10), pp. 279–286, Haifa, 2010.
5. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Natural Computing Series, Springer (2002).
6. Gonçalves, E.C., Plastino, A., Freitas, A.A.: A Genetic Algorithm for Optimizing the Label Ordering in Multi-Label Classifier Chains. In: IEEE 25th Intl. Conf. on Tools with Artificial Intelligence (ICTAI'13). pp. 469–476, Herndon, 2013.
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H.: The WEKA Data Mining Software: an Update. ACM SIGKDD Exploration Newsletter, 11(1), 10–18, (2009).
8. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press (2011).
9. Li, N. and Zhou, Z.-H.: Selective ensemble of classifier chains. In: Proceedings of the 11th International Workshop on Multiple Classifier Systems (MCS'13). pp. 146–156, Nanjing, 2013.
10. Kumar, A., Vembu, S., Menon, A.K., Elkan, C.: Beam Search Algorithms for Multilabel Learning. Machine Learning, 92(1), 65–89 (2013).

11. Luaces, O., Díez, J., Barranquero, J., Coz, J. J., Bahamonde, A.: Binary Relevance Efficacy for Multilabel Classification. Progress in Artificial Intelligence, 1(4), Springer-Verlag, 303–313 (2012).
12. Madjarov, G., Kocev, D., Gjorgjevikj, D. Džeroski, S.: An Extensive Experimental Comparison of Methods for Multi-label Learning, Pattern Recognition, 45(9), 3084–3104 (2012).
13. Read, J., Martino, L., Luengo, D.: Efficient Monte Carlo Methods for Multi-dimensional Learning with Classifier Chains. Pattern Recognition, 47(3), 1535–1546 (2014).
14. Read, J., Pfahringer, B., Holmes, G. Frank, E.: Classifier Chains for Multi-label Classification. In: 20th European Conf. on Machine Learning (ECML 2009), pp. 254–269, Bled, 2009.
15. Read, J., Pfahringer, B., Holmes, G. Frank, E.: Classifier Chains for Multi-label Classification. Machine Learning, 85(3), 333–359 (2011)
16. Schapire, R.E., Singer, Y.: BoosTexter: A Boosting-based System for Text Categorization. Machine Learning, 39(2-3), 135–168 (2000).
17. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.P.: Multi-Label Classification of Music into Emotions. In: 9th Intl. Conf. on Music Information Retrieval (ISMIR'08), pp. 325–330. Philadelphia, 2008.
18. Tsoumakas, G., Katakis I., Vlahavas, I.: Mining Multi-Label Data. Data Mining and Knowledge Discovery Handbook, pp. 667–685, Springer US, 2010.
19. Tenenboim-Chekina, L., Rokach, L., Shapira, B.: Identification of Label Dependencies for Multi-label Classification. In: 2nd Intl. Workshop on Learning from Multi-Label Data (MLD'10), pp. 53–60. Haifa, 2010
20. Tsoumakas, G., Vlahavas, I.: Random k-Labelsets: An Ensemble Method for Multilabel Classification. In: 18th European Conf. on Machine Learning (ECML'07), pp. 406–417. Warsaw, 2007.
21. Tsoumakas, G., Xioufis, E.S., Vilcek, J., Vlahavas, I.P.: MULAN: A Java Library for Multi-Label Learning. JMLR, 12, 2411–2414 (2011).
22. van der Gaag, L., de Waal, P.R,.: Multi-dimensional Bayesian Network Classifiers, In: 3rd European Workshop on Probabilistic Graphical Models (PGM'06) pp. 107–114, Prague, 2006.
23. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques, 3rd ed., Elsevier Science, (2011).
24. Yua, Y., Pedryczb, W., Miao, D.: Multi-label Classification by Exploiting Label Correlations. Expert Systems with Applications, 41(6), 2989–3004 (2014).
25. Zaragoza, J.H., Sucar, L.E., Morales, E.F., Bielza, C., Larrañaga, P.: Bayesian Chain Classifiers for Multidimensional Classification. In: 22nd Intl. Joint Conf. on Artificial Intelligence (IJCAI'11), pp. 2192–2197, Barcelona, 2011.
26. Zhang, M.-L, Zhang, K.: Multi-label Learning by Exploiting Label Dependency. In: 16th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD'10), pp. 999–1008. Washington D.C., 2010.
27. Zhang, M.-L., Zhou, Z.-H.: ML-KNN: A Lazy Learning Approach to Multi-label Learning. Pattern Recognition, 40(7), 2038–2048 (2007).
28. Zhang, M.-L., Zhou, Z.-H.: A Review On Multi-Label Learning Algorithms. IEEE Transactions on Knowledge and Data Engineering, 99(PrePrints) (2013).