Declaratively Capturing Local Label Correlations with Multi-Label Trees

Reem Al-Otaibi 1 2 and Meelis Kull 1 3 and Peter Flach 1

Abstract. The goal of multi-label classification is to predict multiple labels per data point simultaneously. Real-world applications tend to have high-dimensional label spaces, employing hundreds or even thousands of labels. While these labels could be predicted separately, by capturing label correlation we might achieve better predictive performance. In contrast with previous attempts in the literature that have modelled label correlations globally, this paper proposes a novel algorithm to model correlations and cluster labels locally. LaCovaC is a multi-label decision tree classifier that clusters labels into several dependent subsets at various points during training. The clusters are obtained locally by identifying the conditionally-dependent labels in localised regions of the feature space using the label correlation matrix. LaCovaC interleaves between two main decisions on the label matrix with training instances in rows and labels in columns: splitting this matrix vertically by partitioning the labels into subsets, or splitting it horizontally using features in the conventional way. Experiments on 13 benchmark datasets demonstrate that our proposal achieves competitive performance over a wide range of evaluation metrics when compared with the state-of-the-art multi-label classifiers.

1 Introduction and Motivation

In traditional classification each data point is assigned to a single label. In binary classification each point can belong to one of two classes, whereas in multi-class classification the setting is more general, allowing each training point to belong to one of more than two classes. Multi-label classification generalises both by allowing more than one label for each data point [20, 28]. Thus, it allows for a wide range of applications, such as text categorisation, image and movie tagging, and gene function prediction. For example, a medical diagnosis might find a patient has multiple diseases at one time; an article that gives statistics about the number of students who have applied to Medical schools in a country could be categorised as both educational and medical; and an image that captures a beach at sunset could belong to both beach and sunset groups. Thus, all these examples naturally yield multiple labels.

Existing approaches in multi-label learning can be categorised into two main types. Problem transformation approaches decompose multi-label data into several binary problems, in order to use a binary classifier. For example, a multi-label problem with $|L|$ labels can be solved with $|L|$ binary classifiers, in which all predictions are then merged to produce final predictions. In the second type of approaches the algorithms handle multi-label data directly.

The main challenge to note is that labels in real-world applications often have a relationship or connection, whereby the presence of one label affects or depends on another. Several studies argue that exploiting label correlations is important in the area of multi-label classification [7, 27, 28]. Although a considerable amount of work has been done in this area, these have chiefly focused on a global approach, in which label correlations are identified as a pre-processing step prior to training the model.

Decision tree algorithms are among the most widely used algorithms for classification [13, 17]. Considering the advantages of decision tree models, this paper proposes LaCovaC, which is a multi-label decision tree classifier. LaCovaC utilises the label correlation matrix at every node of the tree to find possible clusters among the labels. In addition to internal nodes that split the dataset horizontally based on selected features, LaCovaC introduces a second kind of node for splitting the label space vertically. At deployment, a horizontal split routes to exactly one child node according to a feature value as per normal, while a vertical split tests all outgoing edges to collect predictions about the entire labelset.

The remainder of this paper is organised as follows. Section 2 provides an overview of existing approaches to exploit correlations in multi-label classification. LaCovaC is presented in Section 3, and an experimental evaluation is presented and discussed in Section 4. Section 5 concludes the paper by stating the main findings and possible avenues for further work.

2 Related Work

Two well-known baseline algorithms have been considered for use as problem transformation methods: Binary Relevance (BR) [19] and Label Powerset (LP) [23]. BR applies one binary classifier to each individual label. It transforms the original dataset $D$ into $|L|$ datasets, each of which comprises all examples of the original dataset. The examples are labelled positively if the labelset for the original example contains this label, and negatively if not. To classify a new instance, BR outputs the union of labels that have been positively predicted by the $|L|$ classifiers. Label Powerset (LP), also known as Label Combination, considers each unique set of labels that exists in a multi-label training set as a new class in a multi-class classification task. It is apparent that BR does not model label dependency, whereas LP does. However, overfitting and the exponential number of label combinations are potential difficulties affecting LP. Therefore, many different directions have been taken in the literature to address label correlations. We summarise these into several distinct approaches below.

The first approach is to transform a multi-label problem into several binary classification tasks by considering label correlations. A well-known algorithm called the Classifier Chain (CC) involves $|L|$ binary classifiers as in BR, but orders them along a chain, wherein

---

1 Intelligent System Laboratory, University of Bristol, United Kingdom, emails: ral2404, meelis.kull, peter.flach@bristol.ac.uk
2 King Abdulaziz University, Saudi Arabia
3 University of Tartu, Estonia
each classifier deals with the binary relevance problem associated with a label. Importantly, the feature space of each link in the chain is extended with the 0/1 label predictions for all previous links [16].

The same authors of CC have proposed the Ensembles of Classifier Chains (ECC), an ensemble method where the individual classifiers are trained with different orders of labels in the chain and a random subset of the training data, encouraging variability among the classifiers. These predictions are summed per label so that each label receives a number of votes. A threshold is used to select the most popular labels, which form the final predicted label set [16].

Clearly, CC and ECC are sensitive to the label order in the training process. A number of extensions to the original CC method have been proposed in [6, 7, 11, 15, 26], aiming at eliminating the key drawback in the CC, which is the lack of a principled way to decide on the label ordering.

Although CC considers label correlations by inserting the labels as new features, there is an important point to consider. It is the fact that all labels are inserted as additional features along the chain until the last label in the chain contains all previous labels as extra input features. This can be a limiting factor in high-dimensional label spaces. Furthermore, these methods force all preceding labels to be additional features for the examined label which might not be relevant or useful.

The second approach is to exploit correlations between labels by clustering them. The hierarchy of multi-label classifier (Homer) organises all labels into a tree-shaped hierarchy, with leaf nodes containing a single label [21]. Each node has training instances that are annotated with at least one of its labels. In the training phase, a multi-label classifier is trained for each node to predict a subset of labels in that node. In particular, a leaf node constructs a binary classifier to predict its single label. Given an unseen instance, Homer starts from the root node and proceeds to any successor node only if at least one of its labels was predicted by its parent node. In the end, this process reaches a subset of leaves and the final prediction is combined from predictions of these leaves. A recent work proposed in [4] combines the LP and BR methods, and is called LPBR. Its first step is to explore the dependencies between labels and then to cluster these labels into several independent subsets, according to the chi-squared statistic. Subsequently, a multi-label classifier is learnt: if the set contains only one label, BR is applied; for a group of dependent labels, LP is used. LPBR implements a greedy clustering algorithm that continues clustering as long as the loss function improves. While LPBR showed an improvement in terms of classification accuracy, it is computationally expensive.

In [12] the authors propose a method ML-LOC which first clusters the instances with respect to similarity of features and labels jointly. Subsequently, the feature space is extended with an additional feature encoding the cluster membership. Given a test instance, this additional feature is predicted using a regression model. The final predictions are then obtained from any standard multi-label classifier, trained on the extended feature space.

The final approach is to adopt decision tree algorithms in a multi-label setting. ML-C4.5 was proposed by [5] to deal with multi-label data, while the basic strategy was to define multi-label entropy separately over a set of multi-label examples. The modified entropy sums the entropies for each individual label. Another recent work was proposed in [10], and also builds a single tree for a multi-label dataset. They proposed a hybrid decision tree architecture to utilise support vector machines (SVMs) at its leaves. This approach, known as ML-SVMDT, combines two models: ML-C4.5 and BR. It builds a single decision tree, similar to ML-C4.5, whose leaves contain BR classifiers giving multi-label predictions using SVM. LaCova was proposed in [2] and is a tree based multi-label classifier that uses label covariance as a splitting criterion. The principle of LaCova is to use the label covariance matrix at each node of the tree to treat labels independently (i.e., learn a BR model from then on) or keep them together (LP) for now.

In this work we explore the value of mediating between these extreme decisions at each node in the tree. We propose LaCovaC which – different from previous methods – clusters labels dynamically during the construction of the decision tree, and hence models conditional label correlations at every node of the tree. Although other models attempt to cluster labels, they do so over the entire dataset, e.g. Homer and LPBR. In practice, conditional correlations that are used for clustering may be local, and depend on specific feature values. Hence LaCovaC will not separate labels automatically as happens in BR, but only in cases when labels are uncorrelated. Additionally, LaCovaC would not model the joint distribution at all times, as this can cause overfitting, as in LP. These decisions are taken locally at every node in the tree.

### 3 The Proposed Model

The key theoretical underpinning of decision trees is that they identify regions in instance space with low label variance, which is built into the splitting criterion. Naive extensions to the multi-label setting would either lead to a separate tree for each label (as in BR) or keeping all labels together (as in LP). Learning a separate tree for each label would result in as many trees as there are labels, which can be hundreds or thousands in some domains (in our experiments it can be up to 374). Furthermore, the explanatory power of the decision tree models would be reduced, which is an important factor in medical domains, among many others.

#### 3.1 An Illustrative Example

Before providing a formal definition, a simple example is introduced here to illustrate the proposed algorithm. IMDB is a multi-label dataset that contains keywords describing movies, and the classification task is to predict the movie’s genre. Each movie can be assigned multiple genres from among 27 labels. For simplicity, we have selected two input features: dark and love; and three movie genres: crime, horror and drama.

<table>
<thead>
<tr>
<th>labels</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>crime</td>
<td>horror</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

Inspired by the IMDB dataset we have created a toy dataset, shown in Table 1, which we use to demonstrate the advantages of LaCovaC.

4 http://meka.sourceforge.net/
over binary relevance. Figure 1 shows the tree generated using LaCovaC. In this example, LaCovaC initially finds two label clusters: \{crime, horror\} and \{drama\}. Therefore, it splits the dataset vertically, and recursively creates a tree for each cluster of labels. On the left-hand side of the tree a horizontal split can be observed at the next level using the feature dark, leading to another vertical split on the left and a node that requires further horizontal splitting. Note that the second vertical split did not split further due to a minimum cardinality constraint (2 instances in this toy example). As can be seen on the left-hand side of the tree, \{crime, horror\} are independent when dark is false, and dependent otherwise.

When the feature love is true both labels share the same estimated marginal probability of 0.6, which can lead to predicting both of them as relevant labels, assuming 0.5 as a threshold. However, the proposed model suggests that the two subsets \{crime=0, horror=1\} or \{crime=1, horror=0\} with the highest probability among other subsets can lead to a better prediction. In other words, leaves predict the label subset that has the highest probability among others, regardless of their marginal probabilities. With regards to the third label \{drama\}, the proposed algorithm learns it separately as can be seen on the right-hand side of the tree.

This example demonstrates the power of modelling label correlations locally in the tree instead of globally in pre-processing, and also that the tree model represents such local correlations in a declarative way.

When applying the learned model on a test instance, one outgoing branch of a horizontal split should be followed based on the feature value as in a standard decision tree. In contrast, in a vertical split, all outgoing branches should be followed to collect predictions for all labels reaching the node. For example, in order to label a movie which has features \{dark=1, love=1\}, the two branches of the first vertical split should be visited. On the right-hand side of the tree, we find that the drama label does apply. On the left-hand side of the tree, we will have a horizontal split based on the dark feature leading to the second horizontal node based on the love feature. The right leaf of this second horizontal split suggests two label subsets: \{crime=1, horror=0\} or \{crime=0, horror=1\}. Suppose that the first set is chosen, then the overall predictions for this movie will be the following labels \{crime=1, horror=0, drama=1\}.

Figure 2 shows the result of binary relevance on this example with single-label decision trees as base model. The leftmost tree is for the crime label; the root node splits on the feature dark, when dark=0 it leads to a leaf due to a minimum cardinality constraint (2 instances). On the right-hand side of the tree, it splits further based on the feature love. Leaves show the estimated probabilities and the predicted label. The middle and right-most tree were constructed using the same method to learn labels horror and drama, respectively.

In order to predict genres for a new movie with \{dark=1 and love=1\} all three trees should be tested. The prediction for this movie will be \{crime, horror, drama\}, using 0.5 as a threshold. This is different from what the proposed model LaCovaC predicts.

### 3.2 The Main Algorithm

LaCovaC implements three key decisions at every node of the decision tree, and the process can be summarised as follows. Firstly, if labels are pure or the set of instances reaches the minimum number of data points, the algorithm stops and returns a leaf. Secondly, if the label’s correlation matrix suggests the presence of cluster structure, the algorithm splits the dataset reaching that node according to the label clusters (vertical split). Finally, a set of labels located together at a node are learned, and the best features to split are determined (horizontal split).

The first decision requires a threshold on the label variance. In our experiments we found that, in combination with a minimum number of instances at a leaf, a variance threshold of 0 (i.e. all labels are pure) works well. The detection of cluster structure is presented in the next section. The third decision that splits instance space horizontally also demands a splitting criterion. The most popular splitting criterion in standard decision trees learning are Gini-split (which uses the Gini index to measure impurity) and information gain (which uses Shannon entropy). However, LaCovaC implements a splitting criterion designed specifically for multi-label data, following a previous work [2], which can be summarised as follows. It evaluates for each child both its sum of label variances and its sum of absolute label covariances, and assigns as a splitting measure the minimum of these two (low is better). Then, it selects the feature that has the lowest splitting measure. This criterion can identify regions with either low multi-label variance or low label covariance. We improve on this by clustering the labels in a principled way into independent groups of correlated labels.

Algorithm 1 details the main algorithm implementing the approach described here.

### 3.3 Label Clustering

The labels are clustered based on the correlation matrix Cor, which is a square \(|L| \times |L|\) matrix that contains the Pearson correlation coefficients \(cor_{jk}\) between pair of labels \(l_j\) and \(l_k\). As labels are binary in the multi-label setting, the Pearson coefficient is reduced to the Phi coefficient, a well-known measure of association between two binary (dichotomous) variables [3, 24]. Whenever the absolute value of correlation between two labels is greater than a threshold \(\lambda_D\) calculated from the number of instances (derivation given below), we decide that these labels are correlated and should be in the same cluster.

Algorithmically, this can be achieved by single linkage agglomerative hierarchical clustering as shown in Algorithm 2. The algorithm starts by creating a separate cluster for each label. It then proceeds...
Algorithm 1 LaCovaC(D,L): Learn a tree-based multi-label classifier from training data.

Input: Dataset $D$; Labelset $L$; Minimum number $m$ of instances to split
Output: Tree-based multi-label classifier
if Labels are pure or $|D| < m$ then
    Return Leaf with relative frequencies of labelsets, predicted labelset that has the highest probability
else
    $\text{clust}=\text{CLUST}(D,L)$
    if $|\text{clust}| > 1$ then
        for each set $s$ in $\text{clust}$ do
            $D_s = \text{Split } D \text{ vertically based on the label cluster } s$ /* Learn a decision tree for the label cluster $s$ */
            $T_s = \text{LaCovaC}(D_s,s)$.
        end for
        Return Node splitting on $\text{clust}$ with subtrees $T_s$
    end if
    $f, \{D_i\} = \text{SelectBestFeature}(D)$
    for each child node $D_i$ do
        $T_i = \text{LaCovaC}(D_i,L)$
    end for
    Return Node splitting on $f$ with subtrees $T_i$
end if

Algorithm 2 CLUST(D,L): Cluster labels based on the correlation matrix

Input: Dataset $D$; a set of labels $L = l_1, \ldots, l_L$;
Output: $\text{currClust}$ - the clusters of labels
/* Compute the Pearson correlation coefficients between each pair of labels. */
$\text{Cor} = \text{Correlation Matrix}$
/* Compute the distance between each pair of labels using the absolute correlations. */
$\text{Dist} = 1 - |\text{Cor}|$
/* Build initial clusters with each label in a separate cluster. */
$\text{currClust} = \{\{l_1\}, \{l_2\}, \ldots, \{l_L\}\}$
/* Create a list of label pairs sorted in ascending order of distance value. */
$\text{pairList} = \text{all pairs of labels } (l_j, l_k) \text{ with } \text{dist}_{jk} < 1 - \lambda_{ij}p_{ij}, \text{sorted by } \text{dist}_{jk} \text{ ascendingly}$
for each label pair $(l_j, l_k)$ in $\text{pairList}$ do
    if labels $l_j$ and $l_k$ are in different clusters then
        merge clusters containing $l_j$ and $l_k$ within $\text{currClust}$
    end if
end for
Return $\text{currClust}$

We now derive the threshold $\lambda$ to decide whether a pair of labels are correlated or not, before merging them into one cluster. The idea is to set this threshold equal to two standard deviations in the distribution of correlation under the assumption of independent labels, hence enclosing a 95% confidence interval. The threshold depends on $n$, which is the number of instances reaching the current node in the tree. To derive the threshold we consider two labels with empirical frequencies $p_j$ and $p_k$, respectively. The empirical Pearson correlation between these labels as Bernoulli variables can be calculated as

\[
\hat{\text{cor}}_{jk} \approx \frac{C_{jk}/n - p_jp_k}{\sqrt{p_jp_k(1-p_j)(1-p_k)}},
\]

where $C_{jk}$ is the number of instances with both of those labels, and hence $C_{jk}/n$ is the proportion of such instances. The terms $p_j(1-p_j)$ and $p_k(1-p_k)$ in the denominator are the variances of these Bernoulli variables.

Next, we study the distribution of $\hat{\text{cor}}_{jk}$ under the assumption that labels $l_j$ and $l_k$ are independent. This distribution can be generated by randomly reassigning one of those labels between the instances, keeping the total frequency constant. In such case $C_{jk}$ has a hypergeometric distribution, and we can then directly calculate its mean $\mu$ and variance $\sigma^2$:

\[
C_{jk} \sim \text{Hypergeometric}(np_j, np_k, n)
\]

\[
\mu = np_jp_k
\]

\[
\sigma^2 = \frac{n^2}{n-1}p_jp_k(1-p_j)(1-p_k)
\]
From this we can calculate the mean and variance of the correlation $\text{cor}_{jk}$ between labels $l_j$ and $l_k$, as follows:

$$\text{mean}(\text{cor}_{jk}) = \frac{n p_j p_k / n - p_j p_k}{p_j p_k (1 - p_j) (1 - p_k)} = 0$$

$$\text{var}(\text{cor}_{jk}) = \frac{\sum (p_j p_k (1 - p_j) (1 - p_k))/n^2}{n} = 1 \quad (n - 1)$$

We define $\lambda_n$ as the value enclosing a 95% confidence interval assuming a normal distribution for $\text{cor}_{jk}$ which can be calculated in the usual way:

$$\lambda_n = 1.96 \cdot \sqrt{\text{var}(\text{cor}_{jk})} = \frac{1.96}{\sqrt{n - 1}}$$

(2)

where $n$ is the number of instances reaching a particular decision node in the tree. Note that this threshold is always less than 1 as $n > 5$ in our experiments.

4 Experimental Evaluation

In total, 13 common benchmarks were retrieved for use from the Meka\(^5\) and Mulan [22] repositories. The key statistics of these datasets are shown in Table 2.

Table 2: The statistics for the datasets used in the experiments. $|L|$ is the number of labels, $|D|$ is the number of instances, $\text{att}$ is the number of attributes (features), $\text{card}$ is the average number of labels per instance, and $\text{dens}$ is the label density.

| Dataset  | $|L|$ | $|D|$ | $\text{att}$ | $\text{card}$ | $\text{dens}$ |
|----------|------|------|-------------|-------------|-------------|
| Corel5k  | 574  | 5000 | 499        | 3.522       | 0.94%       |
| Cal500   | 174  | 502  | 68         | 26.044      | 14.96%      |
| Bibtext  | 159  | 7395 | 1836       | 2.402       | 1.51%       |
| Language log | 75  | 1460 | 1004       | 1.797       | 1.57%       |
| Enron    | 53   | 1702 | 1001       | 3.378       | 6.37%       |
| Medical  | 45   | 978  | 1449       | 1.245       | 2.76%       |
| Genebase | 27   | 662  | 1186       | 1.252       | 4.63%       |
| Slashdot | 22   | 3762 | 1079       | 1.180       | 5.36%       |
| Birds    | 19   | 645  | 260        | 1.014       | 5.36%       |
| Yeast    | 14   | 2417 | 103        | 4.237       | 30.26%      |
| Flags    | 7    | 194  | 19         | 3.392       | 48.45%      |
| Emotions | 6    | 593  | 66         | 1.869       | 31.14%      |
| Scene    | 6    | 2407 | 294        | 1.074       | 17.89%      |

4.1 Experimental Setup

BR, LP, and CC algorithms were run in Meka, whereas Mulan was used for the Homer and LPBR algorithms. When employing all these methods, the trees were produced with Weka’s J48 algorithm\(^6\). The Homer algorithm was run using the best setting, as reported by [21]. LPBR requires parameter configurations, such as non-improving counter to prevent clustering. The default parameters settings in Mulan were 5 for the non-improving counter and a 5-fold cross validation for testing the clustering performance. The target loss function to evaluate the clustering was set to exact-match according to Mulan. Exact-match is a strict measure that examines whether the predicted and actual label sets are equal or not.

\(^5\) http://meka.sourceforge.net/

\(^6\) http://sourceforge.net/projects/weka

LaCovaC was implemented in Java on the Meka platform. The LaCova and ML-C4.5 implementations are provided in [2]. The minimal number of instances in the leaves in ML-C4.5, LaCova, and LaCovaC was set to 5.

10-fold cross-validation was performed throughout except the two largest datasets in terms of both number of labels and number of instances: Corel5k and Bibtext. The exception is because cross validation was intensive for LPBR as we were not able to run this algorithm using the available resources. Therefore, a train/test split was used instead, which was provided by Mulan.

For all methods we used the Pcut method to convert the label probabilities into relevant labels [1, 25]. This method uses a single threshold for all labels, chosen such that the average label density in test data is the same as in the training data.

4.2 Evaluation Metrics

All algorithms were evaluated under the Meka framework. We used the most common multi-label metrics, namely multi-label accuracy (Jaccard index), exact-match (subset accuracy), Hamming loss and three versions of the $F_1$ score: micro-averaged over all instance-label combinations, macro-averaged per label, and macro-averaged per instance. We also used a multi-label version of log-loss [16] to evaluate the predicted probabilities. Log-loss evaluates a classifier’s confidence by punishing errors with higher probability more severely.

We used a dataset-dependent maximum of $\log(\frac{1}{|D|})$ to limit the magnitude of penalty as suggested in [16]. The description and formal definition of these evaluation measures for multi-label data are given below.

**Multi-label accuracy** calculates the ratio of the union and intersection of the predicted and actual labelsets, averaged over all examples [28]:

$$\text{mla} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|}$$

(3)

where $Y_i$ and $\hat{Y}_i$ are the set of actual labels and predicted labels for an instance, respectively.

**Exact-match** examines if the predicted and actual label subsets are equal or not [9]:

$$\text{em} = \frac{1}{|D|} \sum_{i=1}^{|D|} I(Y_i = \hat{Y}_i)$$

(4)

where $I(true) = 1$ and $I(false) = 0$.

**Hamming loss** takes the proportion of misclassified labels (labels predicted incorrectly and correct labels that are not predicted) averaged over all examples [18]:

$$\text{hl} = \frac{1}{|L| \cdot |D|} \sum_{i=1}^{|D|} |Y_i \triangle \hat{Y}_i|$$

(5)

where $\triangle$ is the symmetric difference.

\(^7\) 2.7 GHz Intel Core i5 processor and 8GB of memory
Log-loss evaluates a classifier’s confidence by punishing errors with higher probability more severely [16]:

\[ L_s = \frac{1}{|D|} \sum_{i=1}^{|D|} \sum_{j=1}^{|L|} \log(pr_{ij}) \cdot y_{ij} + \log(1 - pr_{ij}) \cdot (1 - y_{ij}) \]  

(6)

where \( y_{ij} \) indicates whether the \( j^{th} \) label is relevant for the \( i^{th} \) instance (value 1) or irrelevant (value 0). \( pr_{ij} \) is the probability estimate for the \( i^{th} \) instance and the \( j^{th} \) label.

Micro \( f_1 \) corresponds to the harmonic mean of precision and recall [14]. It put all predictions on all labels in one vector as in binary classification and then calculates the \( f_1 \):

\[
\text{micro } f_1 = \frac{\sum_{i=1}^{|D|} \sum_{j=1}^{|L|} 2 \cdot y_{ij} \cdot \hat{y}_{ij}}{\sum_{i=1}^{|D|} \sum_{j=1}^{|L|} (y_{ij} + \hat{y}_{ij})}
\]  

(7)

Macro \( f_1 \) is the averaged \( f_1 \) score over labels [14]:

\[
\text{macro } f_1 = \frac{1}{|L|} \sum_{j=1}^{|L|} F_1^{(j)}
\]  

(8)

where \( F_1^{(j)} \) is the \( f_1 \) score for the \( j^{th} \) label vector.

Macro \( f_e \) is the averaged \( f_1 \) score over examples [14]:

\[
\text{macro } f_e = \frac{1}{|D|} \sum_{i=1}^{|D|} F_1^i
\]  

(9)

where \( F_1^i \) is the \( f_1 \) score for the \( i^{th} \) instance row vector.

4.3 Results and Discussion

We conducted the Friedman test based on the average ranks for all datasets [8]. This test ranks the algorithms for each dataset separately, thus the best algorithm gets the rank of 1, the second best the rank of 2, etc. Then, it calculates the test statistic on the ranks averaged over all datasets in order to verify whether the differences between algorithms are statistically significant.

Table 3 shows the average ranks for each algorithm over 13 datasets and seven evaluation measures. We also show the mean of these average ranks aggregated over all evaluation measures, which shows that overall LaCovaC scores highest, followed by CC and BR. The Friedman test gave a significant difference at 5% confidence for all metrics except multi-label accuracy; therefore, we carried out post-hoc Nemenyi tests as shown in Figure 3.

It can be seen that LaCovaC outperforms Homer, ML-C4.5, and LPBR in the average ranks with respect to all the evaluation measures used in this paper, and in several cases statistically significantly. LaCovaC is not significantly worse than the best performing algorithm (LP) in terms of exact-match. In fact, according to Table 4 LaCovaC loses to LP in the 4 datasets with the smallest numbers of labels (up to 14), but wins in 6 out of the 9 datasets with more labels. This fact confirms the assumption that LP can overfit the training data in case of large number of labels and label combinations because it can only model labelsets observed in the training. Notably, for exact-match the proposed algorithm LaCovaC is the overall best method or tied with the best in the three datasets with the biggest numbers of labels (Corel5k, Cal500 and Bibtex with 374, 174 and 159 labels, respectively). The proposed algorithm has better Hamming loss than LP in 9 out of 13 datasets.

Additionally, LaCovaC outperforms BR in terms of exact-match, log-loss and Hamming loss. BR has the best average rank considering \( f_1 \) score per instance average \( f_e \), \( f_1 \) per label average \( f_1 \) and \( f_e \) as it decomposes the multi-label problem into several binary classifiers, which can be better for \( f_1 \) score.

We can further observe that LaCovaC has the best average rank in terms of log-loss that evaluates the classifiers’ scores regardless of the thresholding technique. It is significantly better than CC, ML-C4.5, Homer and LP.

Finally, CC gets top average rank for Hamming loss and LaCovaC is the second best without significant loss against CC. Detailed results of the experiments are given in Tables 4, 5 and 6.

Table 3: Average ranks obtained by the Friedman test over 13 datasets. The last column shows the mean of the average ranks obtained by each algorithm over all the evaluation measures, and the algorithms are sorted on decreasing mean. \( \text{mla} \), \( \text{em} \), \( \text{hl} \) and \( \text{ls} \) denote multi-label accuracy, exact-match, Hamming loss and log-loss, respectively. \( \text{micro } f_1 \) and \( \text{macro } f_1 \) are \( f_1 \) scores averaged in terms of labels and examples, respectively.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( \text{mla} )</th>
<th>( \text{em} )</th>
<th>( \text{hl} )</th>
<th>( \text{ls} )</th>
<th>( \text{micro } f_1 )</th>
<th>( \text{macro } f_1 )</th>
<th>( \text{macro } f_e )</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaCovaC</td>
<td>3.57</td>
<td>2.69</td>
<td>2.73</td>
<td>2.23</td>
<td>3.88</td>
<td>4.34</td>
<td>3.96</td>
<td>3.34</td>
</tr>
<tr>
<td>CC</td>
<td>3.46</td>
<td>2.16</td>
<td>5.19</td>
<td>3.38</td>
<td>4.19</td>
<td>3.69</td>
<td>4.39</td>
<td>3.76</td>
</tr>
<tr>
<td>BR</td>
<td>4.57</td>
<td>6.8</td>
<td>4.8</td>
<td>2.3</td>
<td>2.69</td>
<td>1.92</td>
<td>3.26</td>
<td>3.87</td>
</tr>
<tr>
<td>LaCovaC</td>
<td>3.65</td>
<td>5.46</td>
<td>5.03</td>
<td>3.38</td>
<td>3.69</td>
<td>4.15</td>
<td>3.46</td>
<td>4.12</td>
</tr>
<tr>
<td>LP</td>
<td>4.92</td>
<td>2.42</td>
<td>4.34</td>
<td>6.64</td>
<td>6</td>
<td>5.57</td>
<td>5.61</td>
<td>5.05</td>
</tr>
<tr>
<td>LPBR</td>
<td>4.84</td>
<td>6.41</td>
<td>6.57</td>
<td>4.88</td>
<td>5.38</td>
<td>4.53</td>
<td>4.8</td>
<td>5.09</td>
</tr>
<tr>
<td>Homer</td>
<td>5.3</td>
<td>6.03</td>
<td>6.48</td>
<td>6.07</td>
<td>4.61</td>
<td>4.61</td>
<td>5.15</td>
<td>5.24</td>
</tr>
<tr>
<td>ML-C4.5</td>
<td>5.65</td>
<td>5.15</td>
<td>5.92</td>
<td>5.46</td>
<td>6.34</td>
<td>6.65</td>
<td>5.76</td>
<td>5.85</td>
</tr>
</tbody>
</table>

Figure 3: Critical Difference diagrams using pairwise comparisons for experiments where the Friedman test yields significance at 0.05.
Table 4: Results on 13 datasets with regards to multi-label accuracy and exact-match, the higher the value the better.

<table>
<thead>
<tr>
<th>BR</th>
<th>LP</th>
<th>C.C.</th>
<th>Homer</th>
<th>L.P.B.R.</th>
<th>ML-C1.5</th>
<th>LaCova</th>
<th>LaCovat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>0.391(6,5)</td>
<td>0.396(4,5)</td>
<td>0.397(3)</td>
<td>0.381(8)</td>
<td>0.411(5)</td>
<td>0.416(1)</td>
<td>0.391(6,5)</td>
</tr>
<tr>
<td>Flags</td>
<td>0.534(6)</td>
<td>0.531(7)</td>
<td>0.540(5)</td>
<td>0.544(4)</td>
<td>0.506(8)</td>
<td>0.572(1)</td>
<td>0.558(3)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.402(4)</td>
<td>0.416(2)</td>
<td>0.388(7)</td>
<td>0.390(6)</td>
<td>0.417(1)</td>
<td>0.406(3)</td>
<td>0.400(5)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.403(6)</td>
<td>0.432(5,2)</td>
<td>0.464(1)</td>
<td>0.380(7)</td>
<td>0.421(5)</td>
<td>0.366(8)</td>
<td>0.432(2,5)</td>
</tr>
<tr>
<td>Average ranks</td>
<td>4.87</td>
<td>4.93</td>
<td>4.97</td>
<td>4.98</td>
<td>4.95</td>
<td>4.92</td>
<td>5.09</td>
</tr>
</tbody>
</table>

Table 5: Results on 13 datasets with regards to Hamming loss and log-loss, the lower the value the better.

<table>
<thead>
<tr>
<th>BR</th>
<th>LP</th>
<th>C.C.</th>
<th>Homer</th>
<th>L.P.B.R.</th>
<th>ML-C1.5</th>
<th>LaCova</th>
<th>LaCovat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>0.391(6,5)</td>
<td>0.396(4,5)</td>
<td>0.397(3)</td>
<td>0.381(8)</td>
<td>0.411(5)</td>
<td>0.416(1)</td>
<td>0.391(6,5)</td>
</tr>
<tr>
<td>Flags</td>
<td>0.534(6)</td>
<td>0.531(7)</td>
<td>0.540(5)</td>
<td>0.544(4)</td>
<td>0.506(8)</td>
<td>0.572(1)</td>
<td>0.558(3)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.402(4)</td>
<td>0.416(2)</td>
<td>0.388(7)</td>
<td>0.390(6)</td>
<td>0.417(1)</td>
<td>0.406(3)</td>
<td>0.400(5)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.403(6)</td>
<td>0.432(5,2)</td>
<td>0.464(1)</td>
<td>0.380(7)</td>
<td>0.421(5)</td>
<td>0.366(8)</td>
<td>0.432(2,5)</td>
</tr>
<tr>
<td>Average ranks</td>
<td>4.87</td>
<td>4.93</td>
<td>4.97</td>
<td>4.98</td>
<td>4.95</td>
<td>4.92</td>
<td>5.09</td>
</tr>
</tbody>
</table>
5 Concluding Remarks

We presented a novel decision tree algorithm for multi-label classification called LaCovaC. The key idea of this algorithm is to compute the label correlation matrix at each node of the tree in order to identify label correlations and then cluster them locally. Its main innovation is to introduce vertical splits that separate locally independent labels, in addition to the traditional feature-based horizontal splits that divide the instance space as in the standard decision tree algorithms.

To evaluate LaCovaC we compared it to state-of-the-art approaches. We used seven common evaluation metrics and 13 datasets. LaCovaC has the best average rank for log-loss and the second best for multi-label accuracy, exact-match and Hamming loss (without significant loss). For exact-match, LaCovaC shows a comparable performance to the best algorithm LP, which is a strong baseline for exact-match and hard to beat. In addition, it outperforms LP on 6 of the 9 largest datasets in terms of number of labels. This suggests that combining LP and BR as in LaCovaC leads to improvement over other approaches. We used seven common evaluation metrics and 13 datasets. LaCovaC has the best average rank for log-loss and the second best for multi-label accuracy, exact-match and Hamming loss (without significant loss). For exact-match, LaCovaC shows a comparable performance to the best algorithm LP, which is a strong baseline for exact-match and hard to beat. In addition, it outperforms LP on 6 of the 9 largest datasets in terms of number of labels. This suggests that combining LP and BR as in LaCovaC leads to improvement over other approaches.

Overall, the main strength of LaCovaC is its strong performance across all these metrics, as can be seen in the right-most column of Table 3. Furthermore, on each of the metrics individually LaCovaC is not significantly worse than the top contender. It is widely recognised in the literature that different multi-label evaluation metrics measure different things; hence there is value in demonstrating that an algorithm performs well across the board.

Several directions can be taken for further work. We plan to investigate the parameter configuration for LaCovaC, for example, a stopping criterion for the clustering algorithm. Moreover, this would be interesting to investigate alternative clustering approaches, such as spectral clustering. Furthermore, to counteract the large variance associated with decision tree learning – which carries over to the proposed model – we could use bootstrap aggregates as in random forests. Finally, using the significance threshold on correlation balances the sample size and the strength of the correlation, such that both low correlation on a large sample size and high correlation on a small sample are detected as significant. However, finding an even better balance between effect size and sample size is an interesting future research direction.

Acknowledgments

Reem Al-Otaibi’s PhD study is sponsored by King Abdulaziz University, Saudi Arabia. This work was partly supported by the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences and Technologies ERA-Net (CHISTERA), and funded by the Engineering and Physical Sciences Research Council in the UK under grant EPSRC EP/K018728/1. We acknowledge the comments of the anonymous reviewers which helped us improve the paper.

REFERENCES

Learning and Application (ICMLA-2014), pp. 74–79, Detroit, USA, (December 2014). IEEE.


