Extensions of the Separate-and-Conquer Multilabel Rule Learner

Erweiterungen des Separate-and-Conquer Multilabel-Regellerners
Bachelor-Thesis by Borhan Youssef Salem from Gafsa
Date of Submission:

1. Referee: Prof. Dr. Johannes Fürnkranz
2. Referee: Dr. Eneldo Loza Mencía

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Knowledge Engineering Group
Fachbereich Informatik

Extensions of the Separate-and-Conquer Multilabel Rule Learner
Erweiterungen des Separate-and-Conquer Multilabel-Regellerners

Submitted Bachelor-Thesis by Borhan Youssef Salem from Gafsa

1. Referee: Prof. Dr. Johannes Fürnkranz
2. Referee: Dr. Eneldo Loza Mencía

Date of Submission:

Thesis Statement pursuant to §22 paragraph 7 of APB TU Darmstadt

Darmstadt, 18.7.2017

(Borhan Youssef Salem)

Abstract

Multi-label classification is the task in Machine Learning to assign more than one label to an instance. Opposite to the single-label classification problem, where only a binary or a multi-class can be assigned to an instance, dependencies may exist between different labels in a multi-label problem. These dependencies can be used to improve the classification task and help to better understanding the multi-label dataset. A Separate-and-Conquer Multi-Label Rule Learner was proposed 2016 by Eneldo Loza Mencía and Frederik Janssen, that learn multi-label dependency and use them in the classification task. In this work we made some extensions of the proposed algorithm and evaluate them.

Zusammenfassung

Die Multi-label Klassifikation ist eine Aufgabe im Kontext des maschinellen Lernens, die darin besteht, einer Instanz mehr als ein Label zuzuordnen. Im Gegensatz zur Single-Label Klassifikation, in welcher nur eine binäre oder eine Multi-Klasse einer Instanz zugeordnet werden können, können Abhängigkeiten zwischen verschiedenen Labels vorhanden sein. Diese Abhängigkeiten können benutzt werden um die Klassifikation zu verbessern und die Multi-label Datasets besser zu verstehen. Ein Separate-and-Conquer Multi-Label-Regellerner wurde 2016 von Eneldo Loza Mencía und Frederik Janssen vorgeschlagen, der die Multi-Label Abhängigkeit lernt und in der Klassifikation benutzt. In dieser Arbeit haben wir ein paar Erweiterungen zur diesem Algorithmus hinzugefügt und getestet.

Inhaltsverzeichnis

Tabellenverzeichnis

List of Algorithms

Abbildungsverzeichnis

1 Introduction

Multi-label Classification (MLC) is a classification problem in machine learning, where multiple label can be assigned to an instance. This task is opposed to the single-label classification as binary or multi-class problems, where only a single class can be assigned to an instance [1] Multi-label classification has become the increasing interest of the data mining community and is usedin different domains such as text classification [3] and scene and video classification [2] For single-label problem dependencies exist only between the single class and the instances attribute's values. In opposite to this dependency in multi-label problem may exist between different labels. For examples in text classifications of papers, this thesis has both labels "Machine Learning" and its subtopic "Multi-label Classification". The presence of the second label implies the presence of the first. Such a dependency can be used to perform the classifications task. In this work we will use a multi-label separate-and-conquer rule Learner that are able to learn these multi-label dependencies.
In the first section we introduce a formalization of the multi-label classification and the multi-label dependency. We will present also the evaluation metrics used in this work and a toy example.
In the second section, we present the multi-label separate-and-conquer rule Learner proposed 2016 von Eneldo LozaMencía LozaMencía und Frederik Janssen. We describe the different parameters of this learner and then applied it on the toy example.
In the third section we propose some extensions of the presented multi-label separate-and-conquer rule Learner. We add a stopping criteria for searching of Label Rules. We change the algorithms so that fully covered training Examples will be used in the rule learning process. We add an option to combine positive and negative head-rule in a one rule pair.
In the fourth section, these proposed extensions will be evaluated.

2 Multi-label Classification

An instance $x$can be anything and have different attributes. We can represent it as a vector of n elements x_i. The element x_i corresponds to the value of the n attribute A_i that characterizes the instance.X is the set of all possible instances.

$$x = (x_1, \ldots, x_n) \in X \tag{1}$$

An instance x can be associated to a multiple label λ_i from a predefined set of Labels L.

$$L = \lambda_1, \ldots, \lambda_m \text{ with } m > 2 \tag{2}$$

For each instance x we define a Label vector with m elements that indicate the presence of corresponding label and its absence. is the set of all possible combinations of classes.

$$y = (y_1, \ldots, y_m) \in Y \text{ with } y_i = \begin{cases} 0 & \text{if class } \lambda_i \text{ is absent} \\ 1 & \text{if class } \lambda_i \text{ is present} \end{cases} \tag{3}$$

The value 1 indicates the presence of correspondent label and 0 its absence. Multi-label Classification task is to learn a function that maps instances x to the corresponding label vector $\hat{y}$.

$$\begin{array}{rcccl} f & : & X & \to & Y \\ & & x & \mapsto & f(x) = \hat{y} = (\hat{y}_1, \ldots, \hat{y}_m) \end{array} \tag{4}$$

Such a function is learned from the training data T that is consisting of a set of instances and its correspondent labels vector

$$T = ((x_1, y_1), \ldots, (x_t, y_t)) \tag{5}$$

Different decomposition approaches are used to solve the multi-label classification task. The problem will be divided into single label sub-problems and solved independently of each other. This is main issues these approaches. The dependencies between different classes can be used to perform the classification and this separation can lead to loss of useful information. [2]

2.1 Toy Example

Table 2 presents a Multi-label dataset. This is an extended version of the weather dataset. In the all following we will use this toy Example for explanations. The label c1describeswhether to play tennis depending in the weather forecast.If it is overcast, sunny and normally humid or runny but not windy we can play tennis.Label c2 indicate whether we should eat ice cream. We eat ice cream if it is sunny and too humid. Label c3 and c4indicate whether we should drink tea or lemonade. We drink tea if it is rainy and windy and lemonade if it is warm enough. The label c5 is the negation of the label c1. It is used for illustrative purposes; this is to represent various dependency relations between the labels. [2]

2.2 Rule Learning

Rule learning algorithm try to find a set of simple If-Then rules that approximate in a discrete, piecewise way the classification function $h(x)$ (in Formal 2.4). The set of the learned rule R is called the Theory. These rules are easy to be interpreted and comprehended by human than others complex models such as support vector machines SVMs or neural networks.A propositional rule r is composed of a body and a head.

$$r : \text{head} \leftarrow \text{body} \tag{6}$$

Tabelle 1: Extended multi-label weather dataset.

Instances	A1	A2	A3	A4	Attributes	C1	C2	C3	C4	C5
	outlook	temperature	humidity	windy		play	icecream	tea	lemonade	dontplay
X1	rainy	65	70	yes	Y1	0	0	1	0	1
X2	rainy	71	91	yes	Y2	0	0	1	0	1
X3	sunny	85	85	no	Y3	0	1	0	1	1
X4	sunny	80	90	yes	Y4	0	1	0	1	1
X5	sunny	72	95	no	Y5	0	1	0	1	1
X6	sunny	69	70	no	Y6	1	0	0	1	0
X7	sunny	75	70	yes	Y7	1	0	0	1	0
X8	overcast	83	86	no	Y8	1	0	0	1	0
X9	overcast	64	65	yes	Y9	1	0	0	1	0
X10	overcast	72	90	yes	Y10	1	0	0	1	0
X11	overcast	81	75	no	Y11	1	0	0	1	0
X12	rainy	70	96	no	Y12	1	0	0	1	0
X13	rainy	68	80	no	Y13	1	0	0	1	0
X14	rainy	75	80	no	Y14	1	0	0	1	0

The body is formed of a number of conditions.These conditions are tested on attribute values. They can be combined in a conjunctive or disjunctive way. The head is formed of one condition that set a class value to a label. We can distinguish two types of rule according to the class value to be predicted. Positive head rule with the condition c = 1 that indicate the presence of the class c and negative head (c = 0) rule that indicate the absence of this class. If the body conjunctive condition is true,than the body condition will be predicted. A rule r is said to cover an instance x if it satisfies the body condition of the rule.[5]In this work we use only conjunctive rules with both positive and negative heads.We use instead of c = 1 the expression c and instead of c=0 the expression $\hat{c}$. Following rules r_1 and r_2 are learned from the label from the weather dataset in table 2.

$$\begin{aligned} r_1 &: \text{icecream} \leftarrow \text{outlook} = \text{sunny} \text{ , humidity} >= 82.5 \\ r_2 &: \overline{icecream} \leftarrow \text{temperature} <= 84.0 \end{aligned} \tag{7}$$

The rule r_1 is a positive head rule, that cover all examples whose outlook is sunny and humidity greater than 82.5 and classify the label $icecream = 1$. The rule r_2 is a negative head rule, that cover all example with less temperature less than 84.0 and classify the label $icecream = 0$. The most popular strategy to learn a set of rules is the separate-and-conquer or covering approach. The term separate-and-conquer is invented by Pagallo and Haussler (1990) due to the way of finding the theory.

Algorithm 1 SEPARATEANDCONQUER(Examples)[2]

```
1: $Theory \leftarrow \emptyset$
2: while $POSITIVE(Examples) \neq \emptyset$ do
3:     $Rule = FINDBESTRULE(Examples)$        ▷ the conquer step: find a "good" rule
4:     $Covered = COVER(Examples)$
5:     $Examples = Examples \setminus Covered$        ▷ the separate step: remove the covered examples
6:     $Theory = Theory \cup Rule$
7: end while
```

First a single rule is learned that covers a part of the given training instances. This rule is added to the theory and the covered examples are deleted. This is the separate part. The conquer part is the recursively learning of another rule on the remaining training instances until no examples are left. [2]

2.3 Label Dependency

From a probabilistic point of view. There are two types of dependencies, unconditional and conditional. The unconditional dependency does not depend on the given input instances. Conditional dependency depends on attributes of input instances [2] The exclusion dependency the label and from the weather dataset doesn't depend on particular instances.The presence of label play implies the absence of do not play and vice versa. The probability that the label do not play is absent have to be higher if the label play is present. This probability is unconditional. Such dependencies may also refer to a global and local dependency. The unconditional dependency describes a relation between labels globally and the unconditional one describes locally relations that exist only in a subset of the input instances. [2]

To use thesedependencies to improve the classification task for multi-label datasets, we have to change the rule structure. We have to extend the rule structure so that we are able to have multi-label heads and test on labels in the body of the rule.A rule can have different assignment in the head. We can distinguish two types of rule according to the label assignment. The dense one which must have an assignment of all labels on the head of the rule and the sparse which have only assignment of some of the labels. We extend the rule so that we are able to have a test on labels in the body of the rule and not only tests on attributes. Those changes in the rule structure can model all dependencies between different labels that may be existing. In this work we are interested only in single-label head rules. The following Table 3 presents the different type of multi-label rules.

Tabelle 2: Different forms of single-head multi-label rules.

Multi-label rule type	example rule on the weather Dataset
label-independent	$icecream \longleftarrow outlook = sunny, humidity >= 82.5$ $\overline{icecream} \longleftarrow$
partially label-dependent	$tea \longleftarrow outlook = rainy, lemonade = 0$ $\overline{play} \longleftarrow icecream = 1, outlook = sunny$
fully label-dependent	$play \longleftarrow icecream = 0, lemonade = 1$ $\overline{dontplay} \longleftarrow play = 1$

A rule that contains tests on labels is called label-dependent opposite to the label-independent rules that contain only test on attributes. The unconditional dependency will be modeled by a fully label-dependent rule. Such rules have only tested on labels and represent a global dependency that doesn't depend on the given input instances. The conditional dependency is modeled by partition label-dependent rules. This rule type contains test on both labels and attributes and represent a locally dependency between different labels. Label-dependent rules are suitable to represent different dependencies (implications, subsumptions, or exclusion) in multi-label problems. This can help us to better understand multi-label datasets and improve the classification task using those dependencies. [2]

2.4 Evaluation Metrics

There are many evaluation metrics for multi-label classification. Depending on the type of prediction there are two types of metrics: bipartition and ranking evaluation metrics. In this work we introduce only the bipartition metrics because we do not use any ranking approaches. A bipartition metrics are an error function δ that evaluate the difference between the predicted label vector $\hat{y}$ and the correct label vector y.

$$\begin{array}{rcl} \delta \;:\; Y * Y & \rightarrow & [0, \infty) \\ (y, \hat{y}) & \mapsto & \delta((y, \hat{y})) \end{array} \tag{8}$$

To evaluate this difference a 2x2 dimensional matrix is used to call confusion matrix C_i^j. The variable j is used as an iterator over test examples set $(X1,...,xn)$ and the variable i is used as an iterator over the set of labels L. The elements of this matrix are the true t_p and false positives f_p, and the true t_n and false negatives f_n.

$$C_j^i = \begin{pmatrix} t_p & f_n \\ f_p & t_n \end{pmatrix} \tag{9}$$

For a test instance x_j and a label y_i the elements of the atomic confusion matrix C_i^j are computed as follows

$$\begin{aligned} t_p &= \begin{cases} 1 & y_i = 1 \wedge \hat{y}_i = 1 \\ 0 & \text{otherwise} \end{cases} \quad f_n = \begin{cases} 1 & y_i = 1 \wedge \hat{y}_i = 0 \\ 0 & \text{otherwise} \end{cases} \\ f_p &= \begin{cases} 1 & y_i = 0 \wedge \hat{y}_i = 1 \\ 0 & \text{otherwise} \end{cases} \quad t_n = \begin{cases} 1 & y_i = 0 \wedge \hat{y}_i = 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{10}$$

We need to define two aggregations operators:

$$\sum_{i=1}^{n} C_i = C_1 \oplus ... \oplus C_n \tag{11}$$

With $\oplus$ is the cell-wise addition of matrix.

$$avg_{i=1}^{n} C_i = \frac{1}{n} \sum_{i=1}^{n} C_i \tag{12}$$

For t test instances and m labels there are $t.m$ atomic confusion matrix $C_i j$. These matrices are not suitable to become a single comparative value of a test set andevaluate the quality of a multi-label classifier. For these two averaging strategies are used: micro- and macro-averaging: Micro-averaging aggregates first all atomics matrix and then calculate the value of the resulting confusion matrix with the evaluation function.Macro-averaged aggregates the results of the evaluation function applied to all atomic matrix. For Multi-label task, there are two possibilities to iterate in order to aggregate to a single value: the labels and the examples. This result on four mathematically distinct combinations of the aggregations:

(Label and Example-based)Micro-Averaging δ :	$\delta \circ \sum_i \circ \sum j = \delta \circ \sum_j \circ \sum i$
(example-based (macro-)averaged,label-based micro-averaged δ :	$avg_j \circ \delta \circ \sum i$
(label-based (macro-)averaged,example-based micro-averaged δ :	$avg_i \circ \delta \circ \sum j$
((label and example-based)macro-averaged δ :	$avg_i \circ avgj \circ \delta = avg_j \circ avgi \circ \delta$

As evaluation function we will use in the remain of this work the following measures: Precision, Recall and F-measure. The precision is the percentage of true positives from all the classified as positive.

$$Prec(C) = \frac{t_p}{t_p + f_p} \tag{13}$$

The recall is the percentage of examples classified aspositive fromall the positives.

$$Rec(C) = \frac{t_p}{t_p + f_n} \tag{14}$$

The F1-measure is the harmonic mean between of both Precision and Recall.

$$F1(C) = \frac{2}{\frac{1}{Prec(C)} + \frac{1}{Rec(C)}} = \frac{2Prec(C)Rec(C)}{Prec(C) + Rec(C)} \tag{15}$$

We will also use the metrics Hamming and Subset Accuracy. The Hamming Accuracy is the percentage of correctly classified labels.

$$HamLoss(C) = \frac{t_p + t_n}{t_p + f_p + t_n + f_n} \tag{16}$$

The subset Accuracy is the percentage of perfectly predicted labelsets on the test set

$$Acc(Y,\hat{Y}) = \frac{1}{m}\sum_{j=1}^{m}[y_i = \hat{y}] \text{ with } [x] = \begin{cases} 1 & \text{if x is true} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

3 Separate-and-Conquer Multi-Label Rule Learner

In this section we will present an iterative separate-and-conquer multi-label rule learner proposed by LozaMencía and Janssen[2] Opposite to the Problem Transformation Methods(e.g. Binary Relevance, Pairwise Decomposition or Label Powerset) that divide the multi-label classification tasks into servals sub problems and obtainisolated theories offeach other, treats this algorithm the classification task as one problem and returns onereturns one Global theory explaining the multi-label dataset.

3.1 Training

The proposed rule learner differs to the separate-and-conquer algorithm in the concept of covering status and the removal of the instances from the training set. The following figure represents the training algorithm

Algorithm 2 Training algorithm for the multi-label iterative separate-and-conquer algorithm [2]

```
Require: New training example pairs $T = (x1; y1); \dots ; (xm; ym)$, parameters $\theta$,$\tau$, heuristic $h$, targets
    $B$(either $B = \{1\}$ or $B = \{0, 1\}$) whether using stopping rules, whether re-inserting fully covered
    examples
 1: $T = (x_1; \hat{y}_1), ..., (xm; \hat{y}_m)$ with $\hat{y}_i = (?, ?, ..., ?), i = 1 \dots m$
 2: while $\frac{|T|}{m} > \theta$ do                                  ▷ until, e.g., 95% of examples covered
 3:     $r \leftarrow$ findBestGlobalRule$(B, T)$                    ▷ get best possible rule regardless the head
 4:     add $r$ to decision list $R$
 5:     $(T, T_{part}, T_{full})$ =getCoveredSets$(r, T)$            ▷ separate $T$ according covering by $r$
 6:     $T_{add}$ $\leftarrow$getReAddSet$(T_{part}, T_{full}, \tau)$          ▷ depending on user parameters
 7:     if $T_{add} = \emptyset$ then
 8:         mark $r$ as stopping rule                        ▷ only uncovered examples in $T$ of next round
 9:     else
10:         $T \leftarrow T \cup T_{add}$                     ▷ add also some covered examples, do not remove them
11:     end if
12: end while
13: return decision list $R$
```

The algorithm work with two representations of the label vector: $Y = (y_1, \dots, y_n)$ is the original labels and $\hat{Y} = (\hat{y}_1, \dots, \hat{y}_n)$ the labels accessible by the learner. We initialize the label vector with unknown label information $(?, ?, \dots, ?)$ (Algorithm 2, line 1). The parameter is the percentage of remaining uncovered instances needed to leave the while loop. The outer loop (Algorithm 2, line 2- 12) runs intel only trainings examples are still uncovered (Algorithm 2, line2).First the Best global rule for the current Training set T and the Target B will be searched. Algorithm 2 represents the search algorithm used to find the best current rule.

Algorithm 3 Algorithm findBestGlobalRule for finding the best current rule on the training set for any possible label in the head [2]

```
Require: example pairs $T$ and targets $B$
 1: $r \leftarrow \emptyset, r.h \leftarrow \infty$                          ▷ init best rule and its heuristic value
 2: for each label $y_i$ and target $t \in B$ do
 3:     $T^i \leftarrow T$
 4:     remove all $x$ where $\cup y \neq ?$ from $Ti$                ▷ do not consider $x$ if label already set
 5:     $r^{'} \leftarrow$ findBestRule$(y_i, t, T^i)$                       ▷ find best body for target $y_i = t$
 6:     $r^{'}.h = h(r^{'}, y_i, T^i)$              ▷ heuristic value depends on target label and $T^i$
 7:     if $r^{'}.h > r.h$ then
 8:         $r \leftarrow r^{'}$                                                 ▷ replace by better rule
 9:     end if
10: end for
11: return best rule $r$
```

The best rule for each label and any possible label head will be searched and the best of them. There are two types of rules the positive and negative head rules. If the Targets $B = 1$ only positive head rules will be learned and if $B = 1, 0$ the Negative head rules will also be learned. A good rule for each label $y_i$and any possible label head will be searched on a Training set T^i (Algorithm 3, line 5). The examples x where the label y_i is already set will be removed from the examples set T^i (Algorithm 3, line 3-4). The value of this label is already set and doesn't need to be learned again.For a given label, the other label values will be considered as an attribute and used on the learning of the best rule. Form all founded rules The best one according to the heuristic function h will be searched (Algorithm 3, line 7,8) and then returned (Algorithm 3, line 11). The best Global rule learned will be the added to the decision list

R (Algorithm 2, line 4) and then the training examples will be separated from this rule according to the covering status(Algorithm 2, line 5). The following figure presents the algorithm for computing the different sets.

Algorithm 4 getCoveredSets for computing the covering status of examples for a given rule. [2]

```
Require: Rule $r$,example pairs $T$
 1: $T_{par}t \leftarrow \emptyset, T_{ful}l \leftarrow \emptyset$
 2: for each example $(x;\cup y) \in T$ do          ▷ compute covering status for each example
 3:     $T^i \leftarrow T$
 4:     if  then$r$ covers $x$
 5:         $T \leftarrow T \setminus x$             ▷ remove since it may not be re-added
 6:         apply head of $r$ on $\hat{y}$           ▷ replace corresponding value in $\hat{y}$ if it was unset
 7:         if $\hat{y}$ is fully set  then          ▷ depending on $B$, consider also unset zeros
 8:             $T_{full} \leftarrow T_{full} \cup x$
 9:         else
10:             $T_{part} \leftarrow T_{part} \cup x$
11:         end if
12:     end if
13: end for
14: return uncovered ($T$),partially ($T_{part}$) and fully covered ($T_{full}$) training examples
```

The uncovered, partially covered and fully covered examples have to be stored respectively in the sets T, T_{part} and T_{full}. All covered examples of the current rule will be removed from T (Algorithm 3, line 4). If the label to predicate by the given rule for a covered example isn't already set and still have an unknown value, the head of the rule will be then applied to the label (Algorithm 4, line 5). The fully covered examples, where all labels are already set, will be stored in and the partially covered in (Algorithm3, line 6-10). From these different sets and depending on the used parameter a set of examples will be computed to be re-included in the training set T (Algorithm 2, line 6). There are three different ways to re-add the covered examples. The following algorithm describes these proposed ways.

The uncovered, partially covered and fully covered examples have to be stored respectively in the sets T, and . All covered examples by the current rule will be removed from T (Algorithm 4, line 4). If the label to predicate by the given rule for a covered example isn't already set and still have an unknown value, the head of the rule will be then applied to the label (Algorithm 4, line 5). The fully covered examples, where all labels are already set, will be stored in T_{full} and the partially T_{part}covered in (Figure 4, line 6-10). From these different sets and depending on the used parameter a set of examples will be computed to be re-included in the training set T (figure 2, line 6). There are three different ways to re-add the covered examples. The following algorithm describes these proposed ways.

Algorithm 5 getCoveredSets for computing the covering status of examples for a given rule. [2]

```
Require: Partially and fully covered examples $T_{part}, T_{full}$ , parameter $\tau$, whether using stopping rules,
    whether re-inserting fully covered examples
 1: if use stopping rules then
 2:     if full coverage rate $\frac{|T_{full}|}{|T_{part}|+|T_{full}|} \geq \tau$ then                ▷ e.g.90%
 3:         $T_{add} \leftarrow \emptyset$          ▷ do not re-add any example although $T_{part}, T_{full}$ non empty
 4:         if too many partially covered examples then
 5:         else                                   ▷ too many partially covered examples
 6:             $T_{add} \leftarrow T_{part}$                      ▷ re-add partially covered examples
 7:             $T_{add} \leftarrow T_{add} \cup T_{full}$               ▷ re-add also fully covered examples
 8:         end if
 9:     end if
10: else
11:     $T_{add} \leftarrow T_{part}$              ▷ no stopping rules: re-add partially covered examples
12: end if
13: return partially or fully covered examples $T_{add}$ to be added again to training set
```

In the case that stopping rules are not used or the re-inserting of fully covered examples is not allowed, we have to add only partially covered examples (Algorithm 5, line 10 and 12). In this case all fully covered examples have to be removed from the training data. This is not desired and may lead to inconstancies. In the second Case we re-add both partially and fully covered examples. In the last case no covered examples have to be re-added (Algorithm 5, line 7). This occurs when the percentage of the fully covered examples is greater than the skip threshold (Algorithm 5, line 2). If the set of covered examples to be re-added is empty, the current rule has to be marked as a stopping rule. Otherwise the example will be added in the training data again (Algorithm 2, line 10) and a new rule will be searched recursively.

3.2 Classifications

The training algorithm work in am iterative way and return one single model R inthe form of an ordered decision list

$$R = \langle r_1, r_2, ... \rangle \tag{18}$$

This list will be used to classify test examples. The following algorithm describes the classification process

Algorithm 6 Application of a multi-label decision list to a test example. [2]

```
Require: Test example $x$,multi-label decision list $R$
 1: $\hat{y} = (?, ?, ..., ?)$
 2: for each rule $r$ in rule list $R$ do          ▷ compute covering status for each example
 3:     if  then$r$ covers $x$
 4:         apply head of r on $\hat{y}$ if corresponding value in $\hat{y}$ is unset
 5:         if $r$ marked as stopping rule or $\hat{y}$ is complete  then
 6:             assume all remaining labels in $\hat{y}$ are negative
 7:             return $\hat{y}$
 8:         end if
 9:     end if
10: end for
11: assume all remaining labels in $\hat{y}$ are negative
12: return $\hat{y}$
```

Tabelle 3: Parameters whether re-inserting fully covered examples and the targets of rule-head to be learnedpossible combinations

Case	Predict negative head rule	Re-inserting fully covered
Case 1	False	False
Case 2	False	True
Case 2	True	False
Case 4	True	True

We iterate over all rules in the decision in the order of insertion during the training process and check if it covers the test instance. In contrast to the single-label classification the classification process for multi-label problem doesn't stop if a rule covers the test example. The head of this rule will be applied on the correspondinglabelvalue if it is unset. And then the next rule will be tested, if it fires, until no rule left. There are two stopping condition. The classification has to be stopped if the current rule is marked as stopping rule or if all labels values are set. When the classification process is ended, the value of all remaining labels will be set as negative.

3.3 Used Parameter

The proposed Separate-and-Conquer Multi-Label Rule Learner in the previous section have different parameter. In this work we will use the following parameter. The stopping thresholds θ and τ will have the value 0.01% and 0.1%. To find the best rule for a given label (Algorithm 3, line 5) we use the Ripper implementation of WekaJRip [2] To select the best global rule, we use the F1 measure as a heuristic function h.The use ofstopping rule will be enabled. A stopping rule will be marked with a star* at the head of it. For the parameters whether re-inserting fully covered examples and the targets of rule-head to be learned, we have four possiblecombinations to run this algorithm. The following table presents all these cases.

We have executed the training algorithm with the parameters mentioned below and have become the following models:

3.4 Application of the basic algorithm on the toy example

In this section we will present the Application of the training and a classification algorithm on the dataset weather with parameter of case 4 mentioned below.

3.4.1 Training

First the label vector $\hat{y}$ will be initialized with unknown values. The following table presents the training examples at the first iteration. For each label the original labels yi and the label value accessed by the learner $\hat{y}_i$ notes in its column in the form $\hat{y}_i|y_i$.

Tabelle 4: Learned rules on case 1

$tea \longleftarrow outlook = rainy, windy = TRUE$
$icecream \longleftarrow humidity >= 85.0, outlook = sunny$
$lemonade \longleftarrow \emptyset$
$play \longleftarrow \emptyset$
$dontply \longleftarrow \emptyset$

Tabelle 5: Learned rules on case 2

$tea \longleftarrow outlook = rainy, windy = TRUE$
$icecream \longleftarrow humidity >= 85.0, outlook = sunny$
$lemonade \longleftarrow \emptyset$
$play \longleftarrow \emptyset$
$dontply^{*} \longleftarrow humidity >= 90.0, outlook = sunny$
$dontply^{*} \longleftarrow tea$
$dontply^{*} \longleftarrow \emptyset$

Tabelle 6: Learned rules on case 3

$tea \longleftarrow outlook = rainy, windy = TRUE$
$\overline{tea} \longleftarrow \emptyset$
$lemonade \longleftarrow \overline{tea}$
$\overline{lemonade} \longleftarrow \emptyset$
$\overline{icecream} \longleftarrow \emptyset$
$play \longleftarrow tea = 0$
$\overline{play} \longleftarrow \emptyset$
$\overline{dontply} \longleftarrow play$
$dontply \longleftarrow \emptyset$

Tabelle 7: Learned rules on case 4

$tea \longleftarrow outlook = rainy, windy = TRUE$
$\overline{tea} \longleftarrow \emptyset$
$lemonade \longleftarrow tea = 0$
$\overline{lemonade} \longleftarrow \emptyset$
$\overline{icecream} \longleftarrow \emptyset$
$play \longleftarrow tea = 0$
$\overline{play} \longleftarrow \emptyset$
$\overline{dontply}^{*} \longleftarrow play = 1$
$dontply^{*} \longleftarrow \emptyset$

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	?|0	—	?|1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	?|0	—	?|1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	?|1	1	?|0	—	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	?|1	1	?|0	—	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	?|1	1	?|0	—	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—

For each label and possible rule-head a rule will be searched. The candidate rules are:

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$\overline{icecream} \longleftarrow temperature <= 84.0.$	[[11.0 2.0][0.0 1.0]]	Value: 0.917
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$dontply \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

For each rule a confusion matrix will be computed and thereby the heuristic value will be calculated. The rule with the best F1 measure will be then chosen. In this iteration the best global rule is:

$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0

In the case that more the one rule has the same highest heuristic value (there are two other rules that the chosen one with the heuristic value 1), the rule with that cover more positive examples will be selected. This rule will be then added to the decision list. The rule set was at the begin empty and will contains after the first iteration only this rule. The head of the best global rule will be applied to the training examples. Only three examples are covered by this rule. The new values for each label present in the column next to it in the table below. According to the modified label values the training example will be separated into three sets according to the covering status. The fully covered set is empty and only the three covered instances are stored in the partially covered set. All other example stays in the set of uncovered examples T. Using those set the set of examples to be re-inserted will be then computed. It will be containing the three covered examples. Since T_{add} is not empty the rule will not be marked a stopping rule and the instances stored on it have to be added to the trainings set T. In the next iteration all training examples will be used for leaning of rules. The next iterations of the algorithm will be presented in the Appendix. As a result of the classification algorithm we become the following decision list R.

$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$
$\overline{icecream} \longleftarrow \emptyset$
$tea \longleftarrow outlook = rainy, windy = TRUE.$
$\overline{tea} \longleftarrow \emptyset$
$lemonade \longleftarrow tea = 0.$
$play \longleftarrow icecream = 0, lemonade = 1.$
$\overline{dontply}^{*} \longleftarrow play = 1.$
$\overline{play} \longleftarrow \emptyset$
$dontply^{*} \longleftarrow \emptyset$

3.4.2 Classifications

We apply the classification algorithm with the test instance x = $x = (rainy, 65, 70, yes, ?, ?, ?, ?, ?)$. The first rule r1 doesn't cover the test example and will be skipped. The head of the second rule will be applied, since the test examples are covered: the icecream label has to set to zero. Then the next rule will be tested. The rule r3 fires and the tea label have to set to one. The four next rules do not cover the test instance and will be skipped. The three last rule covers the example x and the head of it will be applied. After application of the last rule, the classification algorithm has to be finished because the classification of thetest example is complete and the last rule is marked as a stopping rule. The remaining labels have to be assumed as negative. Since all labels are set, the instance x will be returned without any modification. The classified test instance x is $(rainy, 65, 70, yes, 0, 0, 1, 0, 1)$.

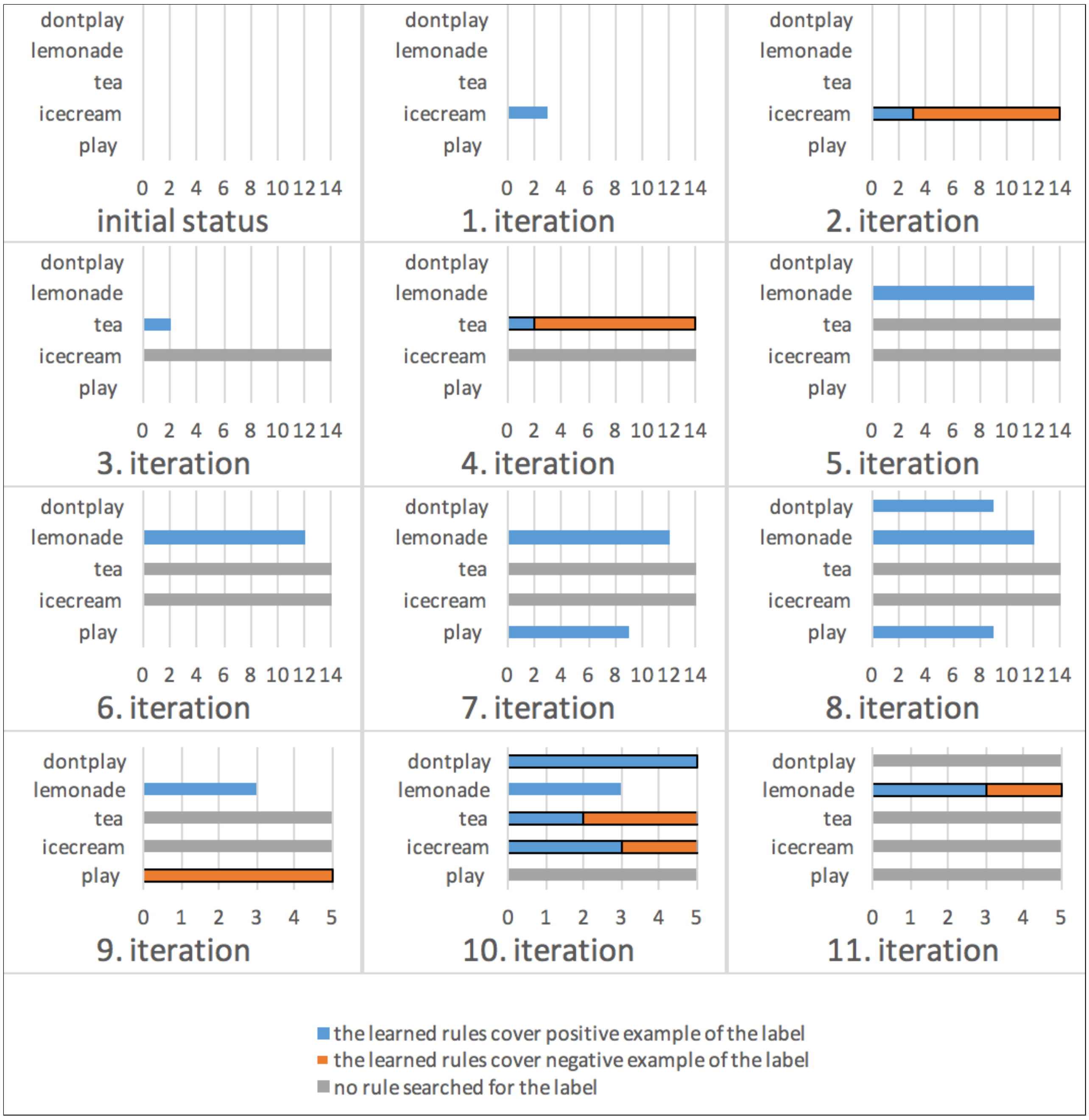
4 Extensions of the base algorithm.

In this section we present the extensions that we made to improve the Separate-and-Conquer Multi-Label Rule Learner mentioned in the previous section. We add new stopping criteria in the learning of rule for multi-labels. We try to use the covered training examples during the search of the best rule for a given label. In the case that negative head rules are allowed to be predicted, we add an option to predict a pair of negative and positive head rule instead of only one rule during a training iteration.

4.1 Searching of Label Rule Stopping Criteria

During the search of the best global rule, for every label yi a rule has to learn and then the best of them will be chosen. If a given label yi is covered by the previews learned rules for all the training examples, we don't need to learn a rule anymore to cover it. For this we add a stopping threshold Υ to prevent the searching of a rule for a given label if the percentage of example, which has an unknown value for this label value is less thanthe value of it. Using this parameter, the number of rule searching operation will fall. Thereby the training algorithm becomes more efficient and the built time of the model will be reduced.

Abbildung 1: Covering status of the different label and if a rule will be searched for it during the training of the weather dataset using parameter of case 4 and $\Upsilon = 0.05$



The training algorithm starts with unknown label values; no label is covered. In every iteration a new rule is learned that covers some of the labels. In the second iteration the label icecream is fully covered and in the next iteration no rule will be searched for it. In the 8. Iteration 9 fully covered examples are removed from the training examples that's why only 5 examples are used in the 9. Iteration.

4.2 Use of Fully covered training Examples

During the search of the Global best rule (Algorithm 3), for every label $\hat{y}_i$ a good rule is learned on a subset of the current training examples T^i (Algorithm 3, line 5). All examples which the label $\hat{y}_i$ is already set by the previous rules will be removed from the training set. The reason for this is that the label $\hat{y}_i$ is already covered and doesn't need to be leaned again. If we do not remove those examples,the global best rule founded by the learner in the first iteration will be the same in the next iterations.No matter if the fully covered examples will be re-added to the training instances or not, they will not be used in the search of a good rule.In effect, the fully covered example will be removed from the training set Ti during the learning process of a rule forall labels.

The remove of training instances during the learning of rulesis not desired and may lead to inconsistencies. Some rules will be learned only on a subset of the training data. These rules may be inconsistent with the removed examples. These deleted examples may help in learning other rules and facilitate the learning process. The use of all training examples on the rule learning process may also help to find more label-dependent rules, since the label are initialized with an unknown value in the begin and dependency can be learned only later when enough labels values are set.

We change the findBestGlobalRule algorithm, so that we use the all the current Training examplesin the learning process. The following figure presents the modified version of it

Algorithm 7 Algorithm findBestGlobalRule for finding the best current rule on the training set for any possible label in the head [2]

```
Require: example pairs $T$ and targets $B$
1: $r \leftarrow \emptyset, r.h \leftarrow \infty$                      ▷ init best rule and its heuristic value
2: for each label $y_i$ and target $t \in B$ do
3:     $r^{'} \leftarrow$ findBestRule($y_i, t, T$)    ▷ find best rule for target $y_i = t$ using all training examples
4:     $r^{'}.h = h(r^{'}, y_i, T^i)$                 ▷ heuristic value depends on target label and $T^i$
5:     if $r^{'}.h > r.h$ then
6:         $r \leftarrow r^{'}$                                ▷ replace by better rule
7:     end if
8: end for
9: return best rule $r$
```

In this version no training examples will be removed on the learning of a rule for a given label, even if the corresponding label is already set. To avoid the same rule in the next iteration will be found, we use another way to count the true tp and false positives fp, and the true tn and false negatives fn, if the label value is already set. The following table presents how these metrics will be computed.

Tabelle 8: Different method how to compute heuristic for fully covered examples

Original value	Predicted value	Set value	v_1	v_2
1	1	1	-	t_p
1	1	0	t_p	t_p
1	0	1	-	f_n
1	0	0	f_n	f_n
0	1	1	f_p	f_p
0	1	0	-	f_p
0	0	1	t_n	t_n
0	0	0	-	t_n

In the first case v_1 we use the covered example, in the computing of the heuristic values only if the new predicted value is a correction of a false set label value. In the second case v_2 we use all examples of calculating the heuristic value.

While Using the covered examplein the training process, the founded rule has to be controlled if it exists already on the current decision list or not. In the casethat a rule already exists, it will be ignored in the choice of the new best global rule.

4.3 Positive and negative head-rule combination

Often for labels that are absent (or present) on the most of the training examples, only one rule will be learned that set this label as negative or only rules will be learned to cover the absence (or the presence) of it.To solve this problem, we have tried to learn the opposite head rule in the next iteration, once a positive or a negative rule is learned the first time for a given label. The following rule set is learned with this strategy

$$\begin{array}{l}
\hline
icecream \longleftarrow outlook = sunny, humidity >= 82.5. \\
\overline{icecream} \longleftarrow \emptyset \\
tea \longleftarrow outlook = rainy, windy = TRUE. \\
\overline{tea} \longleftarrow \emptyset \\
lemonade \longleftarrow tea = 0. \\
\overline{lemonade} \longleftarrow \emptyset \\
play \longleftarrow icecream = 0, lemonade = 1. \\
\overline{play} \longleftarrow \emptyset \\
\overline{dontply} \longleftarrow play = 1. \\
dontply \longleftarrow \emptyset
\end{array}$$

After learning a positive rule for the label ice cream in the first iteration. In the second iteration only one rule will be searched that learn a negative head rule for this label. The problem with this method is that the opposite head rule learned in next iteration may have a bad heuristic value that result in a bad model. We have used another strategy. Instead of learning a single rule in each training iterations a pair of rule will be learned. To find the best global rule, for every label y_i two rule with positive and negative head have to be leaned. A heuristic value will be calculated for the two combination of rules r_1 the r_2' or r_2 then r_1'. The combination of rule with the highest heuristic value will be then chosen as the best global rule. .

4.3.1 Confusion matrix of a pair of rule

To compute the heuristic of a couple of rule. The first rule head has to be applied to the training examples and then the heuristic of the second rule computing on it.
for given label y_i the rule r_1 with a positive head (or negative) was learned. Let C_1 be the confusion matrix of r_1.

$$C_1 = \begin{pmatrix} t_p^1 & f_n^1 \\ f_p^1 & t_n^1 \end{pmatrix} \tag{19}$$

the head of the rule will be applied to the $t_p 1 + f_p 1$ covered example by the rule r_1. Then a negative (or positive) head rule r_2 will be learned on the remain $f : n1 + t_n 1$ uncovered training examples. Let C_1 be the confusion matrix of r_1.

$$C_1 = \begin{pmatrix} t_p^2 & f_n^2 \\ f_p^2 & t_n^2 \end{pmatrix} \tag{20}$$

We define the confusion matrix of the pair of rule $(r1, r2) C_{12}$ with:

$$C_{12} = \begin{pmatrix} t_p^2 + t_p^1 & f_n^2 + f_p^1 \\ f_p^2 & t_n^2 \end{pmatrix} \tag{21}$$

4.3.2 Application on the weather Dataset

All labels are initialized with unknown value. During the search of the best global rule, each label y_i a positive and negative head rule have to be learned. For the two learned rules an opposite head rule is learned after applying the head of it. A confusion matrix of the pairs of rules will be then computed and applied to the heuristic function h. The couple of rules that have the highest heuristic value will be chosen as the best Global rule. In the first iteration on the weather dataset the following rules pairs are learned:

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$play \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$\overline{play} \longleftarrow \emptyset$	[[11.0 1.0][2.0 0.0]]	Value: 0.88
$\overline{icecream} \longleftarrow temperature <= 84.0.$	[[11.0 2.0][0.0 1.0]]	Value: 0.917
$icecream \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$tea \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{lemonade} \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$dontply \longleftarrow \emptyset$	[[11.0 1.0][2.0 0.0]]	Value: 0.88
$dontply \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$\overline{dontply} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923

the best pair of rule is:

$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

This couple of rule is added to the Rule decision list and the both positive head and negative will be applied to the covered training

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	?|0	0	?|1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	?|0	0	?|1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—

In the next iterations other pairs of rules will be searched until no examples left uncovered. In the following is the found rules set.

5 Evaluation

In this section we will evaluate the proposed extensions in the previous section. We will compare the learned models and the predictive performances using these different extensions to the models learned by the base algorithm.

For the evaluation we use the following multi-label data sets from the Java Library for Multi-Label Learning Mulan [9]

Tabelle 9: Multi-label data sets for the evaluations [9]

name	domain	instances	nominal	numeric	labels	cardinality	density	distinct
emotions	music	593	0	72	6	1.869	0.311	27
genbase	biology	662	1186	0	27	1.252	0.046	32
yeast	biology	2417	0	103	14	4.237	0.303	198

In this section we will use the following abbreviation:

$SeCo$ for the original separate-and-conquer multi-label rule learner.

$Seco_{v1}, Seco_{v2}$ for the using of the fully covered examples in learning new rule and computing the heuristic value using the two methods v_1 and v_2.

$SeCo_{1-0}$ for the learning of the opposite head rule when the other head rule was found for the first time.

$SeCo_{com}$ for learning a pair of negative and positive rules during one iteration.

Tabelle 10: Statistic of the separate-and-conquer multi-label rule learner with the different proposed options

dataset/ approach	number of rule	rule Length	part. Label rule	fully Label rule	non Label rule	label Conditions
Emotions						
Seco	19,9	1,0124	0,0185	0,0458	0,935	0,067
SeCo_v1	19,8	1,412	0,0178	0,0037	0,978	0,0158
SeCo v2	21,2	1,3123	0.0	0,0995	0,8993	0,0032
SeCo_1->0	22,9	1,25	0,021	0,12	0,77	0,124
Seco_com	45	1,512	0,0175	0,0087	0,97	0,0185
Genbase						
Seco	52,9	0,569	0.0	0,0	1.0	0.0
SeCo_v1	39	1,917	0,0	0,0	1,0	0,0
SeCo v2	38,5	1,0386	0.0	0,0	1.0	0,0
SeCo_1->0	53,4	0,559	0,0	0,0	1,0	0,0
Seco_com	53,4	0,559	0,0	0,0	1.0	0.0
Yeast						
Seco	42,9	2,149	0,015	0,1118	0,87	0,0278
SeCo_v1	39	1,917	0,0	0,0	1,0	0,0
SeCo v2	34,4	1,465	0,0	0,0	1.0	0,0
SeCo_1->0	48,9	1,727	0,025	0,0058	0,96	0,018
Seco_com	53,4	0,559	0,0	0,0	1.0	0.0

Using the fully covered examples in the predication of new rule using both of method v1 and v2 lead to the learning of smaller models. But only using the method v_2 more label-dependency rules are learned. The model found when we use the combinations of the head rules are bigger (heights number of rules and rule length) and learn more label-dependent rule that the other approaches.

Tabelle 11: The Predictive performances of the separate-and-conquer multi-label rule learner with the different proposed options

dataset/ approach	Hamming Acc.	subset Acc.	Mi. Prec	Mi. Recall	Mi. F1	Ma. Prec	Ma. Recall	Ma. F1
Emotions								
Seco	0,77	0.195	0.71	0.478	0.559	0.537	0.45	0.473
SeCo_v1	0,76	0.18	0.68	0.426	0.52	0.56	0.39	0.44
SeCo v2	0,76	0.17	0.685	0.44	0.53	0.56	0.40	0.45
SeCo_1->0	0,76	0.23	0.67	0.49	0.57	0.59	0.469	0.5
Seco_com	0,76	0.20	0.64	0.57	0.60	0.65	0.5	0.5
Genbase								
Seco	0,99	0.93	0.99	0.95	0.97	0.93	0.93	0.93
SeCo_v1	0,98	0.59	0.99	0.60	0.74	0.67	0.67	0.67
SeCo v2	0,99	0.82	0.99	0.78	0.87	0.69	0.69	0.69
SeCo_1->0	0,99	0.97	0.99	0.98	0.98	0.95	0.95	0.95
Seco_com	0,99	0.97	0.99	0.98	0.98	0.95	0.96	0,95
Yeast								
Seco	0,785	0.075	0.695	0.516	0.59	0.33	0.27	0.28
SeCo_v2	0,78	0.067	0.71	0.48	0.572	0.358	0.25	0.26
SeCo v2	0,78	0.067	0.71	0.48	0.572	0.358	0.25	0.26
SeCo_1->0	0,78	0.072	0.698	0.49	0.575	0.37	0.268	0.0.28
Seco_com	0,78	0.072	0.70	0.49	0.57	0.37	0.26	0,28

The predicted performance of the rule learner using the combinations of head rules is the best one. Also using the fully covered examples in the predication of new rule using improved the performance of the classification algorithm.

6 Conclusion

In this work we have present a formalization of the multi-label classification and the different type of multi-label dependency. Conditional and unconditional dependencies may exist between different labels and can be used to improve the classification task. We have present a multi-label separate-and-conquer rule Learner that can learn such dependency. We have extended this learner with some option and then test them. Using these extensions, the predicted performance can be improved and we learn more label-dependent rules.

Literatur

[1] Jesse Read, Bernhard Pfahringer, Geoff Holmes, Eibe Frank. Classifier Chains for Multi-label Classification

[2] Eneldo LozaMencía, Frederik Janssen, Learning Rules for Multi-label Classification: a Stacking and a Separate-and-Conquer Approach,2016

[3] Eneldo LozaMencía, Johannes Furnkranz. Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain, 2010

[4] Daniel Hsu, Sham M. Kakade, John Langford,Tong Zhang.Multi-Label Prediction via Compressed Sensing,2009

[5] Frederik Janssen. Heuristic Rule Learning 2012

[6] Johannes Fürnkranz. Separate-and-conquer rule learning Artificial Intelligence Review 13: 3–54, 1999

[7] Krzysztof Dembczynski, Willem Waegeman, Weiwei Cheng, Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification, June 2012

[8] Weka Classifiers Rules: Class JRip, `http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/JRip.html`

[9] Mulan: A Java Library for Multi-Label Learning: Datasets, `http://mulan.sourceforge.net/datasets-mlc.html`

Appendix

Application of the basic algorithm on the toy example

1. The used dataset for test :

outlook	temperature	humidity	windy	play	icecream	tea	lemonade	dontplay
rainy	65	70	TRUE	0	0	1	0	1
rainy	71	91	TRUE	0	0	1	0	1
sunny	85	85	FALSE	0	1	0	1	1
sunny	80	90	TRUE	0	1	0	1	1
sunny	72	95	FALSE	0	1	0	1	1
sunny	69	70	FALSE	1	0	0	1	0
sunny	75	70	TRUE	1	0	0	1	0
overcast	83	86	FALSE	1	0	0	1	0
overcast	64	65	TRUE	1	0	0	1	0
overcast	72	90	TRUE	1	0	0	1	0
overcast	81	75	FALSE	1	0	0	1	0
rainy	70	96	FALSE	1	0	0	1	0
rainy	68	80	FALSE	1	0	0	1	0
rainy	75	80	FALSE	1	0	0	1	0

2. The used parameters for test :

   2.1 the remaining instances percentage θ : 0.01

   2.2 the skip threshold percentage τ : 0.1

   2.3 heuristic h : FMeasure with beta=1.0

   2.4 learn negativ head rules : true

   2.5 use stoping rule : true

   2.6 re-inserting fully covered examples : true

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	?|0	—	?|1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	?|0	—	?|1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	?|1	1	?|0	—	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	?|1	1	?|0	—	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	?|1	1	?|0	—	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	?|0	—	?|0	—	?|1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$\overline{icecream} \longleftarrow temperature <= 84.0.$	[[11.0 2.0][0.0 1.0]]	Value: 0.917
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$dontply \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	?|0	0	?|1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	?|0	0	?|1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$icecream \longleftarrow \emptyset$	[[0.0 11.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	0	—	?|1	1	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	0	—	?|1	1	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	0	—	?|0	—	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	0	—	?|0	—	?|1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow icecream = 0.$	[[9.0 2.0][0.0 3.0]]	Value: 0.9
$\overline{icecream} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$icecream \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow icecream = 0.$	[[9.0 2.0][0.0 3.0]]	Value: 0.9
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	?|0	0	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	?|0	0	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	?|0	0	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow icecream = 0.$	[[9.0 2.0][0.0 3.0]]	Value: 0.9
$\overline{icecream} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$icecream \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow \emptyset$	[[0.0 12.0][0.0 0.0]]	Value: 0.0
$\overline{lemonade} \longleftarrow tea = 1.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow icecream = 0.$	[[9.0 2.0][0.0 3.0]]	Value: 0.9
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	0	—	?|1	1	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	0	—	?|1	1	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	0	—	?|1	1	?|1	—
sunny	69	70	FALSE	?|1	—	0	—	0	—	?|1	1	?|0	—
sunny	75	70	TRUE	?|1	—	0	—	0	—	?|1	1	?|0	—
overcast	83	86	FALSE	?|1	—	0	—	0	—	?|1	1	?|0	—
overcast	64	65	TRUE	?|1	—	0	—	0	—	?|1	1	?|0	—
overcast	72	90	TRUE	?|1	—	0	—	0	—	?|1	1	?|0	—
overcast	81	75	FALSE	?|1	—	0	—	0	—	?|1	1	?|0	—
rainy	70	96	FALSE	?|1	—	0	—	0	—	?|1	1	?|0	—
rainy	68	80	FALSE	?|1	—	0	—	0	—	?|1	1	?|0	—
rainy	75	80	FALSE	?|1	—	0	—	0	—	?|1	1	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$icecream \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$tea \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{lemonade} \longleftarrow tea = 1.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0
$\overline{dontply} \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	0	—	1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	0	—	1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	0	—	1	—	?|1	—
sunny	69	70	FALSE	?|1	1	0	—	0	—	1	—	?|0	—
sunny	75	70	TRUE	?|1	1	0	—	0	—	1	—	?|0	—
overcast	83	86	FALSE	?|1	1	0	—	0	—	1	—	?|0	—
overcast	64	65	TRUE	?|1	1	0	—	0	—	1	—	?|0	—
overcast	72	90	TRUE	?|1	1	0	—	0	—	1	—	?|0	—
overcast	81	75	FALSE	?|1	1	0	—	0	—	1	—	?|0	—
rainy	70	96	FALSE	?|1	1	0	—	0	—	1	—	?|0	—
rainy	68	80	FALSE	?|1	1	0	—	0	—	1	—	?|0	—
rainy	75	80	FALSE	?|1	1	0	—	0	—	1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow icecream = 0, lemonade = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$icecream \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$tea \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{lemonade} \longleftarrow \emptyset$	[[2.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[0.0 2.0][0.0 0.0]]	Value: 0.0
$\overline{dontply} \longleftarrow icecream = 0, lemonade = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$play \longleftarrow icecream = 0, lemonade = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0
$play \longleftarrow icecream = 0, lemonade = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	0	—	1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	0	—	1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	0	—	1	—	?|1	—
sunny	69	70	FALSE	1	—	0	—	0	—	1	—	?|0	0
sunny	75	70	TRUE	1	—	0	—	0	—	1	—	?|0	0
overcast	83	86	FALSE	1	—	0	—	0	—	1	—	?|0	0
overcast	64	65	TRUE	1	—	0	—	0	—	1	—	?|0	0
overcast	72	90	TRUE	1	—	0	—	0	—	1	—	?|0	0
overcast	81	75	FALSE	1	—	0	—	0	—	1	—	?|0	0
rainy	70	96	FALSE	1	—	0	—	0	—	1	—	?|0	0
rainy	68	80	FALSE	1	—	0	—	0	—	1	—	?|0	0
rainy	75	80	FALSE	1	—	0	—	0	—	1	—	?|0	0

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0
$play \longleftarrow \emptyset$	[[0.0 5.0][0.0 0.0]]	Value: 0.0
$\overline{icecream} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$icecream \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$tea \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{lemonade} \longleftarrow \emptyset$	[[2.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[0.0 2.0][0.0 0.0]]	Value: 0.0
$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0
$play \longleftarrow icecream = 0, lemonade = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0

1. the remaining training set (5 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	0	0	—	1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	0	0	—	1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	0	1	—	0	—	1	—	?|1	—
sunny	80	90	TRUE	?|0	0	1	—	0	—	1	—	?|1	—
sunny	72	95	FALSE	?|0	0	1	—	0	—	1	—	?|1	—
sunny	69	70	FALSE	1	—	0	—	0	—	1	—	0	—
sunny	75	70	TRUE	1	—	0	—	0	—	1	—	0	—
overcast	83	86	FALSE	1	—	0	—	0	—	1	—	0	—
overcast	64	65	TRUE	1	—	0	—	0	—	1	—	0	—
overcast	72	90	TRUE	1	—	0	—	0	—	1	—	0	—
overcast	81	75	FALSE	1	—	0	—	0	—	1	—	0	—
rainy	70	96	FALSE	1	—	0	—	0	—	1	—	0	—
rainy	68	80	FALSE	1	—	0	—	0	—	1	—	0	—
rainy	75	80	FALSE	1	—	0	—	0	—	1	—	0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0
$play \longleftarrow \emptyset$	[[0.0 5.0][0.0 0.0]]	Value: 0.0
$\overline{icecream} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$icecream \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$tea \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{lemonade} \longleftarrow \emptyset$	[[2.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[0.0 2.0][0.0 0.0]]	Value: 0.0
$\overline{dontply} \longleftarrow \emptyset$	[[0.0 5.0][0.0 0.0]]	Value: 0.0
$dontply \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0
$play \longleftarrow icecream = 0, lemonade = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{play} \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0

1. the remaining training set (5 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	0	—	0	—	1	—	?|0	—	?|1	1
rainy	71	91	TRUE	0	—	0	—	1	—	?|0	—	?|1	1
sunny	85	85	FALSE	0	—	1	—	0	—	1	—	?|1	1
sunny	80	90	TRUE	0	—	1	—	0	—	1	—	?|1	1
sunny	72	95	FALSE	0	—	1	—	0	—	1	—	?|1	1

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$play \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{icecream} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$icecream \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{tea} \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$tea \longleftarrow \emptyset$	[[0.0 0.0][0.0 0.0]]	Value: 0.0
$\overline{lemonade} \longleftarrow \emptyset$	[[2.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[0.0 2.0][0.0 0.0]]	Value: 0.0
$\overline{dontply} \longleftarrow \emptyset$	[[0.0 5.0][0.0 0.0]]	Value: 0.0
$dontply \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0

3. The best rule is :

Rule	confusion matrix	$F1$ measure
$dontply \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[11.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0
$play \longleftarrow icecream = 0, lemonade = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{play} \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0
$dontply \longleftarrow \emptyset$	[[5.0 0.0][0.0 0.0]]	Value: 1.0

Application of Separate-and-Conquer Multi-Label Rule Learner on the weather dataset using positive and negative head-rule combination

1. The used dataset for test :

outlook	temperature	humidity	windy	play	icecream	tea	lemonade	dontplay
rainy	65	70	TRUE	0	0	1	0	1
rainy	71	91	TRUE	0	0	1	0	1
sunny	85	85	FALSE	0	1	0	1	1
sunny	80	90	TRUE	0	1	0	1	1
sunny	72	95	FALSE	0	1	0	1	1
sunny	69	70	FALSE	1	0	0	1	0
sunny	75	70	TRUE	1	0	0	1	0
overcast	83	86	FALSE	1	0	0	1	0
overcast	64	65	TRUE	1	0	0	1	0
overcast	72	90	TRUE	1	0	0	1	0
overcast	81	75	FALSE	1	0	0	1	0
rainy	70	96	FALSE	1	0	0	1	0
rainy	68	80	FALSE	1	0	0	1	0
rainy	75	80	FALSE	1	0	0	1	0

2. The used parameters for test :

   2.1 the remaining instances percentage θ : 0.01

   2.2 the skip threshold percentage τ : 0.1

   2.3 heuristic h : FMeasure with beta=1.0

   2.4 learn negativ head rules : true

   2.5 use stoping rule : true

   2.6 re-inserting fully covered examples : true

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	?|0	0	?|1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	?|0	0	?|1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	?|0	—	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	?|0	0	?|0	—	?|1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$play \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$\overline{play} \longleftarrow \emptyset$	[[11.0 1.0][2.0 0.0]]	Value: 0.88
$\overline{icecream} \longleftarrow temperature <= 84.0.$	[[11.0 2.0][0.0 1.0]]	Value: 0.917
$icecream \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$tea \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{lemonade} \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow temperature <= 84.0, humidity <= 90.0.$	[[8.0 2.0][1.0 3.0]]	Value: 0.842
$dontply \longleftarrow \emptyset$	[[11.0 1.0][2.0 0.0]]	Value: 0.88
$dontply \longleftarrow humidity >= 82.5, outlook = sunny.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$\overline{dontply} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923

3. The best rule is :

$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	—	0	—	1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	—	1	—	?|0	0	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	—	1	—	?|0	0	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	—	1	—	?|0	0	?|1	—	?|1	—
sunny	69	70	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
sunny	75	70	TRUE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	83	86	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	64	65	TRUE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	72	90	TRUE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
overcast	81	75	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
rainy	70	96	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
rainy	68	80	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—
rainy	75	80	FALSE	?|1	—	0	—	?|0	0	?|1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$play \longleftarrow icecream = 0.$	[[9.0 2.0][0.0 3.0]]	Value: 0.9
$\overline{play} \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$\overline{tea} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$tea \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$\overline{lemonade} \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$\overline{dontply} \longleftarrow icecream = 0.$	[[9.0 2.0][0.0 3.0]]	Value: 0.9
$dontply \longleftarrow \emptyset$	[[12.0 0.0][2.0 0.0]]	Value: 0.923
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$\overline{dontply} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923

3. The best rule is :

$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

Application of the algorithm on the toy example

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	?|0	0	0	—	1	—	?|0	—	?|1	—
rainy	71	91	TRUE	?|0	0	0	—	1	—	?|0	—	?|1	—
sunny	85	85	FALSE	?|0	0	1	—	0	—	?|1	—	?|1	—
sunny	80	90	TRUE	?|0	0	1	—	0	—	?|1	—	?|1	—
sunny	72	95	FALSE	?|0	0	1	—	0	—	?|1	—	?|1	—
sunny	69	70	FALSE	1	—	0	—	0	—	?|1	—	?|0	—
sunny	75	70	TRUE	1	—	0	—	0	—	?|1	—	?|0	—
overcast	83	86	FALSE	1	—	0	—	0	—	?|1	—	?|0	—
overcast	64	65	TRUE	1	—	0	—	0	—	?|1	—	?|0	—
overcast	72	90	TRUE	1	—	0	—	0	—	?|1	—	?|0	—
overcast	81	75	FALSE	1	—	0	—	0	—	?|1	—	?|0	—
rainy	70	96	FALSE	1	—	0	—	0	—	?|1	—	?|0	—
rainy	68	80	FALSE	1	—	0	—	0	—	?|1	—	?|0	—
rainy	75	80	FALSE	1	—	0	—	0	—	?|1	—	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{play} \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$play \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923
$play \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{play} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow tea = 1.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{dontply} \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$dontply \longleftarrow icecream = 1.$	[[3.0 0.0][2.0 9.0]]	Value: 0.75
$\overline{dontply} \longleftarrow \emptyset$	[[12.0 2.0][0.0 0.0]]	Value: 0.923

3. The best rule is :

$play \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{play} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$play \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{play} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	0	—	0	—	1	—	0	—	?|1	—
rainy	71	91	TRUE	0	—	0	—	1	—	0	—	?|1	—
sunny	85	85	FALSE	0	—	1	—	0	—	?|1	1	?|1	—
sunny	80	90	TRUE	0	—	1	—	0	—	?|1	1	?|1	—
sunny	72	95	FALSE	0	—	1	—	0	—	?|1	1	?|1	—
sunny	69	70	FALSE	1	—	0	—	0	—	?|1	1	?|0	—
sunny	75	70	TRUE	1	—	0	—	0	—	?|1	1	?|0	—
overcast	83	86	FALSE	1	—	0	—	0	—	?|1	1	?|0	—
overcast	64	65	TRUE	1	—	0	—	0	—	?|1	1	?|0	—
overcast	72	90	TRUE	1	—	0	—	0	—	?|1	1	?|0	—
overcast	81	75	FALSE	1	—	0	—	0	—	?|1	1	?|0	—
rainy	70	96	FALSE	1	—	0	—	0	—	?|1	1	?|0	—
rainy	68	80	FALSE	1	—	0	—	0	—	?|1	1	?|0	—
rainy	75	80	FALSE	1	—	0	—	0	—	?|1	1	?|0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{lemonade} \longleftarrow tea = 1.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$lemonade \longleftarrow tea = 0.$	[[12.0 0.0][0.0 2.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$dontply \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$dontply \longleftarrow play = 0.$	[[5.0 0.0][0.0 9.0]]	Value: 1.0
$\overline{dontply} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

3. The best rule is :

$\overline{lemonade} \longleftarrow tea = 1.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$play \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{play} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow tea = 1.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

1. the remaining training set (14 examples) :

outlook	temperature	humidity	windy	play	*	icecream	*	tea		lemonade	*	dontplay	*
rainy	65	70	TRUE	0	—	0	—	1	—	0	—	?|1	1
rainy	71	91	TRUE	0	—	0	—	1	—	0	—	?|1	1
sunny	85	85	FALSE	0	—	1	—	0	—	1	—	?|1	1
sunny	80	90	TRUE	0	—	1	—	0	—	1	—	?|1	1
sunny	72	95	FALSE	0	—	1	—	0	—	1	—	?|1	1
sunny	69	70	FALSE	1	—	0	—	0	—	1	—	0	—
sunny	75	70	TRUE	1	—	0	—	0	—	1	—	0	—
overcast	83	86	FALSE	1	—	0	—	0	—	1	—	0	—
overcast	64	65	TRUE	1	—	0	—	0	—	1	—	0	—
overcast	72	90	TRUE	1	—	0	—	0	—	1	—	0	—
overcast	81	75	FALSE	1	—	0	—	0	—	1	—	0	—
rainy	70	96	FALSE	1	—	0	—	0	—	1	—	0	—
rainy	68	80	FALSE	1	—	0	—	0	—	1	—	0	—
rainy	75	80	FALSE	1	—	0	—	0	—	1	—	0	—

2. The candidate rules are :

Rule	confusion matrix	$F1$ measure
$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$dontply \longleftarrow play = 0.$	[[5.0 0.0][0.0 9.0]]	Value: 1.0
$\overline{dontply} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

3. The best rule is :

$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0

4. The rule set :

Rule	confusion matrix	$F1$ measure
$icecream \longleftarrow outlook = sunny, humidity >= 82.5.$	[[3.0 0.0][0.0 11.0]]	Value: 1.0
$\overline{icecream} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$tea \longleftarrow outlook = rainy, windy = TRUE.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$\overline{tea} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$play \longleftarrow icecream = 0, tea = 0.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$\overline{play} \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{lemonade} \longleftarrow tea = 1.$	[[2.0 0.0][0.0 12.0]]	Value: 1.0
$lemonade \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0
$\overline{dontply} \longleftarrow play = 1.$	[[9.0 0.0][0.0 5.0]]	Value: 1.0
$dontply \longleftarrow \emptyset$	[[14.0 0.0][0.0 0.0]]	Value: 1.0