ABC-based stacking method for multilabel classification

Weimin DING1,2,* , Shengli WU1,3 ©
1School of Computer Science, Jiangsu University, Zhenjiang, P.R. China
2School Of Mathematics and Information Science, Weifang University, Weifang, P.R. China
3School of Computing, Ulster University, Newtownabbey, UK

Received: 28.02.2019 • Accepted/Published Online: 08.07.2019 • Final Version: 26.11.2019

Abstract: Multilabel classification is a supervised learning problem wherein each individual instance is associated with multiple labels. Ensemble methods are effective in managing multilabel classification problems by creating a set of accurate, diverse classifiers and then combining their outputs to produce classifications. This paper presents a novel stacking-based ensemble algorithm, ABC-based stacking, for multilabel classification. The artificial bee colony algorithm, along with a single-layer artificial neural network, is used to find suitable meta-level classifier configurations. The optimization goal of the meta-level classifier is to maximize the average accuracy of classification of all the instances involved. We run an experiment on 10 benchmark datasets of varying domains and compare the proposed approach to five other ensemble algorithms to demonstrate the feasibility and effectiveness of ABC-based stacking.

Key words: Ensemble learning, stacking, artificial bee colony, cross-entropy, multilabel learning

1. Introduction

The problem of learning from multilabel data affects music, images, text, data streams, and many other applications [1–4]. Multilabel classification involves learning a mapping from an instance to a set of labels [5]. Problem transformation is a common approach to multilabel classification wherein predictions are gathered from a group of single-label classifiers and transformed into multilabel predictions [6]. There are many problem transformation methods, such as binary relevance (BR), label power-set (LP), and pair-wise (PW). The BR method does not consider the correlations between different labels of each instance and its predictive performance can be weak. The drawback of the LP method is that its space of possible label subsets can be very large and it has a tendency to overfit. PW is usually intractable for large-scale problems due to its quadratic complexity. Algorithm adaptation methods are able to handle multilabel data directly by extending an existing algorithm [7] such as neural networks in BP-MLL and the Bayesian network in LEAD. These methods are mostly tailored to a specific classifier and thus lack generality. Ensemble methods have been shown to improve the generalization ability of learning systems [8]; they are extensions of problem transformation or algorithm adaptation techniques.

The best-known ensembles of problem transformation are RAkEL [9], ensembles of classifier chains (ECC) [6], and ensembles of pruned sets (EPS) [10]. In these methods, a set of single-label base classifiers is used to make predictions, and then they are combined by a multilabel classifier. Ensemble methods of algorithm adaptation use ensembles with base classifiers that are themselves algorithm adaptations. Usually, these ensembles encompass a set of multilabel classifiers as base classifiers on samples with replacement or feature

*Correspondence: dwm007dwm007@126.com

This work is licensed under a Creative Commons Attribution 4.0 International License.
selections; the multilabel predictions of these base classifiers are combined via a voting scheme [8]. Ensemble methods of algorithm adaptation include ML-FOREST [11] and RF-PCT [12]. As a departure from previously published ensemble methods, we develop a novel stacking-based ensemble method for multilabel classification in this study.

Stacking is an ensemble technique that employs a two-level structure: base level and meta-level [13]. At the base level, multiple different types of classifiers are chosen to produce meta-level instances. At the meta-level, a meta-classifier is applied to map the predictions of the base-level classifiers to the final class labels. If used properly, stacking can be more effective than an individual classifier-based approach [14–16]. When applying a stacking method we need to consider two problems: 1) selection of multiple accurate and diverse base-level classifiers, and 2) construction of a suitable meta-classifier [17]. For the former question, forward selection [18], backward selection [18], the genetic algorithm [19], ant colony optimization [20], particle swarm optimization [21, 22], and many other approaches [17] can be used. Regarding the meta-classifier, objective optimization-based weighting functions are particularly effective in stacking-based ensembles [4, 23]. For example, GOOWE-ML is a weighted ensemble method for multilabel predictions that trains the weight by minimizing the Euclidean distance between the estimated relevance scores and ground truth [4].

In this paper, we propose a novel stacking-based ensemble algorithm, ABC-based stacking (artificial bee colony-stacking), to address the issue of multilabel classification. In ABC-based stacking, we use the artificial bee colony algorithm for searching the base classifier configuration and a single-layer artificial neural network as the meta-level classifier. The selection of the base classifiers and the corresponding parameter values of the ANN are determined by the optimization goal of maximizing the average accuracy of classification of all the instances involved. Specially, the stochastic gradient descent algorithm is used to train the weights of the neural network. We evaluate the performance of this technique by conducting an experiment on 10 datasets in different domains. The results show that ABC-based stacking is highly competitive compared with several state-of-the-art ensemble approaches.

The rest of this paper is organized as follows. Section 2 presents the proposed methodology. Section 3 reports the experimental setup and results. Section 4 is the conclusion.

2. The proposed methodology

ABC is one of the swarm intelligence algorithms and was proposed by Karaboga in 2005 [24]. It can solve optimization problems by simulating the intelligent foraging behavior of honey bees. The standard ABC model [25–27] consists of “food sources” and three types of “bees”: employed bees, onlooker bees, and scout bees. The food sources are possible solutions to the optimization problem. The “nectar content” of the food sources corresponds to the fitness of the solutions. The search process starts when employed bees leave the hive to search for food sources. Upon finding nectar, the employed bees store it and then return to the hive. They share their information about the food sources through “dancing” and recruit new onlooker bees to explore the most rich food sources. Once the onlooker bees choose a food source to explore, they become employed bees. Each employed bee is assigned a food source; the number of employed bees equals the number of onlooker bees. If the position of a food source is not updated through predetermined cycles, the employed bee becomes a scout bee and the previous food position is replaced by a new one.
2.1. ABC-based stacking for multilabel classification

ABC-based stacking is a multilabel classification method that employs the ABC algorithm along with a single-layer artificial neural network to configure an optimal stacking ensemble. In ABC-based stacking, decision variables are food sources represented by bit vectors. Elements in a bit vector are limited to 0 or 1 and are related to the selection of base-level classifiers in ABC-based stacking. Figure 1 shows an example of the food source evolutionary process.

Figure 1. An example of the food source evolutionary process.

Figure 1 shows that food source \( U_1 \) evolves into 4 neighboring food sources, \( V_i \) \((1 \leq i \leq 4)\). Each of the food sources is represented by a bit vector. For example, the second element of \( U_1 \) is 1, which means that the corresponding base classifier is chosen. At the next stage, all four vectors \( V_i \) \((1 \leq i \leq 4)\) have one more base classifier than \( U_1 \) does, although they are different from each other. Note that each food source, or bit vector, is a possible solution to our problem with proper weights defined for each of the base classifiers.

All individual steps of ABC-based stacking are given as follows.

At the initial stage, \( SN \) base classifiers are built and a learning method, \( f \), which aims to optimize cross-entropy, is set as the meta-classifier. Next, the learning algorithm initializes \( SN \) food sources \( U_i \) \((1 \leq i \leq SN)\), where \( SN \) is equal to the number of base level classifiers. Each food source is initialized with a bit vector of size \( SN \), to which an employed bee is assigned. Each food source only consists of one base classifier. The fitness of any food source \( U_i \) can be evaluated by Eq. 1:

\[
fit(U_i) = \frac{1}{1 + \text{Cost}(U_i)},
\]

where the value of \( \text{Cost}(U_i) \) is defined as the hamming loss and can be calculated by Eq. 2:

\[
\text{Cost}(U_i) = \frac{1}{M} \sum_{j=1}^{M} \frac{1}{L} \sum_{l=1}^{L} \mathbb{I}_{g^l_j(f(x_j)) \neq y^l_j},
\]

where \( L \) represents the number of class labels and \( M \) represents the number of instances considered. \( \mathbb{I}_{true} \) returns 1 and \( \mathbb{I}_{false} \) returns 0. \( f(\cdot) \) is the meta-level classifier and \( g(\cdot) \) is a threshold function applied to find a bipartition of relevant and irrelevant labels. \( y^l_j \) and \( g^l_j(f(x_j)) \) denote the the \( l \)th true label and predicted label of instance \( x_j \), respectively. \( \text{Cost}(U_i) \) computes the average percentage of labels of \( M \) instances whose relevance is wrongly predicted.

At the employed bee stage, each employed bee explores the neighborhood and adds one more base classifier.
to \( U_i \). That is, the employed bee generates a new candidate solution \( V_i = \{v_{i1}, v_{i2}, \ldots, v_{iSN}\} \) using Eq. 3:

\[
v_{ij} = \begin{cases} u_{ij} & j \neq k \\ 1 & j = k, \end{cases}
\]

(3)

where \( k \) is a randomly chosen index from the set \( \{1, 2, \ldots, SN\} \) and satisfies the condition that \( u_{ik} = 0 \). \( V_i \) is a new configuration of base classifiers. A greedy selection mechanism is applied between the fitness of \( V_i \) and \( U_i \): if the fitness score of \( V_i \) is greater than that of \( U_i \), the employed bee is assigned the new food source \( V_i \); otherwise, the employed bee keeps \( U_i \) unchanged.

In the onlooker bee phrase, the food source with the best probability (Eq. 4) to be explored is selected by the onlooker bees. Similarly, a greedy selection mechanism is also applied by the onlooker bees:

\[
prob_i = \frac{fit(U_i)}{\sum_{i=1}^{SN} fit(U_i)}.
\]

(4)

In the scout bee phase, ABC-based stacking checks if there are any food sources to be abandoned; if the food source has not been updated within a given max limit, it is abandoned and replaced with a new food source with only one base classifier.

The proposed algorithm employs three types of bees to explore the suitable subset of base classifiers. The final base classifier configuration is achieved when the entire search process is completed. The pseudocode of the ABC-based stacking algorithm is shown in Algorithm 1.

2.2. Cross-entropy-based meta-level classifier

Classification accuracy in multilabel learning can be evaluated as the similarity of the true labels and the predictions by the defined meta-level classifier for all the instances considered. Many objective functions can be used to measure said similarity, such as the Hilbert–Schmidt independence criterion [28] or Euclidean distance [1]. In ABC-based stacking, we use cross-entropy as the loss function to measure it (Eq. 5). The learning objective is formalized as the minimization of cross-entropy over all the instances:

\[
J(p, q) = -\sum_{j=1}^{L} p_j \log q_j,
\]

(5)

where \( L \) represents the number of class labels. Two probability distributions, \( p = (p_1, p_2, \ldots, p_L) \) and \( q = (q_1, q_2, \ldots, q_L) \), are generated from the ground truth label vector and the predicted score vector using Eq. 6 and Eq. 7, respectively:

\[
p_j = \frac{\exp(R_j)}{\sum_{t=1}^{L} \exp(R_t)},
\]

(6)

\[
q_j = \frac{\exp(score_j)}{\sum_{t=1}^{L} \exp(score_t)},
\]

(7)

where \( R = \{R_1, R_2, \ldots, R_L\} \) is the label judgment vector of the given meta-level instance. The value of \( R_j \) is 1 if the label \( L_j \) is true and 0 otherwise. \( S = \{score_1, score_2, \ldots, score_L\} \) is the score vector predicted by the meta-level classifier. The value of \( score_j \) is the score predicted for the \( j \)th class label.
Algorithm 1. Pseudocode for the ABC algorithm in searching base classifier configuration.

Input:

- $f$: Meta-level classifier;
- $SN$: Number of food sources;
- $MAX\_LIMIT$: Max limit that decides if the food source needs to be updated;
- $Max\_Cycle$: Maximum number of iterations;

Output:

- $C_1, C_2, \ldots, C_N$: Suitable subset of base classifiers; 

{ 
    Initialize $SN$ food sources and build $SN$ base classifiers;
    Each food source only consists of one unique base classifier;
    Assign an initial solution to each employed bee and compute its corresponding fitness using Eq. 1;
    $Cycle = 1$;
    Repeat
        Employed bee phase
        For $i$ from 1 to $SN$, for each employed bee
        { 
            Employed bee associated with $U_i$ explores the neighborhood and produces new solution $V_i$
            using Eq. 3;
            Compute $Cost(U_i)$ based on the output of the meta-level classifier $f$ (see Algorithm 2);
            Evaluate the fitness value of the solution $V_i$ using Eq. 1;
            If the fitness of $V_i$ is greater than that of $U_i$, then update $U_i$ with $V_i$;
        } 
        Calculate the probability $prob_i$ of each solution using Eq. 4;
        Onlooker Bee Phase
        For $j$ from 1 to $SN$, for each onlooker bee
        { 
            Select a food source $U_j$ depending on the value of $prob_j$;
            Produce new solution $V_j$ using Eq. 3;
            Compute the objective value and fitness of $V_j$;
            If the fitness of $V_j$ is greater, then update $U_j$ with the new solution $V_j$;
        } 
        Scout bee phase
        Exhausted sources are replaced with new food sources discovered by the scout bee if its
        $LIMIT$ is greater than that of the $MAX\_LIMIT$;
        $cycle = cycle + 1$;
        Memorize the best food source;
    Until $Max\_Cycle$ is reached;
    return (the searched base classifiers $C_1, C_2, \ldots, C_N$); 
}

Here, we define a weight function formalized as the meta-level classifier as follows:

$$score_j = f\left(\sum_{k=1}^{N} w_k \times s_{kj}\right),$$  \hspace{1cm} (8)$$

where $f$ is the meta-level classifier and $N$ is the number of searched base classifiers by the proposed ABC
algorithm. The value of $s_{kj}$ represents the confidence of the training instance belonging to class $L_j$ predicted by base classifier $C_k$. For each instance, we use the $s_{kj}$ ($k \in 1, 2, ..., N$) predicted by $N$ searched base classifiers for the same class label $L_j$ as an input to calculate $score_j$. The training of $f$ serves to reveal the optimum weight vector that makes all meta-level data achieve cross-entropy minimization. In Eq. 9, the confidences $s_{kj}$ ($j \in 1, 2, ..., L$) predicted by the same base classifier for a given instance are normalized beforehand into the range of [0,1] as follows:

$$s_{kj} = \frac{s_{kj}}{\sum_{j=1}^{L} s_{kj}}.$$  \hspace{1cm} (9)

For simplicity, we use a single-layer ANN model as the meta-level classifier:

$$score_j = f(\sum_{k=1}^{N} w_k \times s_{kj}) = \frac{1}{1 + e^{-(\sum_{k=1}^{N} w_k \times s_{kj})}}.$$  \hspace{1cm} (10)

To obtain function $f$, we apply the stochastic gradient descent algorithm to determine the optimal weights $W = (w_0, w_1, ..., w_N)$. The weight gradient is defined as follows:

$$\nabla W = (\nabla w_0, \nabla w_1, ..., \nabla w_N).$$

Because

$$\frac{\partial f}{\partial w_k} = f(\sum_{k=1}^{N} w_k s_{kj}) \times (1 - f(\sum_{k=1}^{N} w_k s_{kj})) \times s_{kj}$$  \hspace{1cm} (11)

$$\text{then} \quad \nabla w_k = \frac{\partial J(p, q)}{\partial w_k} = \sum_{j=1}^{L} (q_j - p_j) \times score_j \times (1 - score_j) \times s_{kj},$$  \hspace{1cm} (13)

Eq. 13 can be used to obtain the optimal weight.

Algorithm 2 presents the learning process of the ANN. As an input, we need to provide a meta-level training dataset, an initial ANN, and four parameters (iteration number $T$, learning rate $\eta$, number of class labels $L$, and number of chosen base classifiers $N$). The output of the algorithm is the ANN with learned weights. The subfunction $Label\_judgment()$ is used to give the label judgment vector of the given meta-level instance. The subfunction $Normalize()$ is used to normalize the confidences predicted by the searched base classifier. Two subfunctions, $PD\_compute()$ and $SD\_compute()$, are used to calculate probability distributions $p$ and $q$, respectively. The algorithm mainly comprises a nested loop of three levels. The weights are updated for all the instances involved for a given number of times defined by $T$.

For a testing instance $x$ as an input, ABC-based stacking combines the outputs of the searched base classifiers and outputs a score vector $S = \{score_1, score_2, ..., score_L\}$ to quantify the confidence that the instance $x$ belongs to the class labels at hand. In this paper, we use a bipartition of relevant and irrelevant labels based on a threshold function $g_x(S)$ such that:

$$g_x^j(S) = \begin{cases} 1 & \text{if } score_j > t \\ 0 & \text{otherwise,} \end{cases}$$  \hspace{1cm} (14)
where \( g_{xj}(S) = 1 \), indicating the \( j \)th label is relevant regarding \( x \). \( t \in [0, 1] \) is a predefined threshold value and we set \( t = 0.5 \) for simplicity. With the function \( g_x(S) \), instance \( x \) is assigned to the predicted labels whose confidences are larger than the threshold value 0.5.

**Algorithm 2.** Pseudocode for the ANN learning process.

Input:

- \( M \): Meta-level dataset comprising confidences predicted by the chosen base classifiers;
- \( f \): ANN algorithm with the initial weights \( W \); each element in \( W \) is assigned a random number between \(-0.5\) and \(0.5\);
- \( T \): Iteration number;
- \( \eta \): Learning rate;
- \( L \): Number of class labels;
- \( N \): Number of chosen base classifiers;
- \( Label\_judgment() \): Obtain the label judgment vector of the given meta-level instance;
- \( Normalize() \): Normalize the confidences predicted by the same base classifier for a given instance;
- \( PD\_compute() \): Obtain probability distribution \( p \);
- \( SD\_compute() \): Obtain probability distribution \( q \);

Output:

- \( f \): Output the trained ANN;

```plaintext
for (int \( t := 1; t <= T; t++ \)) {
    for (int \( i := 1; i <= M.size; i++ \)) {
        for (int \( j := 1; j <= N; j++ \)) {
            Normalize(M.instance(i));
            // Normalize the confidences predicted by the same base classifier using Eq. 9;
        }
        for (int \( j := 1; j <= L; j++ \)) {
            score\(_j\) = f(M.instance(i));
            // The outputs of N chosen classifiers for label \( L\_j \) as the inputs of the ANN algorithm;
        }
        \( R = Label\_judgment(M.instance(i)); \)
        \( S = \{score\_1, score\_2, ..., score\_L\}; \)
        \( p = PD\_compute(R); \) // Compute probability distribution \( p \) using Eq. 6;
        \( q = SD\_compute(S); \) // Compute probability distribution \( q \) using Eq. 7;
        for (int \( k := 0; k <= L; k++ \)) { // Compute gradient vector \( \nabla w \);
            Compute gradient \( \nabla w_k \) using Eq. 13;
            Update \( w_k = w_k - \eta \times \nabla w_k \);
        }
    }
    return(f); // Output the trained ANN;
}
```
3. Experiments

3.1. Experimental setup

We used 10 multilabel datasets for our evaluation: Enron, Scene, Yeast, Medical, Emotions, Music, Mediamill, Image, VirusGO [29], and GpositivePseACC [30]. In Table 1, most datasets are originally split into two parts. For Mediamill, we produced training/test data with sizes of 1500/1000 via a reservoir sampling algorithm. For Music, the training data constitute 1/2 of the complete dataset and testing data constitute the remaining 1/2. All datasets come from five domains: biology, text categorization, music, image, and video.

Table 1. Multilabel datasets: training examples (#tr.e.) and test examples (#t.e.), feature quantity (#f.), label quantity(#l.), label cardinality (lc), and domain.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#tr.e.</th>
<th>#t.e.</th>
<th>#f.</th>
<th>#l.</th>
<th>lc</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron(^1)</td>
<td>1123</td>
<td>579</td>
<td>1001</td>
<td>53</td>
<td>3.378</td>
<td>Text</td>
</tr>
<tr>
<td>Scene(^1)</td>
<td>1211</td>
<td>1196</td>
<td>294</td>
<td>6</td>
<td>1.074</td>
<td>Image</td>
</tr>
<tr>
<td>Yeast(^1)</td>
<td>1500</td>
<td>917</td>
<td>103</td>
<td>14</td>
<td>4.237</td>
<td>Biology</td>
</tr>
<tr>
<td>Medical(^1)</td>
<td>333</td>
<td>645</td>
<td>1449</td>
<td>45</td>
<td>1.245</td>
<td>Text</td>
</tr>
<tr>
<td>Emotions(^1)</td>
<td>391</td>
<td>202</td>
<td>72</td>
<td>6</td>
<td>1.869</td>
<td>Music</td>
</tr>
<tr>
<td>Mediamill(^1)</td>
<td>1500</td>
<td>1000</td>
<td>120</td>
<td>101</td>
<td>4.376</td>
<td>Video</td>
</tr>
<tr>
<td>VirusGO(^2)</td>
<td>124</td>
<td>83</td>
<td>749</td>
<td>6</td>
<td>1.22</td>
<td>Biology</td>
</tr>
<tr>
<td>GpositivePseACC(^2)</td>
<td>311</td>
<td>208</td>
<td>440</td>
<td>4</td>
<td>1.01</td>
<td>Biology</td>
</tr>
<tr>
<td>Image(^3)</td>
<td>1200</td>
<td>800</td>
<td>294</td>
<td>5</td>
<td>1.236</td>
<td>Image</td>
</tr>
<tr>
<td>Music(^4)</td>
<td>296</td>
<td>296</td>
<td>71</td>
<td>6</td>
<td>1.783</td>
<td>Music</td>
</tr>
</tbody>
</table>

We compare the proposed ABC-based stacking algorithm with five state-of-the-art ensemble multilabel classification algorithms on all 10 benchmark datasets. The ten base level classifiers are CC [6], ECC [7], MLkNN [8], RAkEL [9], EPS [10], BRkNN [31], TSV A [32], TSCCA [33], IBLR [34], and CLR [35]. The five ensemble multilabel classification algorithms are RAkEL [9], ECC [6, 7], EPS [10], RF-PCT [12], and ML-F orest [11].

We compared the multilabel methods in the MULAN\(^4\) library under the machine learning framework WEKA, the CLUS system,\(^5\) and ML-F orest software.\(^6\) The MULAN library was used for CC, ECC, MLkNN, RAkEL, EPS, BRkNN, TSV A, TSCCA, IBLR, and CLR; the CLUS system was used for RF-PCT.

In our experiment, the training set is used to train the ABC-based stacking algorithm and the test set is only used to evaluate its performance. When training the proposed method, a five-fold cross-validation procedure was used to train the meta-level classifier. In each dataset, 70% of the instances are used for training the meta-level classifier and 30% of them are used to evaluate the performance of the meta-level classifier.

We used four example-based metrics (hamming loss, subset accuracy, example-based F, and example-based accuracy) and two ranking-based metrics (ranking loss and average precision) to measure the performance of the algorithms. All are commonly used metrics for the evaluation of multilabel classification. In order to compare the performance of several multilabel classifiers, we used a two-step procedure to assess the statistical significance among the algorithms. We applied a Friedman test with the null hypothesis that all learners have equal performance and, if the null-hypothesis is rejected, a post hoc test is carried out. The Nemenyi test was

---

\(^1\)http://mulan.sourceforge.net/datasets-mlc.html
\(^2\)http://computer.njnu.edu.cn/Lab/LABIC/LABIC_index.html
\(^3\)https://sourceforge.net/projects/meka/files/Datasets/
\(^4\)http://mulan.sourceforge.net/download.html
\(^5\)http://clus.sourceforge.net/
\(^6\)https://sites.google.com/site/qysite/
applied to further analyze the differences between two algorithms. The Friedman–Nemenyi test results [4, 36] are reflected in critical distance diagrams. The algorithms linked with a line within the critical distance (CD) are not statistically significantly better than each other. The CD for Nemenyi test is calculated as follows:

\[
CD = q_{\alpha} \sqrt{\frac{k(k+1)}{6|D|}},
\]

where \(k\) is the number of algorithms that are being compared and \(|D|\) is the number of datasets involved in the experiment. The standard deviations in twenty independent runs are also measured to verify the stability of ABC-based stacking.

3.2. Parameter settings

In ABC-based stacking, the ABC algorithm parameters were initialized with the value of \(SN\) equal to the number of total base classifiers. The maximum number of iterations was set to 50 and the value of max limit was set 3. The base classifier subset can be explored and exploited by three types of bees under these parameter settings. For the meta-level classifier to work in ABC-based stacking, two parameters need to be set in the meta-level classifier: learning rate \(\eta\) and iteration number \(T\). To secure optimal parameters, we used 70% of the meta data as the training set to build the single-layer ANN and 30% of the meta-data as the validation set for tuning the two parameters on the dataset Emotions. Figure 2 shows loss curves when different learning rates are used. All three loss curves decrease rapidly at first and then slow down as iterations progress on the four different configurations of base-level classifiers. To reduce the running time, we set the iteration number to 150 as the termination condition. Three different learning rates, 0.005, 0.015, and 0.025, were tested with different base classifier sets on the Emotions. We took 0.025 as the learning rate value in all subsequent experiments because its loss curves decreased faster than the other two, as shown in Figure 2. In our experiments, ABC-based stacking is conducted for 20 independent runs and the averages and the standard deviations are measured to verify its stability.

3.3. Experimental results

Tables 2–5 show the evaluation results of our method and five state-of-the-art multilabel algorithms on the example-based metrics described above. For each metric, a up-arrow ‘↑’ indicates that a larger value is preferred while the down-arrow ‘↓’ indicates that a smaller value is preferred. The values of the averages and standard deviations achieved by ABC-based stacking are shown in the last column. The numbers in bold font indicate the best performance among all ensemble algorithms. In Tables 2–5, the results indicate that ABC-based stacking has very competitive performance compared to the other ensemble methods. Table 2 shows that ABC-based stacking achieves the best results on 7 out of 10 datasets in terms of hamming loss, while the times for RAKEL, RF-PCT, and ML-Forest are 1, 1, and 1, respectively. ABC-based stacking performed well on all datasets even when it was not the best performer. For Medical, GpositivePseACC, and Music, ABC-based stacking was the 2nd, 2nd, and 3rd best performer, respectively. Table 3 shows that ABC-based stacking achieves the best results on 5 out of 10 datasets in terms of subset accuracy, while the figures for ML-Forest, RF-PCT, ECC, and RAKEL are 2, 2, 2, and 1, respectively. In Yeast, Emotions, and GpositivePseACC, ABC-based stacking is the 2nd best performer. In Enron and Music, ABC-based stacking is the 3rd best performer. The example-based F measure results shown in Table 4 indicate that ABC-based stacking outperforms other methods for 6 out of 10 datasets and is the 2nd best performer for the other four datasets (Mediamill, Yeast, GpositivePseACC, and
Figure 2. Loss curves with three different learning rates on Emotions.

Music). Using example-based accuracy, Table 5 shows that ABC-based stacking achieves the best performance for 4 out of 10 datasets, while the figures for FR-PCT and ML-FOREST are 4 and 2, respectively. ABC-based stacking outperforms RF-PCT for 6 of the 10 datasets. The low values of standard deviation inside parentheses in Tables 2–5 indicate the consistency of our method in varying conditions.

Table 2. Ensemble method performance measured by hamming loss, ↓.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RAkEL</th>
<th>ECC</th>
<th>EPS</th>
<th>RF-PCT</th>
<th>ML-FOREST</th>
<th>ABC-based stacking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>0.0509</td>
<td>0.0488</td>
<td>0.0541</td>
<td>0.0480</td>
<td>0.0510</td>
<td><strong>0.0474</strong> (0.001)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.1144</td>
<td>0.1008</td>
<td>0.1052</td>
<td>0.0927</td>
<td>0.0967</td>
<td><strong>0.0853</strong> (0.004)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.2331</td>
<td>0.2116</td>
<td>0.2126</td>
<td>0.2073</td>
<td>0.1990</td>
<td><strong>0.1922</strong> (0.004)</td>
</tr>
<tr>
<td>Medical</td>
<td><strong>0.0113</strong></td>
<td>0.0124</td>
<td>0.0137</td>
<td>0.0126</td>
<td>0.0144</td>
<td>0.0116 (0.001)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.2236</td>
<td>0.2137</td>
<td>0.2244</td>
<td>0.2005</td>
<td>0.2954</td>
<td><strong>0.1997</strong> (0.001)</td>
</tr>
<tr>
<td>Mediamill</td>
<td>0.0379</td>
<td>0.0341</td>
<td>0.0336</td>
<td>0.0333</td>
<td>0.0371</td>
<td><strong>0.0328</strong> (0.000)</td>
</tr>
<tr>
<td>VirusGO</td>
<td>0.0215</td>
<td>0.0202</td>
<td>0.0349</td>
<td>0.0269</td>
<td>0.0349</td>
<td><strong>0.0188</strong> (0.007)</td>
</tr>
<tr>
<td>GpositivePseACC</td>
<td>0.2127</td>
<td>0.1779</td>
<td>0.1947</td>
<td>0.1694</td>
<td>0.1611</td>
<td><strong>0.1659</strong> (0.015)</td>
</tr>
<tr>
<td>Music</td>
<td>0.2393</td>
<td>0.2117</td>
<td>0.2348</td>
<td><strong>0.2078</strong></td>
<td>0.2432</td>
<td>0.2196 (0.013)</td>
</tr>
<tr>
<td>Image</td>
<td>0.1860</td>
<td>0.1748</td>
<td>0.1760</td>
<td>0.1673</td>
<td>0.1970</td>
<td><strong>0.1520</strong> (0.006)</td>
</tr>
</tbody>
</table>

As discussed above, we also employed a Friedman test and a post hoc Nemenyi test to evaluate the proposed method against the other methods. The Friedman test was run with a significance level of 5%. Since the null hypothesis was rejected, we used a Nemenyi post hoc test to further analyze the differences between each pair of algorithms. The Nemenyi post hoc test results with average rank diagrams are shown in Figure 3,
Table 3. Ensemble method performance measured by subset accuracy, ↑.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RAkEL</th>
<th>ECC</th>
<th>EPS</th>
<th>RF-PCT</th>
<th>ML-FOREST</th>
<th>ABC-based stacking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>0.1071</td>
<td>0.1468</td>
<td>0.1192</td>
<td>0.1192</td>
<td>0.1261</td>
<td>0.1209(0.012)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.5100</td>
<td>0.5594</td>
<td>0.5744</td>
<td>0.5903</td>
<td>0.6463</td>
<td>0.6957(0.016)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.1243</td>
<td>0.1516</td>
<td>0.1658</td>
<td>0.1658</td>
<td>0.2116</td>
<td>0.2007(0.016)</td>
</tr>
<tr>
<td>Medical</td>
<td>0.6403</td>
<td>0.5953</td>
<td>0.6357</td>
<td>0.5844</td>
<td>0.5891</td>
<td>0.6775(0.015)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.1931</td>
<td>0.2574</td>
<td>0.2772</td>
<td>0.3218</td>
<td>0.0396</td>
<td>0.2970(0.007)</td>
</tr>
<tr>
<td>Mediamill</td>
<td>0.0360</td>
<td>0.0700</td>
<td>0.0710</td>
<td>0.067</td>
<td>0.0360</td>
<td>0.0760(0.006)</td>
</tr>
<tr>
<td>VirusGO</td>
<td>0.9032</td>
<td>0.9032</td>
<td>0.8387</td>
<td>0.8548</td>
<td>0.8145</td>
<td>0.9032(0.044)</td>
</tr>
<tr>
<td>GpositivePseACC</td>
<td>0.5433</td>
<td>0.5192</td>
<td>0.5481</td>
<td>0.6010</td>
<td>0.6731</td>
<td>0.6442(0.031)</td>
</tr>
<tr>
<td>Music</td>
<td>0.2027</td>
<td>0.2365</td>
<td>0.2128</td>
<td>0.2973</td>
<td>0.1655</td>
<td>0.2230(0.033)</td>
</tr>
<tr>
<td>Image</td>
<td>0.4013</td>
<td>0.4075</td>
<td>0.4275</td>
<td>0.445</td>
<td>0.3263</td>
<td>0.5325(0.013)</td>
</tr>
</tbody>
</table>

Table 4. Ensemble method performance measured by example-based F measure, ↑.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RAkEL</th>
<th>ECC</th>
<th>EPS</th>
<th>RF-PCT</th>
<th>ML-FOREST</th>
<th>ABC-based stacking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>0.5221</td>
<td>0.5434</td>
<td>0.4402</td>
<td>0.5586</td>
<td>0.5519</td>
<td>0.5626(0.017)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.6051</td>
<td>0.6105</td>
<td>0.6321</td>
<td>0.6856</td>
<td>0.7210</td>
<td>0.7532(0.011)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.5933</td>
<td>0.5912</td>
<td>0.5972</td>
<td>0.6402</td>
<td>0.6489</td>
<td>0.6405(0.182)</td>
</tr>
<tr>
<td>Medical</td>
<td>0.7590</td>
<td>0.7083</td>
<td>0.7356</td>
<td>0.7033</td>
<td>0.7716</td>
<td>0.7976(0.027)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.5936</td>
<td>0.5624</td>
<td>0.5986</td>
<td>0.6772</td>
<td>0.1398</td>
<td>0.6696(0.006)</td>
</tr>
<tr>
<td>Mediamill</td>
<td>0.4844</td>
<td>0.4940</td>
<td>0.4924</td>
<td>0.5331</td>
<td>0.2955</td>
<td>0.6338(0.002)</td>
</tr>
<tr>
<td>VirusGO</td>
<td>0.9419</td>
<td>0.9473</td>
<td>0.9204</td>
<td>0.9387</td>
<td>0.9220</td>
<td>0.9591(0.025)</td>
</tr>
<tr>
<td>GpositivePseACC</td>
<td>0.5529</td>
<td>0.5321</td>
<td>0.5545</td>
<td>0.6490</td>
<td>0.6955</td>
<td>0.6763(0.028)</td>
</tr>
<tr>
<td>Music</td>
<td>0.5633</td>
<td>0.5992</td>
<td>0.5687</td>
<td>0.6746</td>
<td>0.5515</td>
<td>0.6173(0.019)</td>
</tr>
<tr>
<td>Image</td>
<td>0.5818</td>
<td>0.5253</td>
<td>0.5682</td>
<td>0.6207</td>
<td>0.4230</td>
<td>0.6889(0.012)</td>
</tr>
</tbody>
</table>

Table 5. Ensemble method performance measured by example-based accuracy, ↑.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RAkEL</th>
<th>ECC</th>
<th>EPS</th>
<th>RF-PCT</th>
<th>ML-FOREST</th>
<th>ABC-based stacking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>0.4099</td>
<td>0.4388</td>
<td>0.3467</td>
<td>0.4462</td>
<td>0.4400</td>
<td>0.4458(0.015)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.5810</td>
<td>0.5977</td>
<td>0.6176</td>
<td>0.6618</td>
<td>0.7021</td>
<td>0.7835(0.011)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.4777</td>
<td>0.4840</td>
<td>0.4905</td>
<td>0.5294</td>
<td>0.5465</td>
<td>0.5351(0.022)</td>
</tr>
<tr>
<td>Medical</td>
<td>0.7284</td>
<td>0.6800</td>
<td>0.7109</td>
<td>0.6736</td>
<td>0.7247</td>
<td>0.7671(0.023)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.5008</td>
<td>0.4880</td>
<td>0.5241</td>
<td>0.5940</td>
<td>0.1134</td>
<td>0.5771(0.006)</td>
</tr>
<tr>
<td>Mediamill</td>
<td>0.3593</td>
<td>0.3786</td>
<td>0.3801</td>
<td>0.4109</td>
<td>0.2003</td>
<td>0.3969(0.002)</td>
</tr>
<tr>
<td>VirusGO</td>
<td>0.9328</td>
<td>0.9368</td>
<td>0.9005</td>
<td>0.9194</td>
<td>0.8952</td>
<td>0.9462(0.025)</td>
</tr>
<tr>
<td>GpositivePseACC</td>
<td>0.5505</td>
<td>0.5288</td>
<td>0.5529</td>
<td>0.6370</td>
<td>0.6899</td>
<td>0.6683(0.028)</td>
</tr>
<tr>
<td>Music</td>
<td>0.4710</td>
<td>0.5093</td>
<td>0.4817</td>
<td>0.5839</td>
<td>0.4535</td>
<td>0.5177(0.022)</td>
</tr>
<tr>
<td>Image</td>
<td>0.5357</td>
<td>0.4953</td>
<td>0.5324</td>
<td>0.5763</td>
<td>0.3982</td>
<td>0.6494(0.012)</td>
</tr>
</tbody>
</table>

where $CD$ is 2.38 and the significance level is $P = 0.05$ for four example-based metrics. Values on the horizontal axis are the average ranks of the algorithms, where the best rank is placed at the right-most side of the diagram. Algorithms that do not differ significantly are connected by a red line.

As shown in Figure 3, the best-performing methods for each measure are ABC-based stacking followed by RF-PCT and ML-forest. RF-PCT performs 2nd best according to hamming loss, subset accuracy, F, and accuracy. ML-forest performs 3rd best according to F and accuracy, and 4th according to hamming loss and
5th according to subset accuracy. ABC-based stacking achieves the best average rank of hamming loss in Figure 3a. The reason for this is that our method specifically optimizes the hamming loss. The proposed method has the best average rank on the measurement of subset accuracy (Figure 3b). This suggests that the predictions from ABC-based stacking are more accurate than those from other methods. The proposed method also has the best example-based F and accuracy measurements (Figures 3c and 3d) among the algorithms we tested, which reflects its strong robustness and reliability.

Figure 3. Critical diagrams of four example-based measures: Nemenyi post hoc test results on 10 datasets.

We used two ranking-based measures to offer a different perspective on the above results. Tables 6 and 7 show the results of the proposed method in comparison to five state-of-the-art multilabel algorithms on average precision and ranking loss. The values of the averages and standard deviations achieved by ABC-based stacking are shown in the last column. In Table 6, ABC-based stacking achieves the best results on 8 out of 10 datasets for average precision, while the times for RF-PCT and ML-Forest are 1 and 1, respectively. In the Mediamill and GpositivePseACC datasets, ABC-based stacking is the 2nd best performer. Table 7 shows that ABC-based stacking again performs best on 8 out of 10 datasets in terms of ranking loss while the figures for RF-PCT and ML-Forest are 1 and 1, respectively. In GpositivePseACC and Mediamill, ABC-based Stacking is the 2nd and 3rd best performer, respectively. The low values of standard deviations inside the parentheses in Table 6 and Table 7 indicate that the robustness and effectiveness of ABC-based stacking are verified.

Figure 4 shows the results of the Nemenyi post hoc test ($CD=2.38$ and $p=0.05$) for two ranking-based metrics on 10 datasets. The ranks are depicted on the axis in such a manner that the best-ranking algorithm is at the right-most side of the diagram. The algorithms that do not differ significantly are connected with a red line. Most of the pairwise comparisons remain statistically not significant, but the overall picture is clearly in favor of the proposed method. The best-performing method for two measures is ABC-based stacking, RF-PCT
Table 6. Ensemble method performance measured by average precision, ↑.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RAkEL</th>
<th>ECC</th>
<th>EPS</th>
<th>RF-PCT</th>
<th>ML-FOREST</th>
<th>ABC-based stacking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>0.6086</td>
<td>0.6741</td>
<td>0.5915</td>
<td>0.6842</td>
<td>0.6775</td>
<td><strong>0.6955</strong> (0.007)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.8162</td>
<td>0.8395</td>
<td>0.8102</td>
<td>0.8634</td>
<td>0.8505</td>
<td><strong>0.8733</strong> (0.006)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.7094</td>
<td>0.7397</td>
<td>0.7312</td>
<td>0.7544</td>
<td>0.7623</td>
<td><strong>0.7635</strong> (0.006)</td>
</tr>
<tr>
<td>Medical</td>
<td>0.8141</td>
<td>0.8605</td>
<td>0.8089</td>
<td>0.8433</td>
<td>0.8568</td>
<td><strong>0.8717</strong> (0.040)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.7833</td>
<td>0.7853</td>
<td>0.7766</td>
<td>0.8042</td>
<td>0.7880</td>
<td><strong>0.8225</strong> (0.003)</td>
</tr>
<tr>
<td>Mediamill</td>
<td>0.5790</td>
<td>0.6690</td>
<td>0.6267</td>
<td>0.6883</td>
<td>0.6717</td>
<td><strong>0.6879</strong> (0.011)</td>
</tr>
<tr>
<td>VirusGO</td>
<td>0.7432</td>
<td>0.7933</td>
<td>0.7660</td>
<td>0.8048</td>
<td><strong>0.8205</strong></td>
<td>0.8129 (0.016)</td>
</tr>
<tr>
<td>GpositivePseACC</td>
<td>0.7645</td>
<td>0.7883</td>
<td>0.7697</td>
<td>0.7818</td>
<td>0.7693</td>
<td><strong>0.7987</strong> (0.011)</td>
</tr>
<tr>
<td>Music</td>
<td>0.7863</td>
<td>0.7924</td>
<td>0.7886</td>
<td>0.8182</td>
<td>0.8027</td>
<td><strong>0.8343</strong> (0.006)</td>
</tr>
<tr>
<td>Image</td>
<td>0.6683</td>
<td>0.7288</td>
<td>0.6992</td>
<td>0.7386</td>
<td>0.7258</td>
<td><strong>0.7637</strong> (0.011)</td>
</tr>
</tbody>
</table>

Table 7. Ensemble method performance measured by ranking loss, ↓.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RAkEL</th>
<th>ECC</th>
<th>EPS</th>
<th>RF-PCT</th>
<th>ML-FOREST</th>
<th>ABC-based stacking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>0.2027</td>
<td>0.0846</td>
<td>0.1773</td>
<td>0.0839</td>
<td>0.0874</td>
<td><strong>0.0775</strong> (0.004)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.1171</td>
<td>0.954</td>
<td>0.1225</td>
<td>0.0800</td>
<td>0.0958</td>
<td><strong>0.0710</strong> (0.003)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.2244</td>
<td>0.1892</td>
<td>0.2055</td>
<td>0.1702</td>
<td>0.1714</td>
<td><strong>0.1634</strong> (0.004)</td>
</tr>
<tr>
<td>Medical</td>
<td>0.0729</td>
<td>0.0400</td>
<td>0.0668</td>
<td>0.0333</td>
<td>0.0397</td>
<td><strong>0.0331</strong> (0.029)</td>
</tr>
<tr>
<td>Emotions</td>
<td>0.1863</td>
<td>0.1769</td>
<td>0.1874</td>
<td>0.1624</td>
<td>0.1838</td>
<td><strong>0.1431</strong> (0.002)</td>
</tr>
<tr>
<td>Mediamill</td>
<td>0.1598</td>
<td>0.0689</td>
<td>0.1329</td>
<td><strong>0.0553</strong></td>
<td>0.0592</td>
<td>0.0622 (0.002)</td>
</tr>
<tr>
<td>VirusGO</td>
<td>0.0290</td>
<td>0.0065</td>
<td>0.0154</td>
<td>0.0658</td>
<td>0.0881</td>
<td><strong>0.0048</strong> (0.006)</td>
</tr>
<tr>
<td>GpositivePseACC</td>
<td>0.2548</td>
<td>0.1871</td>
<td>0.2131</td>
<td>0.1870</td>
<td><strong>0.1583</strong></td>
<td>0.1631 (0.015)</td>
</tr>
<tr>
<td>Music</td>
<td>0.1967</td>
<td>0.1702</td>
<td>0.1950</td>
<td>0.1697</td>
<td>0.1930</td>
<td><strong>0.1596</strong> (0.007)</td>
</tr>
<tr>
<td>Image</td>
<td>0.1833</td>
<td>0.1733</td>
<td>0.1779</td>
<td>0.1470</td>
<td>0.1703</td>
<td><strong>0.1330</strong> (0.007)</td>
</tr>
</tbody>
</table>

Figure 4. Critical diagrams for two ranking-based measures: Nemenyi post hoc test results.

is in 2nd place, ML-forest and ECC are in the middle, and EPS is in 5th place while RAkEL is the worst among all the methods we tested. The results from Figure 4 suggest that ABC-based stacking is the best performer in the prediction of labels ranking.

4. Conclusions
In this study, we have investigated the use of the ensemble approach to enhance the performance of multilabel classification. We have proposed a novel stacking ensemble algorithm, ABC-based stacking, which uses the
ABC algorithm to search base classifier configurations and then sets the outputs of the chosen classifiers as inputs for a meta-classifier. A single-layer ANN is used as the meta-level classifier. We conducted experiments on 10 datasets from different domains to evaluate the performance of our method by comparison with several state-of-the-art ensemble methods. The results show that the proposed algorithm is highly competitive. The main disadvantage of our proposed method is that, like many other ensemble methods, it is more complex with multiple bases classifiers and requires longer training time compared with some other alternative approaches. In spite of this, it can still be a good choice for multilabel learning in many cases.

References


