Chapter 5
Adaptation-Based Classifiers

Abstract While data transformation is a relatively straightforward way to do multilabel classification through traditional classifiers, an alternative approach based on adapting those classifiers to tackle the original multilabeled data also has been also explored. This chapter aims to introduce many of these method adaptations. Most of them rely on traditional algorithms based on the trees, neural networks, instance-based learning, etc. A general overview of them is provided in Sect. 5.1. Then, about thirty different proposals are detailed in Sects. 5.2–5.7, grouped according to the type of model they are founded on. A selection of four algorithms are experimentally tested in Sect. 5.8. Some final remarks are provided in Sect. 5.9.

5.1 Overview

Unlike the problem transformation-based methods presented in the previous chapter, those following the adaptation approach aim to prepare existing classification algorithms to make them able to manage instances with several outputs. The changes that must be introduced in the algorithms can be quite simple or really difficult, depending on the nature of the original method and also the way the existence of several labels is going to be considered.

There are proposals of multilabel classifiers based on trees, neural networks, vector support machines, instance-based learning techniques, and probabilistic methods, among others. About thirty of them are portrayed in the following five sections, each one devoted to one of the just-mentioned categories. An additional section enumerates other types of proposals, such as the ones based on ant colonies or genetic algorithms.

In the second part of this chapter, the behavior of four MLC adaptation-based methods is experimentally checked. These algorithms are representatives of the main four types of models: trees, neural networks, support vector machines, and instance-based learning.

F. Herrera et al., *Multilabel Classification*,
DOI 10.1007/978-3-319-41111-8_5

5.2 Tree-Based Methods

Decision trees (DT) are among the easiest to understand classification models. Despite their apparent simplicity, some DT algorithms, such as C4.5 [24], yield a performance that makes them competitive against other learning approaches. The following are some of the multilabel classification methods based on DT proposed on the literature.

5.2.1 Multilabel C4.5, ML-C4.5

The research conducted in [10] aimed to classify genes according to their function, taking into account that the same gene can intervene in several functions. The proposed solution is founded on the well-known C4.5 algorithm, appropriately modified to deal with several labels at once. The two key points of the adapted method are as follows:

- The leaves of the tree contain samples that are associated with a set of labels, instead of to one class only.
- The original entropy measure is adjusted to take into consideration the non-membership probability of the instances to a certain label.

The iterative process to build the tree partitions the instances according to the value of a certain attribute, computing the weighted sum of the entropies of each potential subset of labels. If a sample is associated with more than one label, then it is accumulated several times into this weighted sum.

Since each leaf represents a set of labels, this impacts both the process of labeling each node in the tree and also the further pruning task inherent to C4.5.

5.2.2 Multilabel Alternate Decision Trees, ADTBoost.MH

ADTs (*Alternate Decision Trees*) can be seen as a generalization of traditional decision trees. They were introduced in [15], aiming to propose an alternative way to techniques such as boosting in order to improve the precision of tree-based classifiers.

Taking as reference the ADT idea, and with the goal of extending the AdaBoost.MH model presented in [25], ADTBoost.MH is proposed in [11]. This is an ADT-based model adapted to consider the presence of multiple labels per instance. For doing so, the samples are decomposed following the OVA strategy.

Since ADTBoost.MH trains several models, each one of them being an ADT, it is an MLC method that could be included in the ensemble category as well (see Chap. 6).

5.2.3 Other Tree-Based Proposals

Proposed in [34], ML-Tree is an algorithm to induce classification trees following an hierarchical approach. The original data are seen as an hierarchy, and it is decomposed into several simpler problems using an OVA strategy. An SVM classifier is trained for each case. This way the full dataset at the root node is partitioned into several subsets going to the child nodes. This process is recursively repeated, generating the tree that represents the data hierarchy. As the previous one, the obtained model can be considered as an ensemble of simpler classifiers.

Once the tree has been completed, each leaf provides the set of predicted labels. From this information, a predictive label vector is produced, aiming to model the relationships among labels. Those that frequently appear together are supposed to have some relation level, estimated from the concurrence frequency. This automatic discovery of label correlations is used to improve the predictive performance.

As the previous proposal, LaCova [3] is an algorithm which also relies on a tree to recursively divide the original multilabel data. This divide-and-conquer approach horizontally grows the tree deciding which input feature provides more information and vertically grows it by grouping labels into subnodes.

For each node, a label covariance matrix is built, depending on which the tree will be expanded vertically or horizontally. The dependency among labels in each node is locally assessed, instead of elaborating a global dependency model. When a new vertical split is decided, the instances contained in each new subnode are processed using a BR or LP classifier, subject to the obtained dependency information.

5.3 Neuronal Network-Based Methods

Artificial neural networks (ANNs) in general, and particularly Radial Basis Function Networks (RBFNs), have proven their effectiveness in classification problems, as well as in regression and time series prediction. As a consequence, the adaptation of ANNs to accomplish MLC tasks is a recurrent topic in the literature. The goal of this section is to provide a succinct description of several ANN-based multilabel algorithms.

5.3.1 Multilabel Back-Propagation, BP-MLL

Considered as the first multilabel-adapted ANN, BP-MLL [36] is founded on one of the simplest ANN models, as is the perceptron. The training algorithm chosen for building the model is also well known, back-propagation.

The key aspect in BP-MLL is the introduction of a new error function used while training the ANN, and computed taking into account the fact that each sample contains several labels. Specifically, this new function penalizes the predictions including labels which are not truly relevant for the processed instance.

In BP-MLL, the input layer has as many neurons as input attributes there are in the MLD. The number of units in the output layer is determined by the number of labels considered. The amount of neurons in the hidden layer is also influenced by the number of labels, and they use a sigmoid activation function.

The BP-MLL algorithm produces a label ranking as result while classifying new instances. To decide which labels will be predicted as relevant, a parameter in charge of adjusting the cut threshold has to be set. Configuring this parameter is the main difficult in using BP-MLL.

The proposal made in [19], called I-BP-MLL, overtakes the aforementioned difficulty. To do so, the threshold is automatically adjusted as the committed error is computed during the learning process. This approach produces a custom threshold for each label, instead of a global threshold as BP-MLL does.

5.3.2 Multilabel Radial Basis Function Network, ML-RBF

The algorithm proposed in [37] is an specialized method for designing RBFNs adapted to work with multilabel data. It takes the samples associated with each label and then executes a K-means clustering as many times as labels there are. This way the centers of the RBFs are set. The number of clusters by label is controlled by a α parameter. Depending on the value assigned to α, the number of units in the hidden layer will be equal or larger than the number of labels in the MLD.

ML-RBF uses the SVD (*Singular Value Decomposition*) method to adjust the weights of the connections to the output units, minimizing the squared sum of the computed error in each training iteration. The activation of all neurons is set to 1 and a bias is defined for each label.

In addition to the α parameter, ML-RBF needs another two parameters named σ and μ. The former, usual in RBFNs, controls the width of the unit. ML-RBF computes it by means of an equation in which the distance between each pair of patterns and a scaling factor intervene. The latter sets the scaling factor.

There are two ML-RBF variations, called FPSO-MLRBF and FSVD-MLRBF [2], designed as hybridization of techniques which aim to improve the results produced by the original algorithm. FPSO-MLRBF relies on a fuzzy PSO (*particle swarm optimization*) method with the goal to set the number of units in the hidden layer, as well as to optimize the weights between the hidden and the output layers. FSVD-MLRBF resorts to the use of a fuzzy K-means along with SVD with same goal.

5.3.3 *Canonical Correlation Analysis and Extreme Learning Machine, CCA-ELM*

Introduced in [20], this algorithm suggests a methodology for adapting an Extreme Learning Machine (ELM) to be able to deal with multilabeled samples. ELMs are a type of ANN with only a hidden layer, characterized by the speed it can learn. This trait makes it ideal to process multilabel datasets.

The proposed method starts using Canonical Correlation Analysis (CCA) in order to detect potential correlations among input attributes and labels. As a result, a new space which combines inputs and labels is generated, being used to train the ELM. Once trained, the ELM can be used to obtain predictions. Those need to be back-translated by applying the inverse transformation to the original solution space, thus obtaining the predicted set of labels.

5.4 Vector Support Machine-Based Methods

Vector Support Machines have been traditionally applied to solve binary classification problems. As other techniques, they have evolved over time, being extended to face other kind of tasks such as multiclass and multilabel classification. In this section, several of the MLC methods based on the SVMs are outlined.

5.4.1 *MODEL-x*

The target task of the authors in [5] was to label natural landscapes. Each scene can contain several objects at once and, as a consequence, it can be labeled with more than one class. Some scene labels are urban, beach, field, mountain, beach + field, field + mountain, etc.

Since SVMs tend to have a good behavior while dealing with images, the authors selected this kind of model as underlying classifier. The proposed method, called MODEL-x, trains a SVM per label using a novel approach named *cross-training*. This technique takes the samples having several labels as positive cases while training every individual model, instead of negative ones as do the algorithms which consider each label combination as a different class. As many other MLC proposals, MODEL-x can be also considered as an ensemble.

5.4.2 *Multilabel SVMs Based on Ranking, Rank-SVM and SCRank-SVM*

As the authors of Rank-SVM state in [13], binary classifiers are not the best option when some correlations among labels exist, since full independence between them is assumed by most binary-based methods. To alleviate this problem Rank-SVM, a direct approach based on SVM principles, relies in a new metric to be minimized while training, AHL. This a lineal approximation of Hamming Loss.

Once the SVM-like model has been trained to produce the label ranking, a specific function is used to adjust the cutting threshold from which the predicted labelset, as a subset of the full ranking, is extracted.

Although Rank-SVM takes label correlations into account, thus it theoretically should perform better than pure binary models, experimental tests show that its behavior is similar when working with high-dimensional MLDs.

These two proposals are based on the Rank-SVM, aiming to improve their efficiency and performance. The goal of Rank-CVM [35] is to reduce the computational complexity of Rank-SVM. To do so, they decided to use a CVM (*core vector machine*) instead of an SVM. The analytical solution of CVMs is immediate, and much more efficient than that of SVMs. The result is an MLC classifier with similar predictive performance to Rank-SVM but an order of magnitude more efficient.

The goal of the authors of SCRank-SVM [33] also was to improve the efficiency of Rank-SVM, as well as its performance as MLC classifier. The proposal includes reformulating the calculus of the decision boundary in the underlying SVMs, simplifying some of the existent constraints to maximize the margin. In the process, the authors get rid of one of the usual SVM parameters, reducing the complexity of computations.

5.5 Instance-Based Methods

Unlike the algorithms based on inductive learning, instance-based methods do not build a explicit model trough a training process. They rather work as lazy classification procedures, taking the k nearest neighbors (kNN) when a new data sample arrives. There are several instance-based MLC-adapted proposals, and a few of them are summarized in this section.

5.5.1 *Multilabel kNN, ML-kNN*

Presented in [38], ML-kNN is the best-known instance-based MLC algorithm. It internally works as a BR classifier, since a separate set of a priori and conditional probabilities are independently computed for each label. Therefore, any potential

correlation information among labels is disregarded by ML-kNN. The inner working details of ML-kNN were provided in Sect. 3.4.1.

Owing its simplicity and low computational complexity, ML-kNN is usually included in most experimental studies. It also has been used as foundation for other more elaborated MLC algorithms, such as IBLR-ML.

5.5.2 *Instance-Based and Logistic Regression, IBLR-ML*

Two similar MLC methods are introduced in [8], both of them based on the aforementioned ML-kNN algorithm. The core of the two proposals, named IBLR-ML and IBLR-ML+, use Bayesian techniques to consider the labels associated with nearest neighbors of the new instance as additional characteristics. Using this information the a priori probabilities are computed and a regression equation is obtained.

The main difficulty in these methods comes from the need to adjust an α parameter, in charge of setting the weight to be assigned to the additional attributes while computing the a posteriori probabilities. To accomplish this task an statistical parameter estimation method is used. This is an adaptation process which demands the enumeration of all instances in the MLD.

Although IBLR-ML can achieve better predictive performance than ML-kNN, it also has higher computational demands, including more memory consumption and training time.

5.5.3 *Other Instance-Based Classifiers*

The kNNc method was proposed in [6]. It works in two stages, combining instance selection techniques with instance-based classification. Firstly, a reduced set of instances is obtained by prototype selection techniques. The aim is to determine the set of labels which are nearest to the ones in the instance to be classified. Then, the full set of samples is used, but limiting the prediction to the labels inferred in the previous step.

BRkNN and LPkNN [27] are MLC classifiers made up combining BR and LP transformation methods with kNN classification techniques, respectively. They first apply the data transformation to the MLD, then use kNN to find the nearest neighbors, and generate the labelset from them.

The proposal in [16] is called BRkNN-new. As its name indicates, it is an improvement based on the BRkNN method just described above. The goal of the authors is to take advantage of label correlation information in order to improve the performance of the original model.

5.6 Probabilistic Methods

Probabilistic techniques, such as Bayesian models, mixture models, or conditional random fields, are commonly used in many MLC methods. Sometimes, these techniques appear only as a part of other algorithms, for instance the Bayesian model used in IBLR-ML, while in other cases they are the cornerstone of the method. This section enumerates some of the proposals in the latter group.

5.6.1 Collectible Multilabel Classifiers, CML and CMLF

The authors of these two algorithms, presented in [17], state that binary MLC methods assume the labels are fully independent, and do not take into account potential correlations among them. This is the reason to propose a model to capture this information, aiming to improve the predictive performance of the classifiers.

In order to get the correlations between labels, a CRF (*Conditional Random Field*) is used. A CRF is a probabilistic discriminative model, commonly used in tasks such as text segmentation. It associates a probability to each label depending on the existent observations, and with these data, the potential dependencies among outputs are modeled. Four different states can exist for each label pair with respect to the considered instance, none of the labels is relevant, both of them are applicable, only the first one is relevant, or only the second one is relevant.

The first proposed model is CML (*Collectible Multilabel*). It holds a correlation parameter for each label pair. The second one is CMLF (*CML with Features*), and it works with three-variable groups such as *attribute-label1-label2*. For each label pair associated with one attribute, CMLF holds a correlation value. Those labels which do not reach a certain frequency are discarded.

5.6.2 Probabilistic Generic Models, PMM1 and PMM2

These two methods were introduced in [31]. They are probabilistic generative models (PMM), whose goal is to automate text document classification by estimating the label probabilities from the terms appearing into the documents.

PMMs assume that each sample in the text MLD is a mixture of characteristic words related to the labels relevant to the instance. The main difference between PMM1 and PMM2 relies on the way the parameters controlling the algorithms are approximated, PMM2 being a more flexible version of PMM1.

5.6.3 Probabilistic Classifier Chains, PCC

PCC (*probabilistic classifier chains*) [9] is an extension of the CC method previously described in Chap. 4. The goal is to use Bayesian methods to optimize the chaining order of the binary classifiers, thus improving the overall classifier performance.

PCC models the dependency among labels computing the joint distribution for all of them, then deciding which is the optimum chaining order. Although this approach achieves better predictive performance than CC, its computational complexity is also much higher.

5.6.4 Bayesian and Tree Naïve Bayes Classifier Chains, BCC and TNBCC

The major drawback of PCC and other similar approaches based on classifier chains is their computational complexity. This is an inconvenience while working with MLDs having large sets of labels, and sometimes, it could be infeasible to use this kind of algorithms.

In [28], the authors propose several extensions to CC which follow a different path, named BCC (*Bayesian classifier chains*). TNBCC (*Tree Naïve BCC*) is an algorithm based on this approach. By using Bayesian networks to model, the dependencies among labels, it reduces the number of chain combinations to consider to finally compose the multilabel classifier.

5.6.5 Conditional Restricted Boltzmann Machines, CRBM

RBMs (*Restricted Boltzmann Machines*) [26] are a proven mechanism when it comes to produce high-level features from the low-level input attributes existent in a dataset. They are a usual component in the process to build deep belief networks, as well as other deep learning structures. An RBM is a probabilistic graphic model, specifically a two-layer graph, one input layer and one hidden layer. The latter is used to model the relationships between the input attributes.

Introduced in [21], the CRBM (*Conditional RBM*) algorithm relies on an RBM to retrieve dependencies among labels. The goal is to build a model able to operate with MLDs containing incomplete sets of labels. For doing so, the labels of each instance are given as input to the RBM, obtaining as output a dependency model capable of predicting missing labels from others presence.

5.7 Other MLC Adaptation-Based Methods

In addition to the more than twenty MLC methods mentioned above, adaptations of trees, ANNs, SVMs, instance-based, and probabilistic classifiers, in the literature can be found a lot more following alternative ways of attacking the problem. The enumerated below are some of them:

- **HG**: It is based on an hypergraph algorithm, each label being one edge, whose goal is to generate a model containing the relationships among labels. The resulting problem is hard to solve from a computational point of view, but the authors in [29] assume certain premises that allow them to accomplish the task as a simplified problem of minimum squares.
- **CLAC**: One of the major obstacles in facing MLC tasks is the usually huge number of label combinations an MLD can hold. This produces a secondary problem, even harder to solve, since the number of data samples sharing the same combination of labels can be almost negligible. These cases are identified in [32] as *disjuncts*, highlighting that if they are ignored, due to their poor representation, the obtained model can loss precision since in large MLDs many disjuncts can exist. The MLC method proposed is a lazy algorithm named CLAC (*Correlated Lazy Associative Classifier*), which increases the relevance of the potential disjuncts by discarding attributes and samples not related to the processed data instance.
- **GACC**: Introduced in [18], the GACC (*Genetic Algorithm for ordering Classifier Chains*) method is proposed as an alternative to the traditional ECC. While the latter randomly generates chains of binary classifiers, the former resorts to the use of a genetic algorithm to optimize the order of the classifiers in the chain. Besides the usual goal of improving classification results, the authors also aim to make the obtained model more easily interpretable.
- **MuLAM**: Although ant colonies based algorithms [12] are mostly used in optimization problems, as ACO (*Ant Colony Optimization*) methods, there are also techniques such as Ant-Miner [23] able to produce classification rules following the same approach. Taking Ant-Miner as foundation, in [7] the authors present a new MLC algorithm named MuLAM (*Multi-Label Ant-Miner*). It takes the multilabel classification problem as an optimization task, but the reported predictive performance seems to be far from the state-of-the-art MLC classifiers.
- **ML-KMPSO**: The proposal made in [22] is an MLC method which combines the kNN and MPSO (*Michigan Particle Swarm Optimization*) algorithms. Firstly, the a priority probabilities of each label are computed. Then, MPSO is used to optimize the selection of nearest neighbors to the considered instance, obtaining a set of expert *particles* which will help to choose the set of labels to be predicted.
- **GEP-MLC**: In [4], the authors present an MLC method founded on discriminant functions optimized by GEP (*Gene Expression Programming* [14]) techniques. GEP is a genetic programming approach specially suitable for regression problems, being used for facing traditional classification as well. GEP-MLC learns one or more discriminant function for each label, thus working as a BR transformation method in which the underlining classifiers are the functions optimized by GEP.

- **MLC-ACL**: Designed as a combination of transformation and adaptation methods, the MLC-ACL algorithm [4] has tree phases. Firstly, the MLD is transformed into a dataset with only one label, following the least frequent criterion. Secondly, a rule-based classification algorithm is applied. Lastly, an iterative process relies on the association rule mining A priori algorithm [1] to detect correlations among labels, transforming the rules obtained in the first step in a multilabel classifier.

5.8 Adapted Methods in Practice

In the previous sections, around thirty adaptation-based multilabel classification algorithms have been introduced. More than half of them can be included in one of the aforementioned four traditional classification approaches. The aim of this section is to test one algorithm of each one of these approaches, comparing the results of a multilabel tree, a multilabel neural network, a multilabel SVM, and a instance-based multilabel method. The selected algorithms are the following:

- **ML-Tree**: It is a recent multilabel classifier that takes into account the potential dependencies among labels, including this information into the tree model. As explained above (see Sect. 5.2.3), it follows an hierarchical approach to generate the tree.
- **BP-MLL**: This method is the first adaptation of an ANN to the multilabel problem, including a specific error function which considers the existence of multiple outputs.
- **Rank-SVM**: The SVM-based multilabel adaptation, as was explained in Sect. 5.4.2, also takes advantage of label correlation information aiming to improve classification performance.
- **BRkNN**: It is a lazy, instance-based multilabel learner which combines the BR transformation with kNN techniques. It is an alternative to the ML-kNN algorithm already used in experiments of previous chapters.

The BP-MLL and BRkNN algorithms are available in the MULAN package [30], while ML-Tree and Rank-SVM can be obtained from the authors' Web site.[1] The former is written in Java and it relies on the WEKA software package, as the latter is implemented in MATLAB.

[1] ML-TREE code can be downloaded from Dr. Qingyao Wu's Web page at https://sites.google.com/site/qysite. Rank-SVM code can be downloaded from http://cse.seu.edu.cn/people/zhangml/files/RankSVM.rar.

Table 5.1 Basic traits of MLDs used in the experimentation

Dataset	n	f	k	Card	Dens	TCS
`emotions`	593	72	6	1.868	0.485	9.364
`medical`	978	1 449	45	1.245	0.028	15.629
`scene`	2 407	294	6	1.074	0.179	10.183
`slashdot`	3 782	1 079	22	1.181	0.054	15.125
`yeast`	2 417	103	14	4.237	0.303	12.562

5.8.1 Experimental Configuration

The four multilabel classifiers pointed out above have been run over the partitions of five MLDs. These are `emotions`, `medical`, `scene`, `slashdot` and `yeast`. Their basic traits are that shown in Table 5.1. The tables in Chap. 3 provide the main characteristics of each one of them. Looking at the *TCS* values in Table 5.1, these MLDs complexity goes from the 9.364 score of `emotions` to the 15.629 of `medical`.

As in the previous chapter, each MLD was partitioned using a 2×5 fold cross-validation scheme. The prediction performance is assessed with four evaluation metrics. Since some of the classifiers included in this experimentation produce a ranking as output, in addition to the usual example-based *HLoss* (Hamming loss), three ranking-based evaluation metrics have been selected, *RLoss* (Ranking loss), *OneError*, and *AvgPrecision* (Average precision). The goal is to check how the performance of the classifiers is reported by different evaluation metrics. All of them were detailed in Chap. 3. The reported values in the following section are average values computed from the 10 runs for each MLD/classifier.

5.8.2 Classification Results

As we already know, *HLoss* is a metric to minimize since it is an averaged counter of misclassified labels. In Fig. 5.1, the results for this metric have been represented as $1 - HLoss$, aiming to preserve the perception that a larger area corresponds to a better performance of the classifier. From this plot it is easy to infer that BP-MLL is the worst performer, while the other three seem to show a similar behavior.

The raw *HLoss* values for each MLD/classifier combination are the ones shown in Table 5.2. Best values are highlighted in bold. As can be seen, Rank-SVM obtained the best results for the two most complex MLDs, while BRkNN performed better with the simpler ones.

RLoss is also a loss measure, as *HLoss*, but it is ranking based instead of example based. Therefore, the goal of a classifier must be to minimize this metric. The same previous approach has been used to represent *RLoss* values in the plot shown in

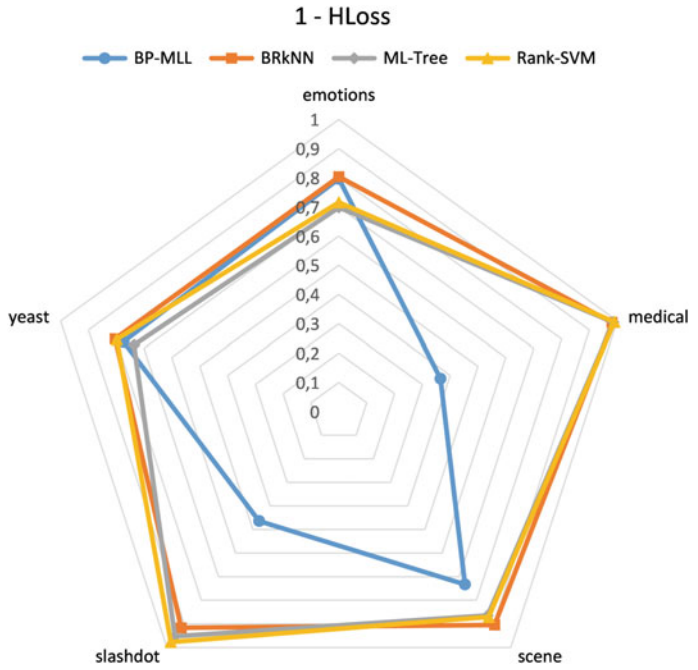

Fig. 5.1 Classification results assessed with Hamming Loss

Table 5.2 Results assessed with *HLoss* (lower is better)

Dataset	BP-MLL	BRkNN	ML-Tree	Rank-SVM
`emotions`	0.2034	**0.1965**	0.3016	0.2842
`medical`	0.6347	0.0182	0.0142	**0.0104**
`scene`	0.2669	**0.0948**	0.1352	0.1288
`slashdot`	0.5360	0.0834	0.0475	**0.0230**
`yeast`	0.2265	**0.1967**	0.2641	0.2024

Fig. 5.2. As can be observed, the shape of the classifiers is quite similar to Fig. 5.1, although some larger differences can be detected among the three best performing methods.

Once again that BP-MLL produces the worst classification results, specifically when used with the more complex MLDs, can be stated. In general, the line which corresponds to the Rank-SVM algorithm seems to be the closest to the outer limit, denoting that it is the best performed when the results are assessed with this evaluation metric. This perception can be confirmed by examining the values provided in Table 5.3. With the only exception of `emotions`, the best values always come from the Rank-SVM method.

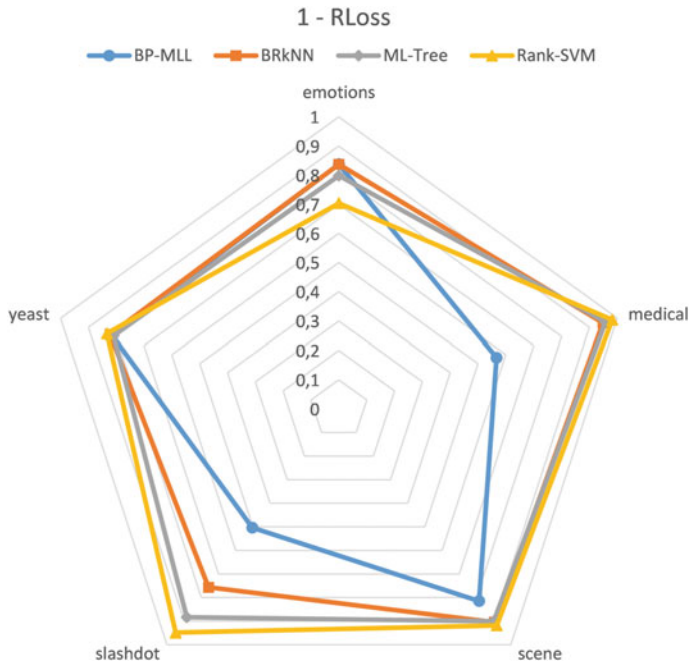

Fig. 5.2 Classification results assessed with Ranking Loss

Table 5.3 Results assessed with *RLoss* (lower is better)

Dataset	BP-MLL	BRkNN	ML-Tree	Rank-SVM
`emotions`	**0.1625**	0.1629	0.2020	0.2961
`medical`	0.4340	0.0512	0.0422	**0.0180**
`scene`	0.1852	0.0955	0.0998	**0.0818**
`slashdot`	0.4968	0.2430	0.1163	**0.0512**
`yeast`	0.1854	0.1786	0.1909	**0.1672**

The second ranking-based evaluation metric is *OneError*. It counts how many times the label predicted with more confidence, ranked highest in the output returned by the classifier, is not truly relevant to the instance. As the two previous ones, *OneError* is also a metric to be minimized.

As can be seen in Fig. 5.3, which shows the spider plot for $1 - OneError$, the differences among the classifiers are much clearer with this metric. It can be verified that the three simplest MLDs, `emotions`, `scene` and `yeast`, put fewer obstacles than `medical` and `slashdot` for most of the classifiers. The performance of BP-MLL is particularly poor with these two last MLDs. BRkNN also shows a degradation of its behavior while working with them.

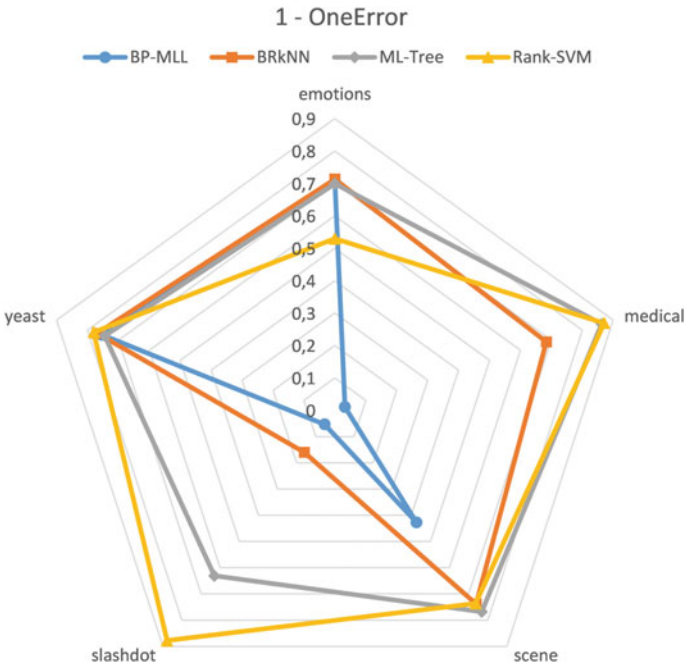

Fig. 5.3 Classification results assessed with One Error

Table 5.4 Results assessed with *OneError* (lower is better)

Dataset	BP-MLL	BRkNN	ML-Tree	Rank-SVM
`emotions`	**0.2942**	0.2866	0.3010	0.4704
`medical`	0.9673	0.3160	0.1375	**0.1293**
`scene`	0.5727	0.2614	**0.2318**	0.2634
`slashdot`	0.9466	0.8399	0.3691	**0.1221**
`yeast`	0.2470	0.2317	0.2561	**0.2203**

The raw *OneError* measures are the shown in Table 5.4, with best values highlighted in bold. Rank-SVM achieves best result with the three more complex MLDs. Paradoxically, this algorithm shows its worst performance with the simplest dataset, `emotions`.

The last metric used to assess the performance in this experimentation is *AvgPrecision*. It measures the number of positions in the ranking that have to be visited before a non-relevant label is found. Therefore, the larger is the value obtained, the better is working the classifier.

In this case, the values produced by the evaluation metric can be plotted without any transformation, as has been done in Fig. 5.4. This plot closely resembles the one in Fig. 5.3; thus, the conclusions drawn from the *OneError* metric can be mostly applied to *AvgPrecision* also.

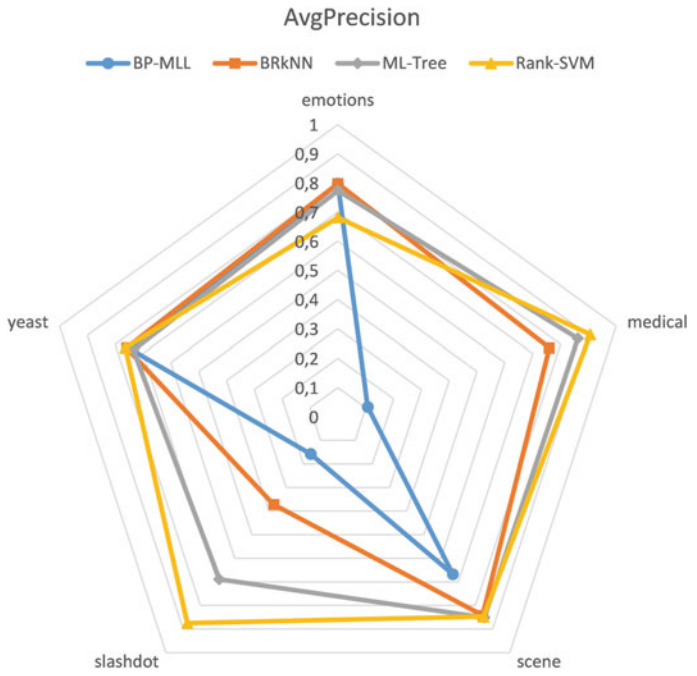

Fig. 5.4 Classification results assessed with *AvgPrecision* (higher is better)

Table 5.5 Results assessed with Average Precision (higher is better)

Dataset	BP-MLL	BRkNN	ML-Tree	Rank-SVM
`emotions`	0.7971	**0.7979**	0.7737	0.6813
`medical`	0.1076	0.7594	0.8626	**0.9064**
`scene`	0.6680	0.8423	**0.8517**	0.8476
`slashdot`	0.1583	0.3723	0.6892	**0.8746**
`yeast`	0.7446	0.7583	0.7342	**0.7636**

The set of best values highlighted in Table 5.5 is also very similar to that of Table 5.4. Rank-SVM produces the best results for the most complex MLDs again, while it works quite poor with the simplest one.

The overall conclusion that can be gathered from this experiments is that, from the four tested algorithms, Rank-SVM tends to perform better in most cases. On the contrary, BP-MLL usually produces the worst result with few exceptions. The other two multilabel classifiers, BRkNN and ML-Tree, are in between.

5.9 Summarizing Comments

Traditional pattern classification is a largely studied problem in DM, with dozens of proven algorithms ready to use over binary and multiclass data. Therefore, the adaptation of these methods to work with multilabel data was a clear path from the beginning. In this chapter, several of these MLC adaptation-based algorithms have been introduced.

The diagram in Fig. 5.5 summarizes the algorithm adaptation methods described in the previous sections, grouping them according to the type of base algorithm they are derived from. The first four branches, counting from left to right, correspond to the main four traditional classification approaches, trees, neural networks, SVMs, and instance-based learners.

Four of the enumerated methods, BP-MLL, BRkNN, ML-Tree, and Rank-SVM, have been used in a limited experimentation processing five MLDs. According to the assessment made by four distinct evaluation metrics, the latter shows the best predictive performance, while the former is the worst one.

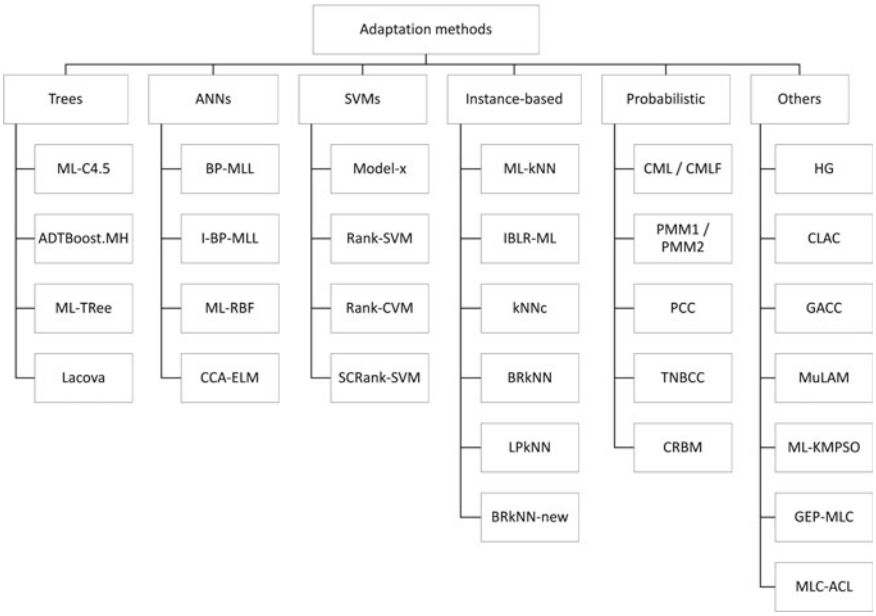

Fig. 5.5 Overview of multilabel algorithm adaptation-based methods

References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB'94, pp. 487–499. Morgan Kaufmann (1994)
2. Agrawal, J., Agrawal, S., Kaur, S., Sharma, S.: An Investigation of fuzzy PSO and fuzzy SVD based RBF neural network for multi-label classification. In: Proceedings of the 3rd International Conference on Soft Computing for Problem Solving, SocProS'13, vol. 1, pp. 677–687. Springer (2014)
3. Al-Otaibi, R., Kull, M., Flach, P.: Lacova: a tree-based multi-label classifier using label covariance as splitting criterion. In: Proceedings of the 13th International Conference on Machine Learning and Applications, ICMLA'14, pp. 74–79. IEEE (2014)
4. Ávila, J., Gibaja, E., Ventura, S.: Multi-label classification with gene expression programming. In: Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems, HAIS'09, pp. 629–637. Springer (2009)
5. Boutell, M., Luo, J., Shen, X., Brown, C.: Learning multi-label scene classification. Pattern Recogn. **37**(9), 1757–1771 (2004)
6. Calvo-Zaragoza, J., Valero-Mas, J.J., Rico-Juan, J.R.: Improving knn multi-label classification in prototype selection scenarios using class proposals. Pattern Recogn. **48**(5), 1608–1622 (2015)
7. Chan, A., Freitas, A.A.: A new ant colony algorithm for multi-label classification with applications in bioinfomatics. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO'06, pp. 27–34. ACM Press (2006)
8. Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Mach. Learn. **76**(2–3), 211–225 (2009)
9. Cheng, W., Hüllermeier, E., Dembczynski, K.J.: Bayes optimal multilabel classification via probabilistic classifier chains. In: Proceedings of the 27th International Conference on Machine Learning, ICML'10, pp. 279–286 (2010)
10. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: Proceedings of the 5th European Conference Principles on Data Mining and Knowledge Discovery, PKDD'01, vol. 2168, pp. 42–53. Springer (2001)
11. De Comité, F., Gilleron, R., Tommasi, M.: Learning multi-label alternating decision trees from texts and data. In: Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM'03, vol. 2734, pp. 35–49. Springer (2003)
12. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press (2004)
13. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems, vol. 14, pp. 681–687. MIT Press (2001)
14. Ferreira, C.: Gene expression programming in problem solving. In: Soft Computing and Industry, pp. 635–653. Springer (2002)
15. Freund, Y., Mason, L.: The alternating decision tree learning algorithm. In: Proceedings of the 16th International Conference on Machine Learning, ICML'99, pp. 124–133 (1999)
16. Genga, X., Tanga, Y., Zhua, Y., Chengb, G.: An improved multi-label classification algorithm BRkNN. J. Inf. Comput. Sci. **11**(16), 5927–5936 (2014)
17. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM'05, pp. 195–200. ACM (2005)
18. Gonçalves, E.C., Plastino, A., Freitas, A.A.: A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In: Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'13, pp. 469–476. IEEE (2013)
19. Grodzicki, R., Mańdziuk, J., Wang, L.: Improved multilabel classification with neural networks. In: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature, PPSN X, pp. 409–416. Springer (2008)

20. Kongsorot, Y., Horata, P.: Multi-label classification with extreme learning machine. In: Proceedings of the 6th International Conference on Knowledge and Smart Technology, KST'14, pp. 81–86. IEEE (2014)
21. Li, X., Zhao, F., Guo, Y.: Conditional restricted boltzmann machines for multi-label learning with incomplete labels. In: Proceedings of the 18th International Conference on Artificial Intelligence and Statistics, AISTATS'15, pp. 635–643 (2015)
22. Liang, Q., Wang, Z., Fan, Y., Liu, C., Yan, X., Hu, C., Yao, H.: Multi-label classification based on particle swarm algorithm. In: Proceedings of the 9th IEEE International Conference on Mobile Ad-hoc and Sensor Networks, MSN'13, pp. 421–424. IEEE (2013)
23. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. **6**(4), 321–332 (2002)
24. Quinlan, J.R.: C4.5: Programs for machine learning (1993)
25. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37**(3), 297–336 (1999)
26. Smolensky, P.: Information processing in dynamical systems: foundations of harmony theory (1986)
27. Spyromitros, E., Tsoumakas, G., Vlahavas, I.: An empirical study of lazy multilabel classification algorithms. In: Artificial Intelligence: Theories, Models and Applications, pp. 401–406. Springer (2008)
28. Sucar, L.E., Bielza, C., Morales, E.F., Hernandez-Leal, P., Zaragoza, J.H., Larrañaga, P.: Multi-label classification with bayesian network-based chain classifiers. Pattern Recogn. Lett. **41**, 14–22 (2014)
29. Sun, L., Ji, S., Ye, J.: Hypergraph spectral learning for multi-label classification. In: Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining, ACM SIGKDD'08, pp. 668–676. ACM (2008)
30. Tsoumakas, G., Xioufis, E.S., Vilcek, J., Vlahavas, I.: MULAN: a java library for multi-label learning. J. Mach. Learn. Res. **12**, 2411–2414 (2011)
31. Ueda, N., Saito, K.: Parametric mixture models for multi-labeled text. In: Proceedings of the 15th Annual Conference on Neural Information Processing Systems, NIPS'02, pp. 721–728 (2002)
32. Veloso, A., Meira Jr, W., Gonçalves, M., Zaki, M.: Multi-label lazy associative classification. In: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD'07, pp. 605–612. Springer (2007)
33. Wang, J., Feng, J., Sun, X., Chen, S., Chen, B.: Simplified constraints Rank-SVM for multi-label classification. In: Proceedings of the 6th Chinese Conference on Pattern Recognition, CCPR'14, pp. 229–236. Springer (2014)
34. Wu, Q., Ye, Y., Zhang, H., Chow, T.W., Ho, S.S.: Ml-tree: a tree-structure-based approach to multilabel learning. IEEE Trans. Neural Networks Learn. Syst. **26**(3), 430–443 (2015)
35. Xu, J.: Fast multi-label core vector machine. Pattern Recogn. **46**(3), 885–898 (2013)
36. Zhang, M.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans. Knowl. Data Eng. **18**(10), 1338–1351 (2006)
37. Zhang, M.: Ml-rbf : RBF neural networks for multi-label learning. Neural Process. Lett. **29**, 61–74 (2009)
38. Zhang, M., Zhou, Z.: ML-KNN: a lazy learning approach to multi-label learning. Pattern Recogn. **40**(7), 2038–2048 (2007)