On the Problem of Error Propagation in Classifier Chains for Multi-Label Classification*

Robin Senge, Juan José del Coz and Eyke Hüllermeier

Draft version of a paper to appear in:

L. Schmidt-Thieme and M. Spiliopoulou (eds.) Data Analysis, Machine Learning and Knowledge Discovery. Proc. GFKL–2012, 36th Annual Conference of the German Classification Society. Springer, 2013

Abstract

So-called classifier chains have recently been proposed as an appealing method for tackling the multi-label classification task. In this paper, we analyze the influence of a potential pitfall of the learning process, namely the discrepancy between the feature spaces used in training and testing: While true class labels are used as supplementary attributes for training the binary models along the chain, the same models need to rely on estimations of these labels when making a prediction. We provide first experimental results suggesting that the attribute noise thus created can affect the overall prediction performance of a classifier chain.

1 Introduction

Multi-label classification (MLC) has attracted increasing attention in the machine learning community during the past few years [4]. The goal in MLC is to induce a model that assigns a *subset* of labels to each example, rather than a single one as in multi-class classification. For instance, in a news website, a multi-label classifier can automatically attach several labels—usually called tags in this context—to every article; the tags can be helpful for searching related news or for briefly informing users about their content.

*This research has been supported by the Germany Research Foundation (DFG) and the Spanish Ministerio de Ciencia e Innovación (MICINN) under grant TIN2011-23558.

Current research on MLC is largely driven by the idea that optimal prediction performance can only be achieved by modeling and exploiting *statistical dependencies* between labels. Roughly speaking, if the relevance of one label may depend on the relevance of others, then labels should be predicted *simultaneously* and not *separately*. This is the main argument against simple *decomposition techniques* such as binary relevance (BR) learning, which splits the original multi-label task into several independent binary classification problems, one for each label.

Until now, several methods for capturing label dependence have been proposed in the literature, including a method called *classifier chains* (CC) [3]. This method enjoys great popularity, even though it has been introduced only lately. As its name suggests, CC selects an order on the label set—a *chain* of labels—and trains a binary classifier for each label in this order. The difference with respect to BR is that the feature space used to induce each classifier is extended by the previous labels in the chain. These labels are treated as additional attributes, with the goal to model conditional dependence between a label and its predecessors. CC performs particularly well when being used in an ensemble framework, usually denoted as *ensemble of classifier chains* (ECC), which reduces the influence of the label order.

Our study aims at gaining a deeper understanding of CC's learning process. More specifically, we address a potential pitfall of this method: Since information about preceding labels is only available for training, this information has to be replaced by estimations (coming from the corresponding classifiers) at prediction time. As a result, CC has to deal with a specific type of attribute noise: While a classifier is learned on "clean" training data, including the true values of preceding labels, it is applied on "noisy" test data, in which true labels are replaced by possibly incorrect predictions. Obviously, this type of noise may affect the performance of each classifier in the chain. More importantly, since each classifier relies on its predecessors, a single false prediction might be propagated and possibly even reinforced along the chain.

The rest of the paper is organized as follows. The next section introduces the setting of MLC, and Section 3 explains the classifier chains method. Section 4 is devoted to a deeper discussion of the aforementioned pitfalls of CC, along with some experiments for illustration purposes. The paper ends with a couple of concluding remarks in Section 5.

2 Multi-Label Classification

Let $\mathcal{L} = \{\lambda_1, \lambda_2, \dots, \lambda_m\}$ be a finite and non-empty set of class labels, and let $\mathcal{X}$ be an instance space. We consider an MLC task with a training set $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, generated independently and identically according to a probability distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$ on $\mathcal{X} \times \mathcal{Y}$. Here, $\mathcal{Y}$ is the set of possible label combinations, i.e., the power set of $\mathcal{L}$. To ease notation, we define $\boldsymbol{y}_i$ as a binary vector $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,m})$, in which $y_{i,j} = 1$ indicates the presence (relevance) and $y_{i,j} = 0$ the absence (irrelevance) of λ_j in the labeling of $\boldsymbol{x}_i$. Under this convention, the output space is given by $\mathcal{Y} = \{0,1\}^m$. The goal in MLC is to induce from S a hypothesis $\boldsymbol{h} : \mathcal{X} \longrightarrow \mathcal{Y}$ that correctly predicts the subset of relevant labels for unlabeled query instances $\boldsymbol{x}$.

The most-straight forward and arguably simplest approach to tackle the MLC problem is *binary relevance* (BR). The BR method reduces a given multi-label problem with m labels to m *binary classification* problems. More precisely, m hypotheses $h_j : \mathcal{X} \longrightarrow \{0,1\}$, $j = 1, \ldots, m$, are induced, each of them being responsible for predicting the relevance of one label, using $\mathcal{X}$ as an input space. In this way, the labels are predicted independently of each other and no label dependencies are taken into account.

In spite of its simplicity and the strong assumption of label independence, it has been shown theoretically and empirically that BR performs quite strong in terms of decomposable loss functions [1], including the well-known *Hamming loss* $L_H(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{1}{m}\sum_{i=1}^{m} [\![y_i \neq h_i(\boldsymbol{x})]\!]$. The Hamming loss averages the standard 0/1 classification error over the m labels and hence corresponds to the proportion of labels whose relevance is incorrectly predicted. Thus, if one of the labels is predicted incorrectly, this accounts for an error of $\frac{1}{m}$. Another extension of the standard 0/1 classification loss is the *subset 0/1 loss* $L_{ZO}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = [\![\boldsymbol{y} \neq \boldsymbol{h}(\boldsymbol{x})]\!]$. Obviously, this measure is more drastic and already treats a mistake on a single label as a complete failure. The necessity to exploit label dependencies in order to minimize the generalization error in terms of the subset 0/1 loss has been shown in [1].

3 Classifier Chains

While following a similar setup as BR, classifier chains (CC) seek to capture label dependencies. CC learns m binary classifiers linked along a chain, where each classifier deals with the binary relevance problem associated with one label. In the training phase, the feature space of each classifier in the chain is extended with the actual label information of all previous labels in the chain. For instance, if the chain follows the order $\lambda_1 \rightarrow \lambda_2 \rightarrow \ldots \rightarrow \lambda_m$, then the classifier h_j responsible for predicting the relevance of λ_j is of the form

$$h_j : \mathcal{X} \times \{0,1\}^{j-1} \longrightarrow \{0,1\} \ . \tag{1}$$

The training data for this classifier consists of instances $(\boldsymbol{x}_i, y_{i,1}, \ldots, y_{i,j-1})$ labeled with $y_{i,j}$, that is, original training instances $\boldsymbol{x}_i$ supplemented by the relevance of the labels $\lambda_1, \ldots, \lambda_{j-1}$ preceding λ_j in the chain.

At prediction time, when a new instance $\boldsymbol{x}$ needs to be labeled, a label subset $\boldsymbol{y} = (y_1, \ldots, y_m)$ is produced by successively querying each classifier h_j. Note, however, that the inputs of these classifiers are not well-defined, since the supplementary attributes $y_{i,1}, \ldots, y_{i,j-1}$ are not available. These missing values are therefore replaced by their respective predictions: y_1 used by h_2 as an additional input is replaced by $\hat{y}_1 = h_1(\boldsymbol{x})$, y_2 used by h_3 as an additional input is replaced by $\hat{y}_2 = h_2(\boldsymbol{x}, \hat{y}_1)$, and so forth. Thus, the prediction $\boldsymbol{y}$ is of the form

$$\boldsymbol{y} = \big(h_1(\boldsymbol{x}),\, h_2(\boldsymbol{x}, h_1(\boldsymbol{x})),\, h_3(\boldsymbol{x}, h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, h_1(\boldsymbol{x}))), \ldots \big)$$

Realizing that the order of labels in the chain may influence the performance of the classifier, and that an optimal order is hard to anticipate, the authors in [3] propose the use of an ensemble of CC classifiers. This approach combines the predictions of different random orders and, moreover, uses a different sample of the training data to train each member of the ensemble. *Ensembles of classifier chains* (ECC) have been shown to increase prediction performance over CC by effectively using a simple voting scheme to aggregate predicted relevance sets of the individual CCs: For each label λ_j, the proportion $\hat{w}_j$ of classifiers predicting $y_j = 1$ is calculated. Relevance of λ_j is then predicted by using a threshold t, that is, $\hat{y}_j = [\![\hat{w}_j \geq t]\!]$.

4 The Problem of Error Propagation in CC

The learning process of CC violates a key assumption of machine learning, namely that the training data is representative of the test data in the sense of being identically distributed. This assumption does not hold for the chained classifiers in CC: While using the *true* label data y_j as input attributes during the training phase, this information is replaced by *estimations* $\hat{y}_j$ at prediction time. Needless to say, y_j and $\hat{y}_j$ will normally not follow the same distribution.

From the point of view of the classifier h_j, which uses the labels $y_1, \ldots, y_{j-1}$ as additional attributes, this problem can be seen as a problem of *attribute noise*. More specifically, we are facing the "clean training data vs. noisy test data" case, which is one of four possible noise scenarios that have been studied quite extensively in [5]. For CC, this problem appears to be vital: Could it be that the additional label information, which is exactly what CC seeks to exploit in order to gain in performance (compared to BR), eventually turn out to be a source of impairment? Or, stated differently, could the additional label information perhaps be harmful rather than useful? This question is difficult to answer in general. In particular, there are several factors involved, notably the following:

- *The length of the chain*: The larger the number $j-1$ of preceding classifiers in the chain, the higher is the potential level of attribute noise for a classifier h_j. For example, if prediction errors occur independently of each other with probability ϵ, then the probability of a noise-free input is only $(1-\epsilon)^{j-1}$. More realistically, one may assume that the probability of a mistake is not constant but will increase with the level of attribute noise in the input. Then, due to the recursive structure of CC, the probability of a mistake will increase even more rapidly along the chain.

- *The order of the chain*: Since some labels might be inherently more difficult to predict than others, the order of the chain will play a role, too. In particular, it would be advantageous to put simpler labels in the beginning and harder ones more toward the end of the chain.

- *The accuracy of the binary classifiers*: The level of attribute noise is in direct correspondence with the accuracy of the binary classifiers along the chain. More specifically, these classifiers determine the input distributions in the test phase. If they are perfect, then the training distribution equals the test distribution, and there is no problem. Otherwise, however, the distributions will differ.

- *The dependency among labels*: Perhaps most interestingly, a (strong enough) dependence between labels is a prerequisite for both, an improvement and a deterioration through chaining. In fact, CC cannot gain (compared to BR) in case of no label dependency. In that case, however, it is also unlikely to loose, because a classifier h_j will most likely[1] ignore the attributes $y_1, \ldots, y_{j-1}$. Otherwise, in case of pronounced label dependence, it will rely on these attributes, and whether or not this is advantageous will depend on the other factors above.

In the following, we present two experimental studies that are meant to illustrate the problem of error propagation in classifier chains.

4.1 Experiment with Real Data

Our intuition is that attribute noise in the test phase can produce a propagation of errors through the chain, thereby affecting the performance of the classifiers depending on their position in the chain. More specifically, we expect classifiers in the beginning of the chain to systematically perform better than classifiers toward the

[1] The possibility to ignore parts of the input information does of course also depend on the type of classifier used.

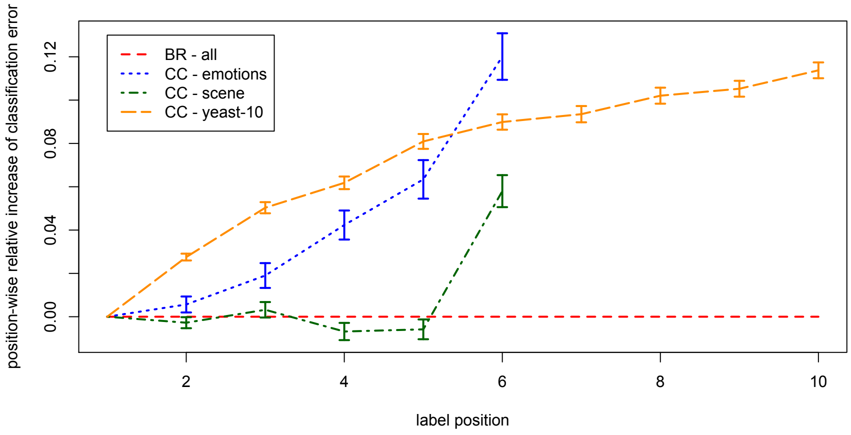

Figure 1: Results of the first experiment: position-wise relative increase of classification error (mean plus standard error bars).

end. In order to verify this conjecture, we perform the following simple experiment: We train a CC classifier on 500 randomly generated label orders. Then, for each label order and each position, we compute the performance of the classifier on that position in terms of the relative increase of classification error compared to BR. Finally, these errors are averaged *position-wise* (not label-wise). For this experiment, we used three standard MLC benchmark data sets: emotions (593 examples, 72 attributes, 6 labels), scene (2407, 294, 6), yeast-10 (2417, 103, 10); the latter is a reduction of the original yeast data set to the ten most frequent labels and their instances.

The results in Figure 1 clearly confirm our expectations. In two cases, CC starts to loose immediately, and the loss increases with the position. In the third case, CC is able to gain on the first positions but starts to loose again later on.

4.2 Experiment with Synthetic Data

In a second experiment, we used a synthetic setup that was proposed in [2] to analyze the influence of label dependence. The input space $\mathcal{X}$ is two-dimensional and the underlying decision boundary for each label is linear in these inputs. More precisely, the model for each label is defined as follows:

$$h_i(\boldsymbol{x}) = \begin{cases} 1 & a_{j1}x_1 + a_{j2}x_2 \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

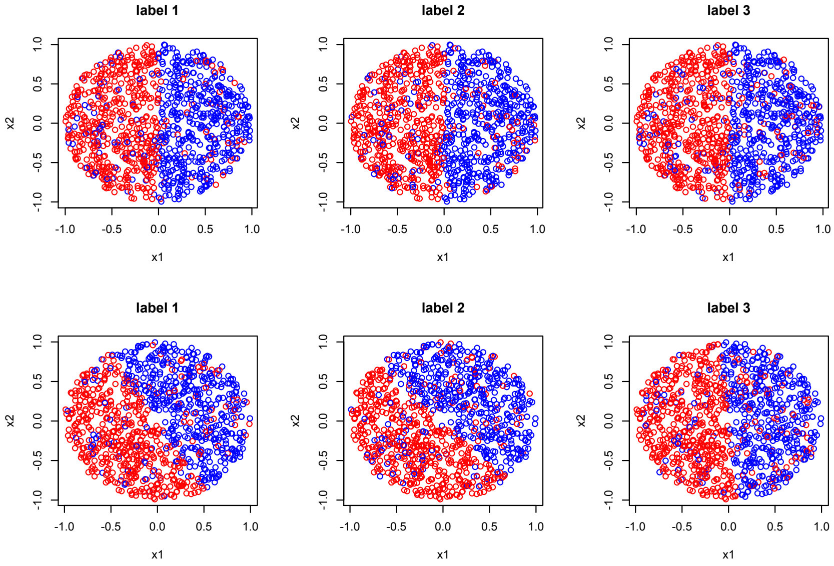

Figure 2: Example of synthetic data: the top three labels are generated using $\tau = 0$, the three at the bottom with $\tau = 1$.

The input values are drawn randomly from the unit circle. The parameters a_{j1} and a_{j2} for the j-th label are set to $a_{j1} = 1 - \tau r_1$, $a_{j2} = \tau r_2$, with r_1 and r_2 randomly chosen from the unit interval. Additionally, random noise is introduced for each label by independently reversing a label with probability $\pi = 0.1$. Obviously, the level of label dependence can be controlled by the parameter $\tau \in [0, 1]$: The smaller τ, the stronger the dependence tends to be (see Figure 2 for an illustration).

For different label cardinalities $m \in \{5, 10, 15, 20, 25\}$, we run 10 repetitions of the following experiment: We created 10 different random model parameter sets (two for each label) and generated 10 different training sets, each consisting of 50 instances. For each training set, a model is learned and evaluated (in terms of Hamming and subset 0/1 loss) on an additional data set comprising 1000 instances.

Figure 3 summarizes the results in terms of the average loss divided by the corresponding Bayes loss (which can be computed since the data generating process is known); thus, the optimum value is always 1. Comparing BR and CC, the big picture is quite similar to the previous experiment: The performance of CC tends to decrease with an increasing number of labels. In the case of less label dependence, this can already be seen for only five labels. The case of high label dependence is more interesting: While CC seems to gain from exploiting the dependency for a small to moderate number of labels, it cannot extend this gain to more than 15 labels.

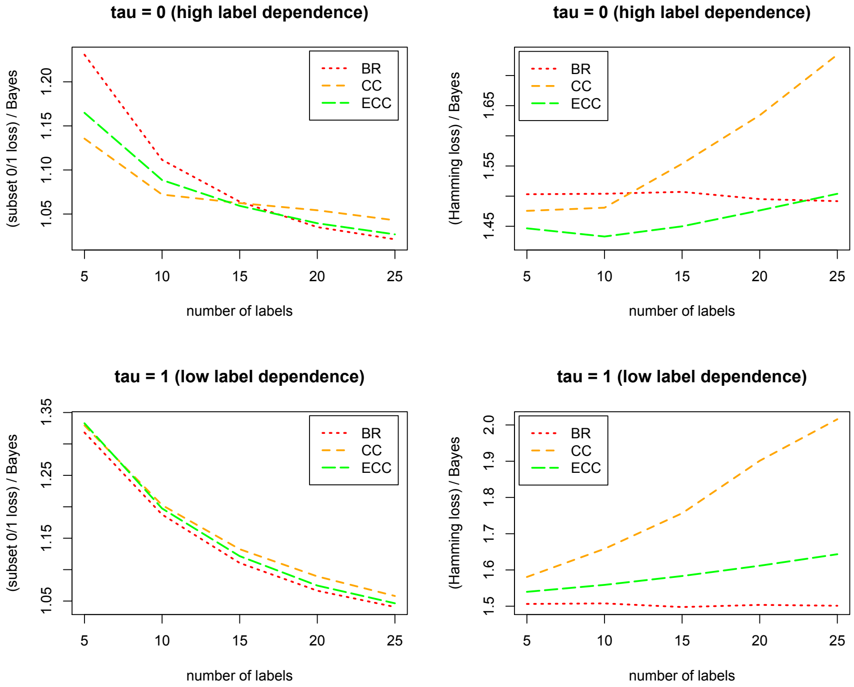

Figure 3: Results of the second experiment for $\tau = 0$ (top—high label dependence) and $\tau = 1$ (bottom—low label dependence).

5 Conclusion

This paper has thrown a critical look at the classifier chains method for multi-label classification, which has been adopted quite quickly by the MLC community and is now commonly used as a baseline when it comes to comparing methods for exploiting label dependency. Notwithstanding the appeal of the method and the plausibility of its basic idea, we have argued that, at second sight, the chaining of classifiers begs an important flaw: A binary classifier that has learned to rely on the values of previous labels in the chain might be misled when these values are replaced by possibly erroneous estimations at prediction time. The classification errors produced because of this attribute noise may subsequently be propagated or even reinforced along the entire chain. Roughly speaking, what looks as a gift at training time may turn out to become a handicap in testing.

Our results clearly show that this problem is relevant, and that it may strongly impair the performance of the CC method. There are several lines of future work. First, it is of course desirable to complement this study by meaningful theoretical results supporting our claims. Second, it would be interesting to investigate to what extent the problem of attribute noise also applies to the probabilistic variant of classifier chains introduced in [1].

References

[1] K. Dembczyński, W. Cheng, and E. Hüllermeier (2010) Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286.

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

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

[4] G. Tsoumakas and I. Katakis (2007) Multi label classification: An overview. *International Journal of Data Warehouse and Mining*, 3(3):1–13.

[5] X. Zhu and X. Wu (2004) Class noise vs. attribute noise: a quantitative study of their impacts. *Artificial Intelligence Review*, 22(3):177–210.