Improving stacking approach for multi-label classification

Elena Montañés, José Ramón Quevedo, and Juan José del Coz
{elena,quevedo,juanjo}@aic.uniovi.es

Artificial Intelligence Center, Universidad de Oviedo en Gijón (Spain)

Resumen The goal of multi-label classification is to obtain models able to tag labels to objects. The most straightforward method to tackle this task is called binary relevance and it consists of building an independent classifier for each label. However, it is well-known that taking into account possible dependences among label would help to provide more precise models. A method of this kind is the stacking approach, which tries to detect dependencies between labels in the last level of the stack. This paper studies this method, analyzing its properties and comparing it with other state-of-the-art approaches. Besides, another contribution consists of adding a modification that allows to improve its performance for some of the measures employed to evaluate the multi-label classification.

1. Introduction

Nowadays, all multimedia resources (text, video, music, films) are usually tagged in order to make easy the task of information retrieval. The labels assigned belong to a set of predefined labels and they are useful to provide a fast description of the resource. In machine learning, the task of obtaining models able to automatically tag new resources with these labels is called multi-label classification. From a formal point of view, the main difference with regard to the traditional classification is that each resource may simultaneously have more than one label, or in terms, of multi-class classification, it may simultaneously belong to more than a class. Hence, it is not possible to directly apply the known multi-class classifiers. This is the reason why researchers have been adapted several machine learning techniques to cope with this new problem, for instance, decision trees [3], instance-based algorithms [14] or support vector machines [5].

Binary relevance (BR) is the most simple approach to face with multi-label classification and it is usually employed as a baseline to be compare with other new methods. It consists of building an independent classifier for each label. The main disadvantage of BR is the assumption of considering that labels are independent each other, what it is not true in many real applications. Typical multi-label data include label dependences and it is necessary to take them into account to improve the performance of the classifiers.

This is the reason why most of the current research in multi-label classification is focused on developing new algorithms that detect and exploit label

dependencies. The proposed methods so far can be classified under two criteria: i) The label subset size within the dependences are explored, and ii) the kind of correlations they try to find. According to i), there are approaches that look for pairwise correlations [5,6], whereas others explore bigger subsets[9,12], including those that study the influence of the rest of the labels to predict one of them [2,8]. Regarding ii), there are algorithms that detect conditional dependence (for a certain instance), for instance [4,9,11]; or in conditional dependence (a kind of global dependence that does not depend on the individual instances) [2,8].

This paper deeply study a method able to induce models that take into account label dependencies. This is the approach proposed by Godbole and Sharawagi [8], called stacking, whose main contribution consists of employing stacked models [13] in the context of multi-label classification. These kind of approaches are based on building groups of stacked classifiers that the outputs of the classifiers of one level are used as inputs for the classifiers of the next level of the stack. The classifiers on the top of the stack determine the final prediction. The idea of the authors is to stack two levels of classifiers, where the first level is composed by independent classifiers for each label as those of BR, whose outputs are employed in the second level to build another group of classifiers that try to capture dependences among labels. This paper shows that this approach does not allow to discover all the correlations present among the labels and that the stacking idea introduce cumulative errors that dismiss the performance of the method. In this direction, we propose a modification that increase the performance for some measures commonly employed in multi-label classification.

The rest of the paper is organized as follows: Next section introduce the formal framework of multi-label classification. Section 3 describe the methods based on stacking paradigm, including our proposal. The paper ends with an experimental study and some conclusions.

2. Multi-label classification

Let $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$ be a finite and non-empty set of labels, and let $\mathcal{X}$ be an input space. We consider a multi-label classification task given by a training set $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, whose instances were independently and randomly obtained from an unknown probability distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$ on $\mathcal{X} \times \mathcal{Y}$, in which the output space $\mathcal{Y}$ is the power set of $\mathcal{L}$, in symbols $\mathcal{P}(\mathcal{L})$. In order to make the notation easier to read, we define $\boldsymbol{y}_i$ as a binary vector, $\boldsymbol{y}_i = \{y_1, y_2, \ldots, y_m\}$, in which each component $y_j = 1$ indicates the presence of label ℓ_j in the set of relevant labels of $\boldsymbol{x}_i$. Using this convention, the output space can be also defined as $\mathcal{Y} = \{0, 1\}^m$.

The goal of a multi-label classification is to induce a hypothesis $\boldsymbol{h} : \mathcal{X} \longrightarrow \mathcal{Y}$ from S, that correctly predicts the subset of labels from $\mathcal{L}$ for a new unlabeled instance $\boldsymbol{x}$. Without any loss of generality, this hypothesis can be seen as a combination of a collection of sub-hypotheses, $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots h_m(\boldsymbol{x}))$, one per label, in which each h_j takes the form of

$$h_j : \mathcal{X} \longrightarrow \{0, 1\}, \tag{1}$$

and it is able to predict if the label ℓ_j must be attached to the instance $\boldsymbol{x}$ or not.

This is the case of BR, which learns a binary classifier h_j for each label that only takes information of that label isolated from the rest. Despite its simplicity, BR present several advantages: i) many methods can be employed to obtain the binary classifiers h_j, ii) the complexity is linear with regard to the number of labels, and iii) it is easy to parallelize. The main drawback of BR is that it does not take into account any label dependence among labels, and then its performance is affected when these correlations happen. Despite that, when the binary classifiers h_j are induced by powerful algorithms with a special process of choosing their parameters, BR obtains acceptable results, even for some measures it is able to produce quite competitive results, as it is the case of Hamming loss (Eq. 6).

In order to evaluate the performance of the multi-label classifiers, several measures are commonly employed. A good description of them can be found in [11]. They can be classified under several criteria: example-based, label-based and ranking-based measures. This paper only considers example-based measures, since on one hand, some of them were proposed by the authors of stacking approach [8], and on the other hand they allow us to study if the compared method capture or not the dependencies among labels. Most of them are taken from the Information Retrieval field:

- **Jaccard index**, computes the percentage of relevant labels predicted in the subset formed by the union the returned and relevant labels[1],

$$Jaccard(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{\sum_{i=1}^{m} [\![y_i = 1 \text{ and } h_i(\boldsymbol{x}) = 1]\!]}{\sum_{i=1}^{m} [\![y_i = 1 \text{ or } h_i(\boldsymbol{x}) = 1]\!]}. \tag{2}$$

- **Precision**, determines the fraction of relevant labels in the predicted labels,

$$Precision(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{\sum_{i=1}^{m} [\![y_i = 1 \text{ and } h_i(\boldsymbol{x}) = 1]\!]}{\sum_{i=1}^{m} [\![h_i(\boldsymbol{x}) = 1]\!]}. \tag{3}$$

- **Recall** is the proportion of relevant labels correctly predicted,

$$Recall(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{\sum_{i=1}^{m} [\![y_i = 1 \text{ and } h_i(\boldsymbol{x}) = 1]\!]}{\sum_{i=1}^{m} [\![y_i = 1]\!]}. \tag{4}$$

- $\boldsymbol{F_1}$ is the evenly weighted harmonic mean of Precision and Recall,

$$F_1(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{2 \sum_{i=1}^{m} [\![y_i = 1 \text{ and } h_i(\boldsymbol{x}) = 1]\!]}{\sum_{i=1}^{m} ([\![y_i = 1]\!] + [\![h_i(\boldsymbol{x}) = 1]\!])}. \tag{5}$$

All the measures presented above are biased towards those methods that correctly predict the relevant labels. Besides, the performance of these kind of classifiers are usually evaluated in terms of other two measures:

[1] The expression $[\![p]\!]$ evaluates to 1 if the predicate p is true, and to 0 otherwise.

- **Hamming loss**, which is defined as the proportion of labels whose relevance is incorrectly predicted:

$$Hamming(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{1}{m}\sum_{i=1}^{m}[\![y_i \neq h_i(\boldsymbol{x})]\!]. \tag{6}$$

- **0/1 loss**, determine if the relevant and predicted label subsets are equal or not,

$$Zero - One(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = [\![\boldsymbol{y} \neq \boldsymbol{h}(\boldsymbol{x})]\!]. \tag{7}$$

3. Stacking approaches

Godbole and Sharawagi present a method [8] based on employing stacked models [13] in an attempt of overcoming the label independence of BR. In the learning stage, the method builds two groups of stacked classifiers. The first one includes the same classifiers yielded by BR, denoted by $\boldsymbol{h}^1(\boldsymbol{x}) = (h_1^1(\boldsymbol{x}), \ldots, h_m^1(\boldsymbol{x}))$. The second one, called meta-level, are formed again by one classifiers per label, but in this case they consider an augmented feature space:

$$h_j^2 : \mathcal{X} \times \{0,1\}^m \longrightarrow \{0,1\}, \tag{8}$$

where the m new features correspond to the outputs of the first-level classifiers, that is, $\boldsymbol{h}^2(\boldsymbol{x}, \boldsymbol{y}') = (h_1^2(\boldsymbol{x}, \boldsymbol{y}'), \ldots, h_m^2(\boldsymbol{x}, \boldsymbol{y}'))$, where $\boldsymbol{y}' = \boldsymbol{h}^1(\boldsymbol{x})$. The goal is that these second-level classifiers capture the relationships among labels. In order to predict the labels of a new example, the outputs of these second-level classifiers are provided, $\boldsymbol{h}^2(\boldsymbol{x}, \boldsymbol{h}^1(\boldsymbol{x}))$. Hence, the outputs of the first-level classifiers, $\boldsymbol{h}^1(\boldsymbol{x})$, are only taken for obtaining the values of the augmented feature space of the second-level classifiers.

3.1. The improved stacking approach

In terms of probabilities, the meta-level binary classifiers of stacking approach estimate the probability $\mathbf{P}(y_j|\boldsymbol{x}, \boldsymbol{y}')$, being $\boldsymbol{y}'$ in turn a estimation of the relevant labels that only depend on the description of the object $\boldsymbol{x}$. This chain of estimation may explain why the information contained in $\boldsymbol{y}'$ is not adequate enough to infer the dependence of label y_j from the rest. Stacking approach is absolutely formal from the learning point of view, since the source of information is the same both in training and testing stages (the outputs of the first-level classifiers). But considering the reliability of the training data, their information is less noisy and it contains the actual information about the relationships between labels. In the original proposal of stacking method, the outputs of the first-level classifiers $\boldsymbol{y}'$ contain the typical errors of the models that estimate them, that, in turn may produce new errors in the meta-level classifiers.

For this reason, our proposal is based on modifying the way classifiers are trained in the meta-level. Instead of using the predictions provided by the first-level models, we employ the actual information of the labels, that is, the information

included in the training data, obviously removing the information of the own label j. Our classifiers h_j^2 are of the form:

$$h_j^2 : \mathcal{X} \times \{0,1\}^{m-1} \longrightarrow \{0,1\}, \tag{9}$$

where the feature space will be completed by the actual information contained in the rest of the $m-1$ labels. That is, the second group of classifiers will be $\boldsymbol{h}^2(\boldsymbol{x},\boldsymbol{y}) = (h_1^2(\boldsymbol{x}, y_2, \ldots, y_m), \ldots, h_m^2(\boldsymbol{x}, y_1, \ldots, y_{m-1}))$. According to the classification mentioned in section 1, these classifiers try to detect conditional dependence among all labels. Theoretically the estimations they carry out, $\mathbf{P}(y_j|\boldsymbol{x}, y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_m)$, must be more precise than those provided by the stacking models, $\mathbf{P}(y_j|\boldsymbol{x}, \boldsymbol{y}')$. This issue will be experimentally shown in section 4.

3.2. Other related methods

Some variants of the approach proposed in [8] have been explored, mainly focused on reducing the meta-level space of features. The target is to remove the information provided by those labels that are not related with the label j that is predicted by the model h_j^2. For instance, in [10] the authors propose to compute the χ coefficient for each pair of labels (j,k) taking all training data. The method removes the information of all labels k whose correlation with label j falls below certain threshold. This approach is more efficient, keeping its performance.

In [1], authors include two modifications to the original stacking approach: i) They remove the estimation of the classifier h_j^1 when the h_j^2 is built, that is, they do not include the information of the own label, and ii) they used a predefined order for estimating each label based on its occurrence in the prediction stage.

Read et al. describe [9] classifier chains (CC) able to model correlations among labels keeping the same computational complexity as BR. In training phase, CC orders the labels in a chain and builds m binary classifiers, one per each label. The feature space of each classifier increases with information of the labels previously placed in the chain. For instance, if the chain follows the same order of the labels, then the classifier h_j will be:

$$h_j : \mathcal{X} \times \{0,1\}^{j-1} \longrightarrow \{0,1\}, \tag{10}$$

using the labels located first in the chain, $y_1, \ldots, y_{j-1}$, to augment the feature space. In the prediction time, the classifiers are applied following the order of the chain in a way that the outputs of the previous classifiers are employed to augment the successive feature space of the next classifiers. In the example presented before, it will be $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, h_1(\boldsymbol{x})), h_3(\boldsymbol{x}, h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, h_1(\boldsymbol{x}))), \ldots)$. Obviously, the order of the chain makes influence on the performance of the method. Although it is possible to include heuristics for determining the order, the authors propose to ensemble several classifier chains (ECC) using different random orders of the labels and different random subsets of data.

Notice that CC has in common with our proposal that it employs actual label information in order to build the augmented feature spaces. The difference

Tabla 1. Properties of the datasets employed in the experiments

Dataset	Features	Examples	Labels	Cardinality
bibtex	1836	7395	159	2.40
emotions	72	593	6	1.87
enron	1001	1702	53	3.38
genbase	1185	662	27	1.25
image	135	2000	5	1.24
mediamill	120	5000	101	4.27
medical	1449	978	45	1.25
reuters	243	7119	7	1.24
scene	294	2407	6	1.07
slashdot	1079	3782	22	1.18
yeast	103	2417	14	4.24

is that CC does not use all the labels, otherwise it only employs the previous ones in the chain. This can lead to poor estimations, especially when a label that depends on certain other labels is placed in the chain before, which may more frequently occur for the labels located in the first positions of the chain. In fact, CC assumes a chain relationship among labels, when the dependences actually may be much more complex. However, our approach guarantees that when a classifier for one label is built, it always has information of all other labels, including always those that it may depend on, although detecting dependences may be more complex because the feature spaces are bigger.

There exist other methods, less related with the ones mentioned above, but that they try to find dependences among labels. RAkEL(RAndom k-labELsets) [12] iteratively obtains and combines several classfiers LP (Label Power-set). An LP considers each subset of labels present in the training data as a class of a multi-class problem. In each iteration i, RAkEL randomly selects, without replacement, a set of labels $\boldsymbol{Y}_i$ of size k. Then, it learns a LP classifier of the form $\mathcal{X} \rightarrow \mathcal{P}(\boldsymbol{Y}_i)$. Finally, the labels are predicting through a voting scheme.

IBLR (Instance-Based Learning by Logistic Regression) [2] unifies instance-based learning and logistic regression. Focusing on the way of detecting dependencies, the main idea consists of extending the description of each example $\boldsymbol{x}$ adding features that express the presence of each label in the neighborhood of $\boldsymbol{x}$.

4. Experiments

The goal of the experiments was twofold. Firstly, the aim was to study the behavior of our method and the original stacking approach. Secondly, the target was to widely compare these methods with other state-of-the-art approaches to cope with multi-label classification. Several public and benchmark datasets, whose properties can be seen in Table 1, were taken. Notice that they are quite

Tabla 2. Precision, Recall, F_1 and Jaccard for all the methods

Precision	BR	STA	STA$^{\boldsymbol{y}}$	ECC*	MLkNN	IBLR	RAkEL
Bibtex	48,19(1)	48,01(2)	47,17(4)	47,69(3)	26,60(7)	28,92(6)	47,08 (5)
Emotions	56,36(4)	56,58(3)	62,09(2)	56,26(5)	52,42(7)	67,54(1)	52,79 (6)
Enron	69,99(1)	65,18(4)	65,51(2)	65,19(3)	54,90(6)	52,75(7)	56,05 (5)
Genbase	99,52(3,5)	99,40(5)	99,60(1)	99,52(3,5)	97,70(7)	98,90(6)	99,57 (2)
Image	44,23(7)	44,54(5)	53,88(1)	45,99(4)	44,33(6)	48,52(2)	46,66 (3)
Mediamill	78,81(2)	43,49(7)	70,11(6)	77,87(3)	76,93(4)	73,52(5)	80,40 (1)
Medical	78,94(5)	79,47(4)	81,33(1)	81,04(2)	62,43(7)	63,40(6)	80,79 (3)
Reuters	85,79(5)	85,89(4)	87,50(2)	86,41(3)	82,23(6)	70,71(7)	89,62 (1)
Scene	61,46(7)	67,13(5)	66,14(6)	67,45(4)	69,71(2)	71,40(1)	69,69 (3)
Slashdot	46,06(4)	47,91(3)	53,20(1)	42,79(5)	6,15(7)	8,09(6)	50,91 (2)
Yeast	71,13(3)	70,81(4)	66,80(7)	70,58(5)	72,92(1)	71,75(2)	68,62 (6)
Prom. rank.	(3,86)	(4,18)	(3,00)	(3,68)	(5,45)	(4,45)	(3,36)
Recall	BR	STA	STA$^{\boldsymbol{y}}$	ECC*	MLkNN	IBLR	RAkEL
Bibtex	33,80(4)	35,49(2)	34,82(3)	33,78(5)	14,06(7)	21,50(6)	41,97 (1)
Emotions	48,16(6)	49,81(4)	65,84(1)	48,92(5)	37,73(7)	64,54(2)	57,19 (3)
Enron	50,50(4)	56,68(2)	57,48(1)	45,79(5)	37,04(7)	38,04(6)	54,07 (3)
Genbase	99,07(4,5)	99,07(4,5)	99,07(4,5)	99,07(4,5)	94,96(7)	99,14(2)	99,57 (1)
Image	43,32(6)	43,54(5)	53,68(1)	44,10(3)	39,11(7)	43,68(4)	49,73 (2)
Mediamill	52,33(5)	60,02(1)	54,34(3)	51,25(6)	53,78(4)	56,69(2)	49,57 (7)
Medical	78,34(5)	83,09(1)	81,02(2)	79,01(4)	59,01(7)	65,05(6)	81,00 (3)
Reuters	84,90(5)	84,93(4)	90,29(1)	85,19(3)	81,09(6)	69,45(7)	89,63 (2)
Scene	62,87(7)	68,17(5)	82,38(1)	66,43(6)	68,73(4)	69,75(2)	69,52 (3)
Slashdot	44,21(4)	45,96(3)	70,37(1)	39,93(5)	5,69(7)	7,67(6)	53,18 (2)
Yeast	58,86(6)	59,38(5)	60,93(2)	59,40(4)	56,89(7)	60,41(3)	61,84 (1)
Prom. rank.	(5,14)	(3,32)	(1,86)	(4,59)	(6,36)	(4,18)	(2,55)
$\boldsymbol{F_1}$	BR	STA	STA$^{\boldsymbol{y}}$	ECC*	MLkNN	IBLR	RAkEL
Bibtex	37,02(4)	37,92(2)	37,54(3)	36,86(5)	16,98(7)	22,38(6)	41,28 (1)
Emotions	49,20(6)	50,33(4)	60,87(2)	49,81(5)	41,60(7)	62,97(1)	51,95 (3)
Enron	55,66(3)	57,78(2)	58,20(1)	51,14(5)	41,82(6)	41,52(7)	52,43 (4)
Genbase	99,18(3,5)	99,10(5)	99,21(2)	99,18(3,5)	95,81(7)	98,78(6)	99,50 (1)
Image	42,12(6)	42,38(5)	51,68(1)	43,46(4)	40,63(7)	44,91(3)	46,32 (2)
Mediamill	59,17(3)	47,06(7)	57,23(6)	58,15(4)	59,55(2)	60,17(1)	57,72 (5)
Medical	77,33(5)	79,45(3)	79,82(1)	78,83(4)	59,41(7)	62,19(6)	79,65 (2)
Reuters	84,13(5)	84,21(4)	87,05(2)	84,69(3)	80,50(6)	69,10(7)	88,58 (1)
Scene	61,25(7)	66,75(5)	68,46(4)	66,31(6)	68,49(3)	69,97(1)	68,84 (2)
Slashdot	44,33(4)	46,05(3)	55,47(1)	40,66(5)	5,84(7)	7,73(6)	50,49 (2)
Yeast	61,68(5)	61,86(3)	60,98(6)	61,80(4)	60,97(7)	62,85(1)	62,48 (2)
Prom. rank.	(4,68)	(3,91)	(2,64)	(4,41)	(6,00)	(4,09)	(2,27)
Jaccard	BR	STA	STA$^{\boldsymbol{y}}$	ECC*	MLkNN	IBLR	RAkEL
Bibtex	31,50(4)	32,15(3)	32,32(2)	31,46(5)	13,61(7)	18,09(6)	34,28 (1)
Emotions	42,27(6)	43,46(3)	51,76(2)	43,24(4)	34,09(7)	55,08(1)	42,98 (5)
Enron	44,69(3)	45,95(2)	47,09(1)	39,68(5)	31,83(7)	31,99(6)	41,30 (4)
Genbase	98,94(3,5)	98,82(5)	98,97(2)	98,94(3,5)	94,86(7)	98,25(6)	99,29 (1)
Image	38,60(6)	38,87(5)	47,32(1)	40,15(4)	38,45(7)	42,46(2)	42,15 (3)
Mediamill	46,70(3)	34,09(7)	45,42(4)	44,69(6)	48,11(2)	48,82(1)	45,10 (5)
Medical	74,51(5)	75,43(4)	76,95(1)	76,33(3)	56,76(7)	58,19(6)	76,88 (2)
Reuters	81,67(5)	81,76(4)	84,00(2)	82,41(3)	78,11(6)	67,10(7)	86,38 (1)
Scene	59,41(7)	64,88(5)	64,00(6)	65,05(4)	67,03(3)	68,77(1)	67,28 (2)
Slashdot	42,71(4)	44,29(3)	49,70(1)	39,32(5)	5,68(7)	7,42(6)	47,18 (2)
Yeast	50,71(5)	50,93(4)	49,68(7)	50,96(3)	50,50(6)	52,65(1)	51,75 (2)
Prom. rank.	(4,68)	(4,09)	(2,64)	(4,14)	(6,00)	(3,91)	(2,55)

Tabla 3. *Hamming* loss y 0/1 loss for all the methods

Hamming	BR	STA	STA$^{\boldsymbol{y}}$	ECC*	MLkNN	IBLR	RAkEL
Bibtex	1,21(2)	1,26(4)	1,21(2)	1,21(2)	1,36(5)	1,60(7)	1,49 (6)
Emotions	22,03(4)	21,83(3)	23,15(5)	21,72(2)	26,21(6)	18,72(1)	28,01 (7)
Enron	4,46(1)	4,83(3)	4,88(4)	4,72(2)	5,22(5)	5,60(6)	5,85 (7)
Genbase	0,08(3,5)	0,09(5)	0,07(2)	0,08(3,5)	0,45(7)	0,19(6)	0,06 (1)
Image	20,25(5)	20,23(4)	21,61(6)	19,80(3)	19,28(2)	18,75(1)	24,40 (7)
Mediamill	2,76(2)	5,50(7)	3,10(6)	2,86(5)	2,70(1)	2,82(4)	2,81 (3)
Medical	0,99(4)	1,12(5)	0,98(3)	0,95(1,5)	1,56(6)	1,90(7)	0,95 (1,5)
Reuters	4,58(3)	4,60(4)	5,77(5)	4,42(2)	6,03(6)	8,32(7)	3,85 (1)
Scene	9,83(4)	9,85(5)	18,31(7)	9,06(3)	8,66(2)	8,38(1)	9,90 (6)
Slashdot	3,73(1)	3,86(3)	8,80(7)	3,78(2)	5,18(6)	5,17(5)	4,65 (4)
Yeast	19,81(3)	19,85(4)	21,56(7)	19,98(5)	19,43(2)	19,18(1)	20,30 (6)
Prom. rank.	(2,95)	(4,27)	(4,91)	(2,82)	(4,36)	(4,18)	(4,50)
0/1	BR	STA	STA$^{\boldsymbol{y}}$	ECC*	MLkNN	IBLR	RAkEL
Bibtex	82,83(3)	82,99(4)	81,54(1)	82,61(2)	94,06(7)	91,64(6)	83,56 (5)
Emotions	79,42(5)	77,91(4)	74,70(2)	77,05(3)	87,00(7)	68,97(1)	83,45 (6)
Enron	86,90(2)	88,07(3)	85,25(1)	93,07(6)	94,89(7)	92,30(5)	88,49 (4)
Genbase	1,81(3)	2,11(5)	1,81(3)	1,81(3)	8,16(7)	4,08(6)	1,51 (1)
Image	71,50(7)	71,25(6)	65,20(2)	69,40(4)	67,95(3)	64,70(1)	69,70 (5)
Mediamill	90,36(4)	97,90(7)	88,14(3)	93,80(6)	86,24(2)	85,86(1)	91,14 (5)
Medical	33,94(4)	36,61(5)	31,49(3)	31,19(1)	51,12(6)	52,98(7)	31,39 (2)
Reuters	25,69(5)	25,54(4)	24,60(3)	24,43(2)	29,04(6)	38,88(7)	20,24 (1)
Scene	46,03(7)	40,63(5)	44,99(6)	38,72(4)	37,35(2,5)	34,82(1)	37,35 (2,5)
Slashdot	61,95(2)	60,82(1)	62,85(4)	64,57(5)	94,76(7)	93,47(6)	62,11 (3)
Yeast	84,53(7)	83,95(5)	84,07(6)	83,58(4)	82,29(2)	79,19(1)	83,08 (3)
Prom. rank.	(4,45)	(4,45)	(3,09)	(3,64)	(5,14)	(3,82)	(3,41)

different in terms of number of features, examples, labels and also in terms of cardinality (number of labels per example).

The compared algorithms were BR, the original stacking approach [8] (denoted by STA), our variant (STA$^{\boldsymbol{y}}$) and other state-of-the-art algorithms: MLkNN [14], RAkEL [12], IBLR [2] and ECC [9] (version presented in [4] and called ECC*). Logistic regression was employed as base learner to build the binary individual models. In all cases, the value of the parameter C was set for each binary classifier isolated from the rest using a grid search in $C \in [10^{-3}, \ldots, 10^3]$ and optimizing the binary classification error, estimated by means of a stratified and 2-fold cross-validation with 5 repetitions.

Tables 2 and 3 show the results of all methods in terms of the evaluation measures described in section 2. The results are expressed in percentage and they correspond to a stratified and 10-fold cross-validation. The order of the method in the ranking is shown in brackets. In case of ties, the results are averaged. The average of the positions of each method is indicated in the last row of the correspondent table. Following the recommendations presented in [7], a two-stage statistical procedure was carried out. Firstly, a Friedman test was employed to discard the null hypothesis that states that all the methods perform equally. Secondly, a pairwise comparison was carried out using a Bergmann-Hommel test [7].

Looking at Tables 2 and 3, our proposal outperforms the original stacking method in terms of all measures, except for Hamming loss. Considering all the algorithms, our method obtains the best results in three of the measures (Precision, Recall and 0/1 loss), it is placed in second position in two of them (F_1 and Jaccard), behind RAkEL and it is the worst for Hamming loss, for which ECC* is the best. The method STA does not reach the first two positions for any measure and BR only does it for Hamming loss.

All these results indicates that our proposal is able to better detect relevant labels, as Recall shows, but it tends to add some irrelevant labels, as Hamming loss indicates. In any case, the performance is good taking into account the values reached by F_1.

In any case, the experiments also shown that there hardly exist significant differences among all the algorithms. Particularly, none significant differences exist in Precision, Hamming loss and 0/1 loss. However, RAkEL and our method are significantly better than MLkNN in terms of F_1 and Jaccard at significant level of 95 %. The only measure in which more differences appear is Recall, where our method is significantly better than ECC*, MLkNN and BR at 95 %. Besides, RAkEL is significantly better than MLkNN and BR at 95 % and STA is better than MLkNN also at 95 %.

5. Conclusions

This paper proposed a variant of the stacking approach for multi-label classification that allows improving the performance in terms of the most common evaluation measures. The main idea of the method is to detect the conditional dependence among all the labels. For this purpose, the models are built taking into account not only the description of the objects, otherwise they consider information of the rest of the labels during the learning stage. The results of the experiments carried out over eleven datasets show that our method obtains good results in terms of those evaluation measures that favor relevant labels. The main drawback is that it tends to predict irrelevant labels as relevant ones.

Referencias

1. E. Alvares, J. Metz, and M. Monard. A Simple Approach to Incorporate Label Dependency in Multi-label Classification. In *Advances in Soft Computing*, volume 6438 of *Lecture Notes in Computer Science*, chapter 3, pages 33–43. Springer, 2010.
2. W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.
3. A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. In *European Conf. on Data Mining and Knowledge Discovery*, pages 42–53, 2001.
4. K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In *ICML*, pages 279–286, 2010.
5. A. Elisseeff and J. Weston. A Kernel Method for Multi-Labelled Classification. In *ACM Conf. on Research and Develop. in Infor. Retrieval*, pages 274–281, 2005.

6. J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73:133–153, 2008.
7. S. García and F. Herrera. An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.
8. S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia Conf. on Know. Disc. and Data Mining*, pages 22–30, 2004.
9. J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *ECML'09*, LNCS, pages 254–269. Springer, 2009.
10. G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, and I. Vlahavas. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Workshop on learning from multi-label data*, pages 101–116, 2009.
11. G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Data Mining, 2010.
12. G. Tsoumakas and I. Vlahavas. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, LNCS, pages 406–417. Springer, 2007.
13. D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:214–259, 1992.
14. M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.