Multi-label Classification With Weighted Labels Using Learning Classifier Systems

Shabnam Nazmi[a], Mohammad Razeghi-Jahromi[b], Abdollah Homaifar[c]*
School of Electrical and Computer Engineering
North Carolina A&T State University
[a] snazmi@aggies.ncat.edu, [b] mrazeghijahromi@ncat.edu, [c] homaifar@ncat.edu

***Abstract*—In this work the Michigan style strength-based learning classifier system, which is a rule-based supervised learning algorithm, is extended by changing its action space representation to handle multi-label classification tasks. Moreover, it is assumed that the class membership for training data is partially known and the uncertainty is represented by confidence values that reflects the probability of each label being true. Here, it is assumed that confidence values are provided by the expert or generated from a probability distribution function. Necessary parameters are introduced, and learning classifiers are modified to learn simultaneously the confidence level and multi-label in the training data. Therefore, the classifier's output for unseen data is a set of labels and their respective confidence values. Additionally, to quantify the classifier performance, a novel loss measure is introduced that generalizes the well-known Hamming loss criteria to takes into account the classification error and confidence estimation error simultaneously. The algorithm is tested on one real-world data and two synthetic data sets. Results show the ability of the model in learning multi-class and multi-label data with low confidence estimation error.**

Keywords: Learning classifier systems, Multi-label classification, Uncertain labels, Evolutionary based learning

I. Introduction

In supervised classification tasks, usually it is assumed that available data is confidently labeled by a human. Nonetheless, in many applications human supervision is imprecise, difficult or expensive, therefore it is useful that if available, to incorporate some level of confidence on the labels assigned to the samples while training the model. Some of the supervised learning approaches have been adapted to handle uncertainty in labels. In many studies the Dempster-Shafer (DS) belief theory is employed to process the uncertainty in labels, which can include probabilistic, possibilistic or imprecise labels. In [1] and [2] belief decision trees [3] are used to handle multiple label data. In [4] and [5], each sample is generated from a mixture model assigned with a "soft" label defined as a DS basic belief and maximum likelihood estimation is extended to solve the problem.

The sparse literature that deals specifically with partial knowledge provided for sample-label association, has motivated the authors to try and find a model that takes the maximum benefit from the information contained in the data set. In this work, the objective is to represent the partial information contained in uncertainly labeled part of data with confidence levels that are possibly provided by the expert or a known probability distribution. Thus, the first contribution of this work is a learning algorithm that learns the labels with their respective confidence values. This goal is achieved by employing a learning classifier system (LCS) [6], [7] that is a rule-based evolutionary classifier system. In learning classifiers, the complete model is formed of a population of rules in the form of *IF condition-THEN action* that solve the classification task cooperatively. Each rule (also called classifier) has some parameters that evolve during the learning process to model the problem. The genetic algorithm (GA) helps the algorithm to enumerate the population with the fittest and most useful rules. LCSs use a scheme of reinforcement learning [8] that guides the population towards the solution by rewarding and punishing its actions.

When dealing with multi-label data, association of a sample to multiple classes simultaneously becomes more uncertain and employing a confidence-based labeling helps to train a more reliable multi-label classifier. Multi-label classification (MLC) problems have a broad range of approaches that solve the MLC task from different perspectives. There are a number of methods that transform the original multi-label data set into single-label data sets to facilitate the use of existing classification techniques. Binary relevance (BR) [9], [10] is a popular problem transformation method that trains k separate binary classifiers, one for each class. Label powerset (LP) [10] deals with the unique combinations of labels that exist in data as a new label. To improve its efficiency, [11] has considered a random combination of labels with different sizes as label sets to train classifier. The second family of MLC approaches are algorithm adaptation methods, that modify the existing classification algorithms to adapt to the MLC task. For instance, in [12] a lazy multi-label classification algorithm using *k-NN* method is proposed. In [13], the MLC task is formalized within the Bayes probability setting that explicitly takes into account the label dependence. In [14], using boosting of decision tree models, each weak hypothesis produces not only a class label, but also its confidence in the prediction. However, this approach does not consider the possible, a-priori information about label confidences in training data.

In [15], classifier action is adapted to learn BR vectors and evolve default hierarchies within multi-label data using organization classifier systems, though results are reported only

*Corresponding author.

for a small binary data set with three bits and three classes. In this work, the same representation is used to extend the LCS model to deal with multi-label data. Thus, in the multi-label learning classifier systems (MLCS), algorithm training instances can belong to more than one class with respective confidence levels. In contrast to most problem transformation methods, a single model is trained and is able to predict a set of labels for unseen samples, as well as its confidence in each of those labels. Note that, for a certain sample the sum of the values in the confidence vector does not have to sum up to one. This is because the sample can belong to more than one class with complete confidence (i.e. one) for each label.

Finally, a novel accuracy measure is proposed that reflects the classification error of the model and its confidence estimation error simultaneously.

The rest of this paper is organized as follows. In the next section, a summary of the notations and abbreviations that are used in the paper is provided. Next, learning classifier systems are introduced and the proposed structure and required modifications to handle multi-label data and confidence level in labels is explained in detail. Next, training results for three data sets are provided and are compared to other techniques. Finally conclusions and future work are provided.

II. Notations

The notations that are used in this paper are as follows. It is assumed that $X \subset \mathcal{R}^m$ represents the input space and $Y = \{y_1, y_2, \ldots, y_k\}$ is the finite set of class labels. Each multi-label instance $x \in X$ is associated with a subset of classes as $y \subset Y$ (for single label data y is a single label), that may have its confidence levels ($C \in \mathcal{R}^k$) assigned to each label. If no confidence is assigned to samples, it is assumed to be one by default and for each label-set assigned to instances, a binary relevance vector $\lambda \in \mathcal{R}^k$ is created. Therefore, $D = \{(x_1, \lambda_1, C_1), (x_2, \lambda_2, C_2), \ldots, (x_n, \lambda_n, C_n)\}$ is the set of training instances such that for every λ_i and $j = 1, \ldots, k$ we have,

$$\lambda_i(j) = \begin{cases} 1 & y_j \in y \\ 0 & y_j \notin y \end{cases} \tag{1}$$

If a sample does not belong to a specific class, its respective $\lambda(j)$ value is zero and thus no confidence is defined for it. If it belongs to a specific class, its $\lambda(j)$ value will be one and its confidence value can take a value between zero and one. $|A|$ represents the cardinality of set A, $d_H(a, b)$ is the hamming distance between two sets, and $d_{EH}(a, b)$ stands for the extended hamming distance between two sets that is introduced in this work. Therefore, the objective is to find a hypothesis $H : X \rightarrow (Y \times C)$, that for each instance generates a set of predicted label binary relevance vector ($\hat{\lambda}$), along with their respective confidence values ($\hat{C}$).

III. Learning classifier system with weighted labels for multi-label data

Learning classifiers have many variations that aim to solve different problem domains. In *XCS* [16], classifier fitness is based on its accuracy in labeling instances. *UCS* [17], [18] is derived from *XCS* that works in a supervised learning framework, and *ExSTraCS* [19], [20] that has shown to solve heterogeneous supervised learning problems with noisy data. In this work strength-based classifier systems [6] are chosen that are primarily developed for binary-valued problems and have shown to successfully solve the Boolean Multiplexer benchmark problem [16] up to 20-multiplexer (multiplexer with four address bits). Nonetheless, strength-based classifiers have not been proven to be efficient is solving complex problems with real-valued attributes or problems with large number of data. However, their simpler architecture compared to *XCS* or *ExSTraCS*, lets us interpret the impact of proposed extension on the process of learning. In the proposed method, classifier system is extended to solve problems with real-valued attributes to show the actual potential of the proposed weighted-label multi-label classifier structure. This idea can be extended to *XCS* or *UCS* classifier systems to benefit their powerful inference capability, specially for problems with mixed attributes and larger number of samples [19], [21].

In this part LCS is formulated and discussed to solve the problem of MLC, but it is easily applicable to multi-class problems. In LCS each rule has a condition that represents the input space, an action that represents the labels, a strength value (S) that shows its usefulness, and finally an experience that is a measure of the rule's contribution in training. In addition to the parameters original to strength-based classifiers, a confidence estimate value and its respective error (ε) that will effect the amount of rule's participation in learning and its reward from the environment is introduced to each rule.

For binary features, the condition part is composed of ternary alphabet with '#' representing a wild card that matches with either 1 or 0. For real-valued features, the center-spread representation [22] is employed, which represents the i^{th} attribute in the classifier condition by a pair (c_i, s_i). In other word, each attribute covers the interval $(c_i - s_i, c_i + s_i)$ of its respective feature space. In contrast to widely used binary coding, the action of classifier is the binary relevance of the class labels and can represent association to more than one class at a time [15].

LCS starts with an empty population and gradually learns the problem by generating new rules to cover a new instance. For every new rule, its strength initiates with a small value (S_0) and for correctly classified instances, its strength will increase through the external reward. Confidence estimates will initialize with zero and by means of a simple update rule, gradually approaches the correct value provided for the instances. In addition, error and experience are initialized at zero.

The process of learning begins by providing one instance at a time (x_l, λ_l, C_l). Classifiers in the population are scanned to find rules that have matching conditions with the sample. A classifier with binary attributes will match an input if for every attribute it has either identical values with the given instance or the *wild card* (#), which matches every value of the attribute. For real-valued attributes the condition $c_i - s_i < x_i < c_i + s_i$

must satisfy for every attribute. If there is no match for the current sample, the covering process [16] creates a new rule with a random condition that matches the sample and has its correct label as its action. For this purpose, with probability equal to $1 - P_{\#}$ the attribute in the condition is specified, i.e. for binary-valued attributes the value of the instance (x_l) at the i^{th} position is considered as the i^{th} bit of condition and for real-valued attribute as the center and a random spread is generated with respect to the range of each attribute. With probability equal to $P_{\#}$, the attribute will be a wild card. The new rule is placed into $[P]$ until the population size reaches its limit.

Rules with matching conditions constitute the *match list* ($[M]$) and bid a portion of their strength to compete for labeling the sample and possibly win the external reward. A classifiers with higher strength and lower error should have more potential to outbid others, therefore the bid (B) that was originally a function of strength and specificity (μ), has been modified to reflect classifier error as well. Therefore for *rule*$_i$ the bid value is

$$B_i = C_{bid} S_i \mu_i e^{-\alpha \varepsilon_i}. \tag{2}$$

In this equation, $0 < \alpha < 1$ is a constant that defines the accepted threshold for classifier error to be effective and C_i is the bid coefficient. The motivation for integrating the classifier error in its bid is that, it will create a pressure toward a population with smaller confidence estimation error. Without this parameter during the bidding process, the model might converge to a population with rules that are equally incorrect in confidence estimation and their survival is only affected by their strength. Therefore, there is a probability that the model may trap in local solutions.

After the winner is selected, an *action list* ($[A]$) is formed from those classifiers in $[M]$ that advocate exactly the same action as the winning action. The winning action is then executed, a possible reward is received and the parameters of the classifiers in $[A]$ are updated. Following (3), classifier errors is updated to reflect its confidence estimate distance with the actual confidence of the sample as

$$\varepsilon_i = \|\hat{C}_i - C_l\|_1. \tag{3}$$

The parameter that controls the minimum effective value of ε for each rule is α; the larger this coefficient is, the less chance a rule has to win the bid and has a smaller share of external reward. Since GA uses strength as the fitness measure of rules, selecting a larger value for α, decreases the selection probability of rules that have a relatively larger error but have matching conditions and correct actions, as result delays discovery and reproduction of correct classifiers. Accordingly, a trade off between small confidence estimation error and faster convergence is desirable. One can start training the model in a more relaxed condition with smaller α value to let the algorithm discover useful rules, and continue the training by increasing the α value to decrease the confidence estimation error even more.

Next, classifier confidence value is updated using the delta rule

$$\hat{C}_i := \hat{C}_i + \beta |C_l - \hat{C}_i|. \tag{4}$$

Classifier confidence estimate is updated using the *mayonne adaptive modifée* [23], according to which parameters are set to the average of the so far encountered cases as long as the average update causes a stronger change than the update due to the learning rate β. If the winning action is correct, an external reward will be distributed among classifiers in $[A]$ proportional to their strength and error values as

$$R_i = \frac{S_i e^{-\alpha \varepsilon_i}}{\sum_{i \in [A]} S_i e^{-\alpha \varepsilon_i}} R_0. \tag{5}$$

R_0 is the total reward provided by the environment. Here the resource sharing scheme is an extension of FPRS [24] to reflect classifier confidence estimate accuracy as well. Thus, classifier strength is a measure of its correct prediction of class label, in addition to its correct estimation of confidence value.

Finally, classifier strength is updated following (6) by adding a reward and deducting taxes. Also, from the rest of the population that are a part of $[M]$ or remained in $[P]$, some tax values are deducted that are proportional to their strengths

$$S_i := S_i(1 - C_{tax} - C_{bid}) + R_i. \tag{6}$$

Therefore, one iteration of learning finishes and LCS waits for the next training instance. This process continues for the number of defined iterations.

Moreover, at an initially defined frequency, genetic algorithm selects two individuals from $[A]$ (niche GA [7]) and creates two off-springs after cross-over and mutation. Created off-springs are replaced into $[P]$ for rules with the lowest strength. Note that in interaction with genetic algorithm, classifier strength acts as its fitness value, therefore GA tends to favor classifiers with high strength and low error. Procedure represented in (1) shows the flow of the learning algorithm during training.

In the test stage, for each test data, the population of classifiers is scanned to find those with a matching condition part. Assume that there are sub-populations offering r different label-sets, such that $\mathcal{M} = \{m_1, \ldots, m_r\}$ represents the number of classifiers advocating each of those label-sets. Class labels predicted by the model in the union of the labels existing in r unique label set. To calculated the collective confidence votes of the model for each of the labels, first the average confidence estimate of each subpopulation is calculated as, $\bar{C} = \{\bar{C}_1, \ldots, \bar{C}_r\}$. Then to combine the effect of all subpopulation, weighted average of the $\bar{C}_i$'s is calculated

$$\tilde{C} = \frac{\sum_{i=1}^{r} \bar{C}_i m_i}{\sum_{i=1}^{r} m_i}. \tag{7}$$

Thus $\tilde{C}$ is the confidence level vector of the model for each label. For a class that is suggested at least by one rule, the respective $\tilde{C}$ value is nonzero, and for the the rest of the classes, $\tilde{C}$ contains zeros. Procedure represented in (2) shows the flow of the learning algorithm during test.

Algorithm 1 MLCS-training

Initialize parameters;
while Maximum iteration not reached **do**
  for i = 1:Number of training instances **do**
    $Sample_i = (x_i, \lambda_i, C_i)$;
    Create $[M]$;
    if $[M] = \emptyset$ **then**
      Do covering;
    end if
    for j = 1:$|[M]|$ **do**
      Calculate bids (B_j) using (2);
    end for
    Select the maximum bid;
    Create $[A]$ and determine if the winning action is correct;
    if If winning action is correct **then**
      for j = 1:$|[A]|$ **do**
        Calculate reward share (R_j) using (5);
        Update error (ε_j) using (3);
        Update confidence estimate ($\hat{C}_j$) using (4);
        Update classifier strength using (6);
      end for
    end if
    Deduct taxes from $[P]$ and $[M]$;
    Apply GA according to the defined frequency;
    Update accuracy of the model;
  end for
end while

Algorithm 2 MLCS-test

for i = 1:Number of test instances **do**
  $Sample_i = (x_i, \lambda_i, C_i)$;
  Create $[M]$;
  Extract the union of all label sets as predicted labels;
  Extract number of rules advocating each label set ($\mathcal{M}$);
  Calculate $\bar{C}$ for each label set;
  Calculate weighted average of $\bar{C}$ using (7)
end for

IV. Performance measure

The performance of classification in MLC is mostly reported in terms of Hamming loss [13], which is defined as the number (or fraction) of labels whose relevance is incorrectly predicted. Using the notation introduced in section II, the Hamming loss $L_H(\cdot)$ for a hypothesis is defined as,

$$L_H(\lambda, H(x)) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{k}d_H(\lambda_i, H_i(x)) \tag{8}$$

In which k is the total number of classes in D.

In this work, to assess the accuracy of the model, a novel measure is proposed that takes into account both classification accuracy and confidence estimation accuracy, that can be assumed as an extension to the hamming loss. More specifically, hamming distance is extended to take into account confidence estimation error as well. Assume that label prediction and confidence estimation, constitute a two dimensional space. The locus of the label vector λ (and its prediction $\hat{\lambda}$) and confidence vector C (and its estimation $\hat{C}$) is represented by the solid line in figure (1) for one class. Considering class i of a k class data set, if the label value is shown as 't' and its prediction shown as 'o', the classification error for this class can be calculated as the Euclidean distance ($d_i(o,t)$) between 't' and 'o', shown by the dashed line. Finally, the collective error considering all classes will be the sum of each individual distance.

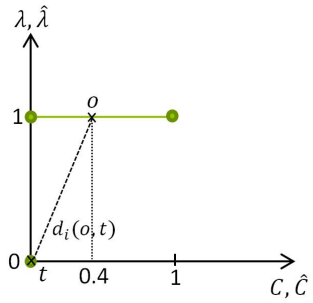

Fig. 1. Locus of label-confidence value pairs in two-dimensional space per class.

$$L_{EH}((\lambda \times C), H(x)) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{k}d_{EH}((\lambda_i, C_i), H_i(x)) \tag{9}$$

In which, d_{EH} is calculated as

$$d_{EH} = \sum_{i=1}^{k} d_i(o,t). \tag{10}$$

To have a better understanding, one can consider an instance from a three class data set such that $\lambda = [0,0,1]$ with $C = [0,0,1]$, while model has predicted $\hat{\lambda} = [1,0,1]$ with $\hat{C} = [0.4, 0, 0.95]$. Then for the first class, error is the collective outcome of misclassification and missed prediction value, as shown in figure (1) by the dotted line. For the third class, the error is only because of error in confidence estimation, that lies on the vertical line corresponding to $\lambda = 1$, and collective error is the average of these two distances.

V. Simulation results and discussion

To examine the performance of the proposed learning algorithm, three experiments are conducted using leave-one-out (LOO) cross-validation technique which is an almost unbiased estimator of the true error rate of a classifier [25]. Results are reported in terms of Hamming loss and extended Hamming loss for multi-label data sets, and the standard accuracy for a multi-class data set. To show how well classifier population learns the confidence levels, the average confidence estimation error ($\bar{\varepsilon}$) of an entire population is also reported for each experiment. Parameter specifications used in the proposed method are listed in Table (I).

TABLE I
LIST OF PARAMETERS USED TO TRAIN THE MLCS MODEL.

Population size limit	150
α	3
β	0.2
$S_0 = R_0$	100
$P_{\#}$	0.6
C_{bid}	0.1
$P_{crossover}$	0.8
$P_{mutation}$	0.04

1) Experiment 1: The first experiment is on a multi-class real-world data set. For this purpose, iris data set [26] is selected which has three features, four classes and 150 samples. For training the model, labels are coded into BR vectors and since the data set is multi-class, each vector has only one non-zero element. For the sake of simplicity, it is assumed that confidence values are all one, i.e. all data is confidently labeled, but it is also possible to assign lower confidences to labels. Results are compared to the decision tree classifier and also to the results reported in [27] for fuzzy rule mining using GA and multi-rule-table method [28] as shown in Table (II).

TABLE II
LOO CROSS-VALIDATION RESULTS FOR IRIS DATA SET.

	Rate %	$\bar{\varepsilon}$	Average number of rules
MLCS	97.57	0.0073	91.33
Fuzzy+GA	96	-	12.13
Multi-rule	94.67	-	691.11
DT	100	-	-

Results show that the proposed method has a slightly better performance in learning a multi-class problem compared to other reported rule-based methods, although the number of generated rules is larger than the fuzzy rule-based classifier with random search. Note that, in this experiment reducing the final number of rules was not our objective, though it is possible to employ appropriate rule condensation [16] techniques to reduce the number of rules in the trained model. Decision tree algorithm outperforms all other methods which is not unexpected due to its strong induction capability. Moreover, according to the value of $\bar{\varepsilon}$, MLCS has learned the confidence values of labels with a small average error over the population of rules. This means rules that are generated with zero confidence, have gradually learned the objective confidence levels of the data with small error.

2) Experiment 2: In the second experiment, a Boolean multi-label data is used that is an extension of the binary multiplexer benchmark problem. For this purpose, the Boolean 6-multiplexer is selected that is a binary-class problem taking a six-bit string as input. The two left-most bits can be considered as the address to a position in the remaining four bits. In disjunctive normal form, the function that specifies the class label is given by $F_6 = \acute{b}_0\acute{b}_1b_2 + \acute{b}_0b_1b_3 + b_0\acute{b}_1b_4 + b_0b_1b_5$, where subscript index bits from left to right and primes denote negation. To show the proposed algorithm's ability to learn multi-label data, the class definition of the multiplexer problem is modified. In the modified definition, the address bits of a string are concatenated with the label specified by F_6 and form the label for that string. The new labeling scheme, generally can contain more than one non-zero bits, therefor represents a multi-label sample. In the first test it is assumed that all labels are confidently assigned, i.e. $C = 1$ for each class. Then the experiment is repeated for the same data set when confidence value for class one in equal to 0.5 for all samples. The results are compared to support vector machine (SVM) and decision tree (DT) algorithms as reported in Table (III).

TABLE III
LOO CROSS-VALIDATION RESULTS FOR F_6 PROBLEM.

	HL	EHL	$\bar{\varepsilon}$	Label confidence
MLCS	0	0.0565	0.028	C = 1
MLCS	0	0.0165	0.012	C(Class 1) = 0.5
SVM	0.077	-	-	-
DT	0	-	-	-

According to the results, the proposed algorithm is performing as good as decision tree with zero HL, which means that the class labels are learned completely, while support vector classifier has some misclassification and does not perform as well as MLCS and decision tree. EHL is nonzero due to the small confidence estimation error of individual rules. When the data is labeled with partial confidence, i.e. confidence values are smaller than one, ε reaches to zero in few number of iterations. This can be inferred from the two simulation results with complete confidence and partial confidence for class one ($C_i(1) = 0.5$). Note that when samples in training data have different label confidence values, the solution search space becomes larger and the number of rules that are generated is larger than simple case of equal confidence. A larger population size, will affect the convergence of the solution and may decrease the computational efficiency of the algorithm.

3) Experiment 3: In the third experiment, a real-valued multi-label data set is tested and for this purpose "real 6-multiplexer" (RF6) [22] benchmark problem is selected. In this problem for each feature of the vector $x = [x_1, \ldots, x_6]$, if $x_i < \theta$, the feature is interpreted as 0 and 1 otherwise. Therefore, the value of RF_6 is the values of F_6 applied to the bit string. It is assumed that features x_i are scaled to $[0, 1]$. In order to obtain a multi-label data set, the label extension defined in experiment (II) is employed here. A randomly created data set with 150 samples and $\theta = 0.5$ is provided to the algorithm for training and results are compare to SVM and DT as shown in Table (IV).

Results show that the proposed method performs better than SVM with smaller HL value, in addition to the fact that when using SVMs to solve multi-label data a separate model is trained for each class, while MLCS trains only a single model. DT has a better performance with zero hamming loss. Also, the algorithm is performing well in learning the confidence

TABLE IV
LOO CROSS-VALIDATION RESULTS FOR RF_6 PROBLEM.

	HL	EHL	$\bar{\varepsilon}$
MLCS	0.011	0.0205	0.0004
SVM	0.08	-	-
DT	0	-	-

values and the estimation error is very small for the entire population.

VI. CONCLUSION

In this paper action space representation of rules in strength-based learning classifier systems is modified to handle multi-label classification tasks. Moreover, it is assumed that sample-label association is partially known and confidence values are used to represent this membership. During training, classifier learns the degree of confidence and during the test, generates a predicted label set and an average confidence level for each label. To measure the performance of the model, a novel loss function is proposed that is an extension to the well-known Hamming loss measure and considers classification error and confidence estimation error at the same time.

Simulation results show that for the iris data set, which is a multi-class data set, the accuracy of the model is slightly better than the other rule-based classification methods and label confidences are learned with a small error. For the multi-label Boolean and real-valued benchmark data sets, the Hamming loss performance is better than support vector classifiers and confidence levels are learned with small estimation error.

In the future, this work will be extended to more efficient learning classifier algorithms such as UCS, to achieve a better performance and solve data sets with a larger number of samples and features. Moreover, the results of the proposed algorithm in learning partial knowledge in sample-label association, will be compared to other approaches that are able to handle label uncertainty.

ACKNOWLEDGMENT

This work is supported by Air Force Research Laboratory and OSD under agreement number FA8750-15-2-0116.

REFERENCES

[1] P. Vannoorenberghe and T. Denoeux, "Handling uncertain labels in multiclass problems using belief decision trees," in *Proceedings of IPMU*, vol. 3, 2002, pp. 1919–1926.

[2] T. Fenoeux and M. S. Bjanger, "Induction of decision trees from partially classified data using belief functions," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 4. IEEE, 2000, pp. 2923–2928.

[3] Z. Elouedi, K. Mellouli, and P. Smets, "Belief decision trees: theoretical foundations," *International Journal of Approximate Reasoning*, vol. 28, no. 2-3, pp. 91–124, 2001.

[4] E. Côme, L. Oukhellou, T. Denoeux, and P. Aknin, "Learning from partially supervised data using mixture models and belief functions," *Pattern recognition*, vol. 42, no. 3, pp. 334–348, 2009.

[5] T. Denoeux, "Maximum likelihood estimation from uncertain data in the belief function framework," *IEEE Transactions on knowledge and data engineering*, vol. 25, no. 1, pp. 119–130, 2013.

[6] L. Bull, "A brief history of learning classifier systems: from cs-1 to xcs and its variants," *Evolutionary Intelligence*, vol. 8, no. 2-3, pp. 55–70, 2015.

[7] R. J. Urbanowicz and J. H. Moore, "Learning classifier systems: a complete introduction, review, and roadmap," *Journal of Artificial Evolution and Applications*, vol. 2009, p. 1, 2009.

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[9] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 22–30.

[10] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, 2006.

[11] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *European Conference on Machine Learning*. Springer, 2007, pp. 406–417.

[12] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *Granular Computing, 2005 IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 718–721.

[13] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 279–286.

[14] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 1998, pp. 80–91.

[15] R. M. Vallim, D. E. Goldberg, X. Llorà, T. S. Duque, and A. C. Carvalho, "A new approach for multi-label classification based on default hierarchies and organizational learning," in *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*. ACM, 2008, pp. 2017–2022.

[16] S. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, 1995.

[17] E. Bernadó-Mansilla and J. M. Garrell-Guiu, "Accuracy-based learning classifier systems: models, analysis and applications to classification tasks," *Evolutionary computation*, vol. 11, no. 3, pp. 209–238, 2003.

[18] A. Orriols-Puig and E. Bernadó-Mansilla, "A further look at ucs classifier system," *GECCO06*, pp. 8–12, 2006.

[19] R. Urbanowicz, N. Ramanand, and J. Moore, "Continuous endpoint data mining with exstracs: A supervised learning classifier system," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1029–1036.

[20] R. Urbanowicz and J. Moore, "Retooling fitness for noisy problems in a supervised michigan-style learning classifier system," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 591–598.

[21] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a theory of generalization and learning in xcs," *IEEE transactions on evolutionary computation*, vol. 8, no. 1, pp. 28–46, 2004.

[22] S. W. Wilson, "Get real! xcs with continuous-valued inputs," in *Learning Classifier Systems*. Springer, 2000, pp. 209–219.

[23] G. Venturini, "Adaptation in dynamic environments through a minimal probability of exploration," in *Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats 3: from animals to animats 3*. MIT Press, 1994, pp. 371–379.

[24] A. Workineh and A. Homaifar, "Fitness proportionate niching: Maintaining diversity in a rugged fitness landscape," in *Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012, p. 1.

[25] S. M. Weiss and C. A. Kulikowski, *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc., 1991.

[26] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of human genetics*, vol. 7, no. 2, pp. 179–188, 1936.

[27] Y.-C. Hu, R.-S. Chen, and G.-H. Tzeng, "Finding fuzzy classification rules using data mining techniques," *Pattern Recognition Letters*, vol. 24, no. 1, pp. 509–519, 2003.

[28] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Transactions on fuzzy systems*, vol. 3, no. 3, pp. 260–270, 1995.